Resultados

En la tabla 5.50 se muestran los resultados del experimento 21: nitrógeno no eliminado y porcentajes de eliminación en diferentes unidades.

Suma integrada de nitrógeno en la salida (g/d)	
$N\text{-}NH_4^+$	1.188
$N\text{-}NO_3^-$	0.255
$N\text{-}NO_2^-$	0.285
N total	1.728

Eliminación de nitrógeno	
Eliminación de N (g/d)	11.294
% eliminación de N	86.7
% $N\text{-}NH_4^+$ del N total salida	68.7
% $N\text{-}NO_3^-$ del N total salida	14.8
% $N\text{-}NO_2^-$ del N total salida	16.5
mg N eliminado / g SSV d	26.01
mg $N\text{-}NH_4^+$ salida / g SSV d	2.74
mg $N\text{-}NO_3^-$ salida / g SSV d	0.59
mg $N\text{-}NO_2^-$ salida / g SSV d	0.66

Tabla 5.50. Resultados experimento 21

Comentarios

Variables "in-line"

Caudales de recirculación. La relación de recirculación externa utilizada es 0.5 en condiciones normales y 1 en condiciones de alta carga. En la relación de recirculación interna se observan unos valores de 1,2 y 4, aunque aparece mayoritariamente este último.

Temperatura. Se obtienen valores de 19.5 a 23 °C.

pH. Hay variación de pH en el reactor 3 debido al problema con el electrodo. En el reactor 2 aparecen valores en el margen entre 7.25 y 8.00. En el primer reactor, se observan las variaciones provocadas por el cambio de condiciones óxicas. Aparecen incrementos de pH relacionados con las condiciones aeróbicas, y disminuciones de pH por las condiciones anóxicas. En este reactor se obtienen valores entre 7.10 y 7.90. Las perturbaciones en el pH producidas por el cambio de condiciones en el primer reactor pueden observarse también en los reactores 2 y 3, aunque bastante reducidas.

Redox. En el reactor 1 existen oscilaciones entre 0 y –150 mV en condiciones óxicas alternantes, mientras que en condiciones anóxicas se obtiene valores alrededor de –200 mV. En el reactor 2 puede observarse el comportamiento relacionado con las diferentes consignas de oxígeno. Con consignas de 3 mg/l se obtiene valores de redox alrededor de 100 mV. En condiciones anóxicas de 10:00 a 12:00 horas, baja a valores alrededor de 0 mV. Durante el último periodo con consigna 0 mg/l, el redox no baja porque todavía existe oxígeno presente en el medio.

Oxígeno. Se puede apreciar en general el adecuado control, con cierta oscilación por las estimaciones de OUR. En este caso se observa la relación entre las estrategias de control de secuencia anóxica/aeróbica del primer reactor y el aumento de recirculación unido al cambio de consigna del oxígeno en el tercer reactor. Al aumentar la concentración de nitrógeno amoniacal en la salida, el sistema experto inicia las secuencias anóxicas/aeróbicas en el primer reactor. Esto se traduce en un aumento de nitratos en el periodo aeróbico, unido a una menor desnitrificación. Esto provoca un aumento de nitratos en la salida del sistema detectada por el analizador. El sistema experto comprueba que el valor de $N\text{-}NO_x$ en la salida del sistema es alto, por lo que aumenta la recirculación interna a una relación 4 y disminuye la consigna de oxígeno del tercer reactor a 1 mg/l. Cuando finaliza el periodo aeróbico en el reactor 1, el sistema empieza a desnitrificar, por lo que poco después baja la concentración de $N\text{-}NO_x$. Al detectar medidas bajas, el sistema experto devuelve las condiciones de recirculación y oxígeno en el reactor 3 a sus valores iniciales. La aparición de otro ciclo óxico en el primer reactor hace repetir de nuevo el cambio de condiciones.

Aireación. Se observa un consumo de aire elevado en el reactor 1 durante los periodos aeróbicos. En el reactor 2 el consumo es bastante bajo, aunque oscila bastante en función de las condiciones del reactor 1. En el 3 el consumo es más elevado que en anteriores experimentos, sobre todo en condiciones en que el reactor 2 la consigna de oxígeno es 0 mg/l.

Variables analíticas

NH_4^+ salida. Se obtienen valores máximos algo elevados, 7.1 mg/l sobre las 20:00 horas.

NO_3^- salida. El máximo es reducido, 1.4 mg/l a las 1:00 horas. Se observa la oscilación de las medidas provocada por los ciclos de anóxicos/aeróbicos del primer reactor.

NO_2^- salida. Los valores máximos (1.4 mg/l) se presentan sobre las 17:30. También se observa el comportamiento oscilante.

OUR. El consumo es elevado en el reactor 1, ligeramente bajo en el 2 y el 3. El máximo es 0.023 mg O_2 / l s y aparece sobre las 19:30 horas en el reactor 1.

Valoración global

La eliminación de nitrógeno es de 86.7 %, valor elevado. La eliminación por unidad de biomasa es de 26.01 mg N / g SSV d, rendimiento de eliminación elevado. Las concentraciones de nitrógeno amoniacal a la salida son ligeramente altas, pero las de $N\text{-}NO_x$ son reducidas, por lo que globalmente la eliminación es correcta.

EXPERIMENTO 22. Perturbación de caudal

Objetivos

En este experimento, se mantienen todas las estrategias de control, pero se modifica la entrada a la planta. Se introducen dos caudales diferentes durante dos periodos, pero la cantidad de nitrógeno adicionado mediante las bombas dosificadoras se mantiene constante. De este modo se producen dos periodos de diferente caudal pero con igual carga. Los caudales y la dosificación se han calculado de tal modo que el caudal total y el nitrógeno total introducido durante el experimento es igual al de los experimentos anteriores.

Condiciones experimentales

En la tabla 5.51 se muestran las condiciones experimentales de oxígeno, recirculación y entrada total de la planta piloto.

Estado óxico		
Reactor	*Estado*	*Consigna OD*
0	Anaeróbico	0 mg/l
1	Anóxico	0 mg/l / 3 mg/l
2	Aeróbico	0 mg/l / 3 mg/l
3	Aeróbico	1 mg/l / 3 mg/l

Recirculación	
R. Interna	Caudal RI / Caudal Entrada = 1 / 2 / 4
R. Externa	Caudal RE / Caudal Entrada = 0.5 / 1

Suma integrada de nitrógeno en la entrada (g/d)	
$N\text{-}NH_4^+$	7.264
$N\text{-}N_{org}$	5.049
$N\text{-}NO_3^-$	0.709
N total	13.022

Sólidos en suspensión (09:00 horas)	
Reactor	*SSV (mg/l)*
0	4450
1	4450
2	4540
3	4520
Total sistema	418.3 gramos
DSVI	275 ml

Tabla 5.51. Condiciones experimento 22

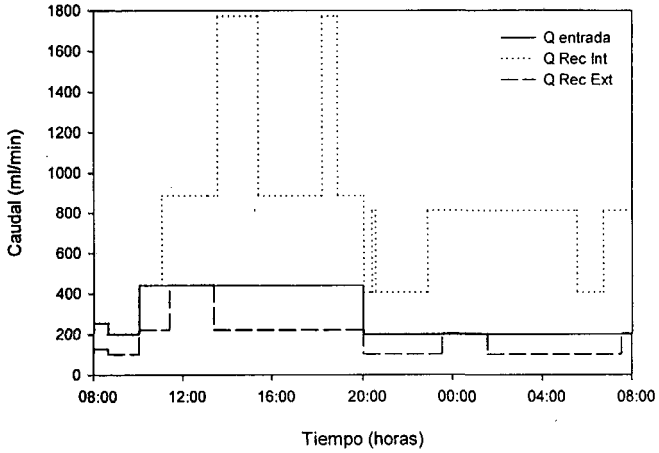
Incidencias de operación

Planta piloto

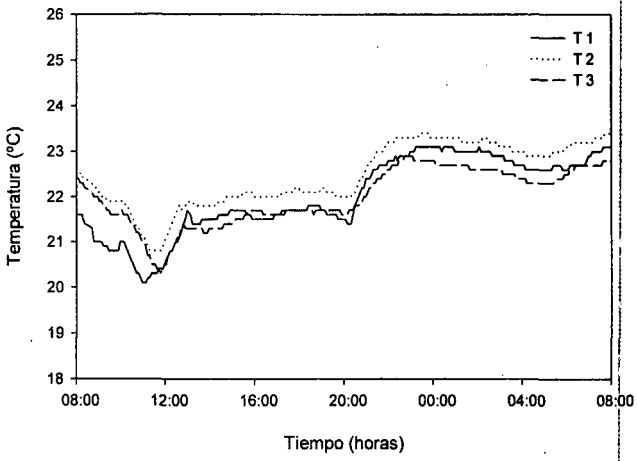
- Parada por escape de agua 11:44 a 12:44 horas.
- El control de oxígeno en los tres reactores funciona adecuadamente durante todo el experimento. Se mantienen correctamente las consignas fijadas en cada reactor. Durante el experimento se mantiene las condiciones de carga constantes, por lo que no se observan periodos de baja carga donde el control de oxígeno no está tan optimizado.
- No se disponen de medida de redox del reactor 3, por mal funcionamiento del electrodo.
- El control de pH funciona correctamente, excepto una pequeña perturbación sobre las 13:00 horas provocada por un exceso de adición de bicarbonato.

Analizadores

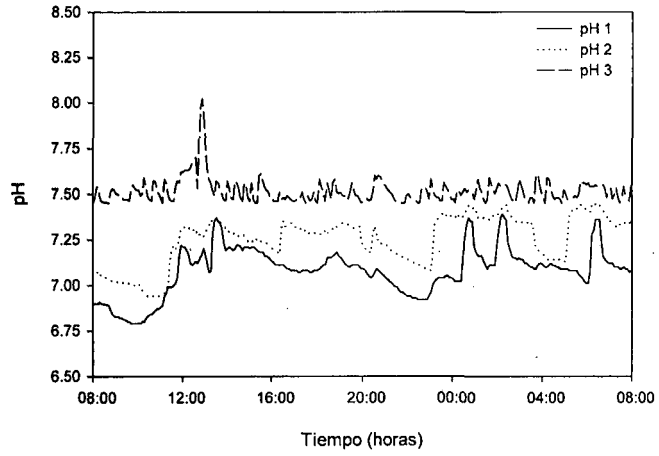
- No se disponen datos de nitrato, nitrito ni amonio de 8:00 a 9:15 horas por recalibración de los analizadores.



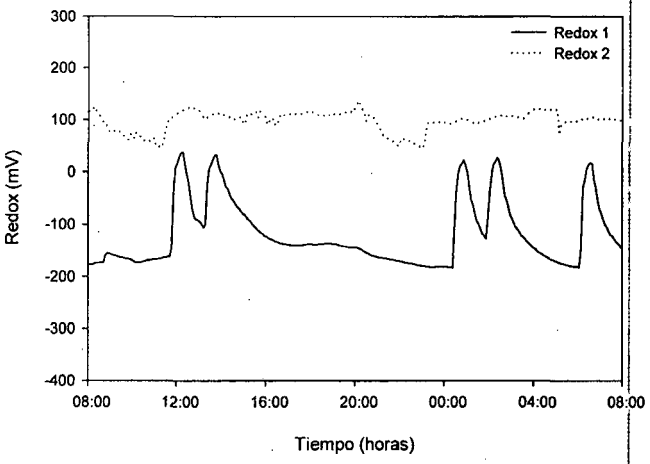
(a) Caudales de entrada, recirculación interna y recirculación externa



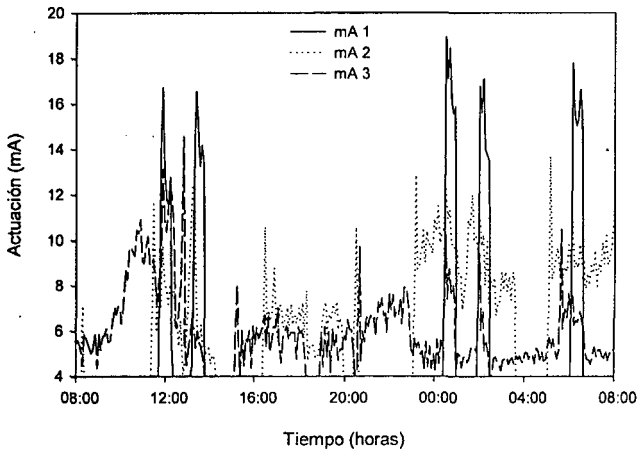
(b) Temperatura en los reactores 1, 2 y 3



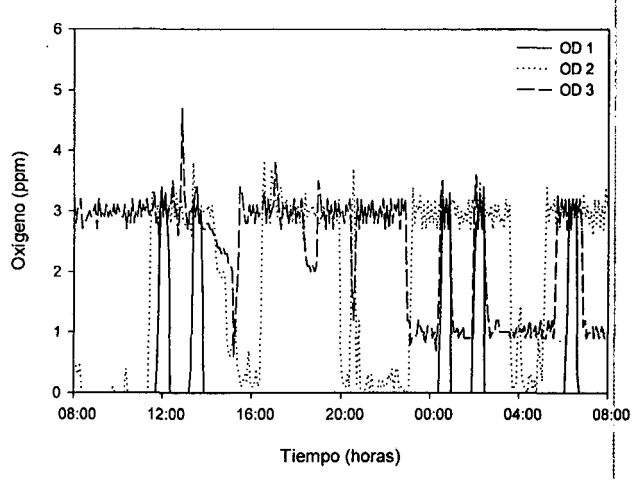
(c) Evolución del pH en los reactores 1, 2 y 3



(d) Evolución del potencial redox en los reactores 1 y 2

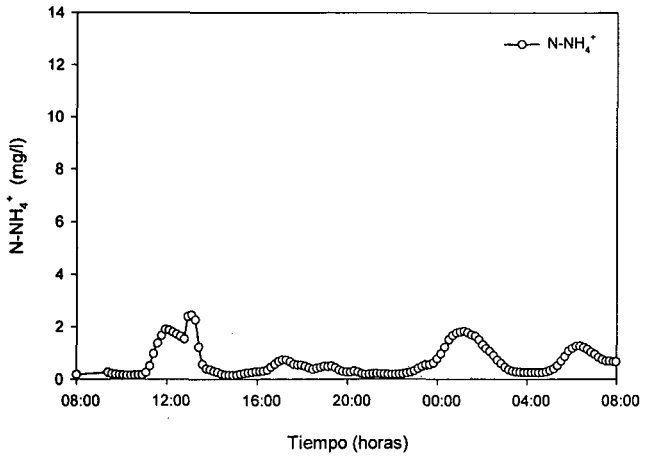


(e) Actuación sobre las válvulas de aireación de los reactores 1, 2 y 3

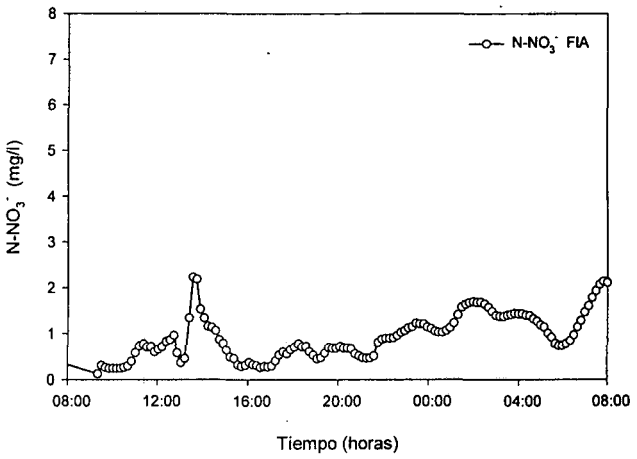


(f) Evolución del oxígeno disuelto en los reactores 1, 2 y 3

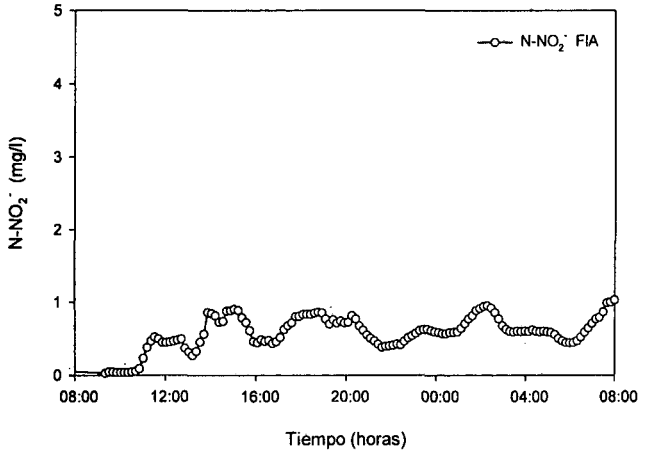
Fig. 5.42. Evolución de los parámetros del proceso. Experimento 22



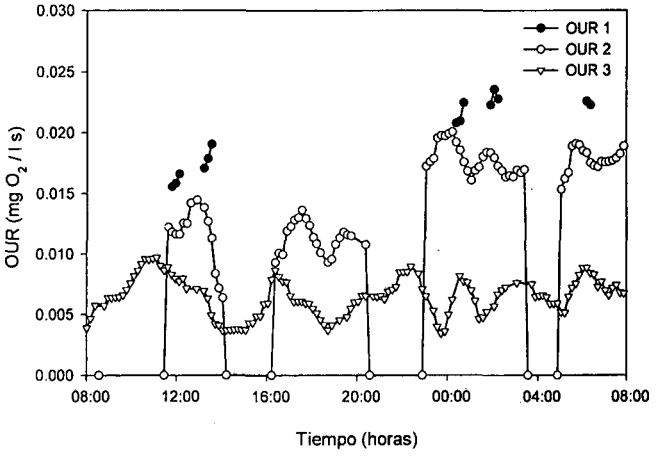
(g) Análisis del nitrógeno amoniacal "on-line" (CFA) de la salida



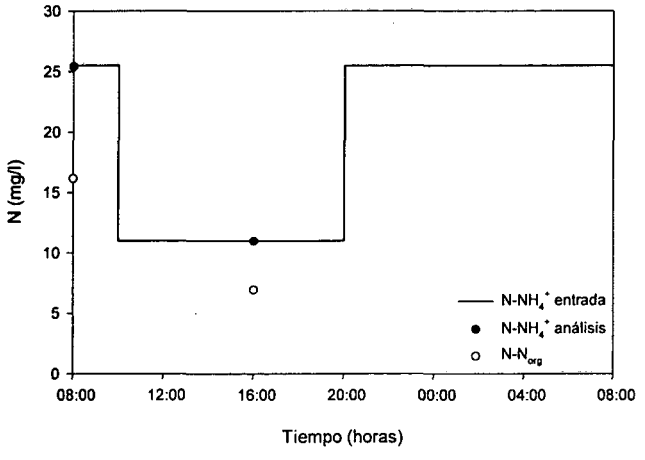
(h) Análisis de nitrato "on-line" (FIA) de la salida



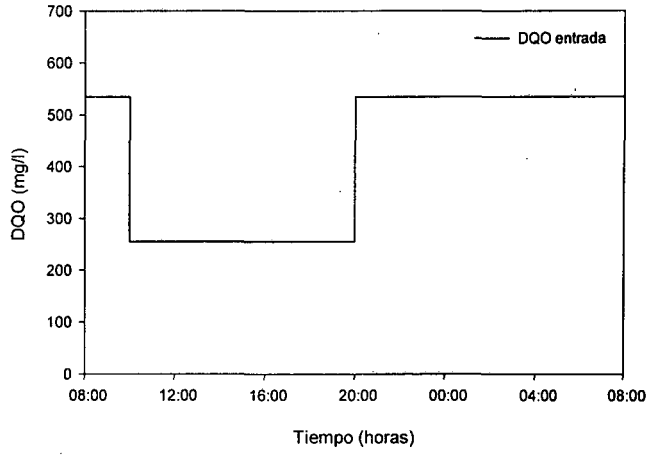
(i) Análisis de nitrito "on-line" (FIA) de la salida



(j) Velocidad de consumo de oxígeno (OUR) en los reactores aeróbicos



(k) Evolución de diferentes formas de nitrógeno en la entrada



(l) Evolución de la DQO de entrada de la planta

Fig. 5.42. Evolución de los parámetros del proceso. Experimento 22

Resultados

En la tabla 5.52 se muestran los resultados del experimento 22: nitrógeno no eliminado y porcentajes de eliminación en diferentes unidades.

Suma integrada de nitrógeno en la salida (g/d)	
$N\text{-}NH_4^+$	0.270
$N\text{-}NO_3^-$	0.351
$N\text{-}NO_2^-$	0.240
N total	0.861

Eliminación de nitrógeno	
Eliminación de N (g/d)	12.161
% eliminación de N	93.4
% $N\text{-}NH_4^+$ del N total salida	31.4
% $N\text{-}NO_3^-$ del N total salida	40.8
% $N\text{-}NO_2^-$ del N total salida	27.8
mg N eliminado / g SSV d	29.07
mg $N\text{-}NH_4^+$ salida / g SSV d	0.65
mg $N\text{-}NO_3^-$ salida / g SSV d	0.84
mg $N\text{-}NO_2^-$ salida / g SSV d	0.57

Tabla 5.52. Resultados experimento 22

Comentarios

Variables "in-line"

Caudales de recirculación. La relación de recirculación externa utilizada generalmente es 0.5, pero en condiciones de alta carga se utiliza una relación de 1. En la relación de recirculación interna se observan diferentes periodos con relación 1, 2 y 4.

Temperatura. Se observan valores de 20 a 23.5 °C.

pH. En el reactor se mantiene en torno a la consigna con oscilaciones mayores que en anteriores experimentos. Esto es debido a la utilización de una solución de carbonato en vez de bicarbonato. En el reactor 2 aparecen valores en el margen entre 7.00 y 7.40. En el primer reactor, se observan las variaciones provocadas por el cambio de condiciones óxicas. Se obtienen valores de pH entre 6.75 y 7.40.

Redox. En el reactor 1 existen oscilaciones entre 25 y −100 mV en condiciones óxicas alternantes, mientras que en condiciones anóxicas, las mayoritarias, se obtiene valores alrededor de −150 mV. En el reactor 2 se observan valores entre 50 y 125 mV, dependiendo de las diferentes consignas de oxígeno. Con consignas de 3 mg/l se obtiene valores de redox alrededor de 125 mV. En condiciones anóxicas baja a valores alrededor de 50 mV.

Oxígeno. Se puede apreciar en general el adecuado control, con la oscilación provocada por las estimaciones de OUR.

Aireación. Se observa un consumo de aire elevado en el reactor 1 durante los periodos aeróbicos. En los reactores 2 y 3 el consumo es bastante reducido, incluso con condiciones aeróbicas de 3 mg/l.

Variables analíticas

NH_4^+ salida. Se obtienen valores máximos muy pequeños, 2.45 mg/l sobre las 13:00 horas. La parada de la planta sobre las 11:45 horas provoca que la concentración de salida no se incremente e incluso disminuya ligeramente. Cuando la planta se vuelve a encender, se recuperan los valores de salida esperados.

NO_3^- salida. El máximo es limitado, 2.2 mg/l a las 13:30 horas. Esta perturbación también está producida por la parada, durante la cual se ha podido incrementar la nitrificación ligeramente.

NO_2^- salida. Los valores máximos (1.0 mg/l) se presentan sobre las 8:00 horas. Se pueden observar unas pequeñas oscilaciones asociadas a la secuencia anóxica/óxica del primer reactor.

OUR. El consumo no es demasiado elevado en el reactor 1 y existen pocos periodos aeróbicos. Tiene valores máximos de 0.024 mg O_2 / l s sobre las 2:00 horas. En el reactor 2 se observa un consumo mayor durante el periodo con menor caudal. En el reactor 3 se mantiene bastante constante y con valores reducidos.

Valoración global

La eliminación de nitrógeno es de 93.4 %, valor muy elevado. La eliminación por unidad de biomasa es de 29.07 mg N / g SSV d, el mayor rendimiento de eliminación obtenido. Las concentraciones de nitrógeno a la salida son muy reducidas. Teniendo en cuenta que la carga total es igual a la de anteriores experimentos, los resultados demuestran la importancia de la adecuada distribución durante todo el periodo para obtener eliminaciones casi totales. Aunque disponer de un depósito de homogeneización es posible para caudales reducidos, no es una opción demasiado económica para una planta de tratamiento de aguas residuales urbanas de tamaño medio.

EXPERIMENTO 23. Baja carga

Objetivos

En este experimento, también se mantienen todas las estrategias de control, pero se modifica la entrada a la planta. Se simula una condición de baja carga de nitrógeno y de DQO. Se introducen dos caudales diferentes durante dos periodos, pero la cantidad de nitrógeno adicionado mediante las bombas dosificadoras se mantiene constante. De este modo se producen dos periodos de diferente caudal pero con igual carga.

Condiciones experimentales

En la tabla 5.53 se muestran las condiciones experimentales de oxígeno, recirculación y entrada total de la planta piloto.

Estado óxico		
Reactor	*Estado*	*Consigna OD*
0	Anaeróbico	0 mg/l
1	Anóxico	0 mg/l / 3 mg/l
2	Aeróbico	0 mg/l / 3 mg/l
3	Aeróbico	1 mg/l / 3 mg/l

Recirculación	
R. Interna	Caudal RI / Caudal Entrada = 1 / 2 / 4
R. Externa	Caudal RE / Caudal Entrada = 0.5 / 1

Suma integrada de nitrógeno en la entrada (g/d)	
$N\text{-}NH_4^+$	3.822
$N\text{-}N_{org}$	2.656
$N\text{-}NO_3^-$	0.562
N total	7.040

Sólidos en suspensión (10:30 horas)	
Reactor	*SSV (mg/l)*
0	3640
1	4570
2	4720
3	4790
Total sistema	427.0 gramos
DSVI	300 ml

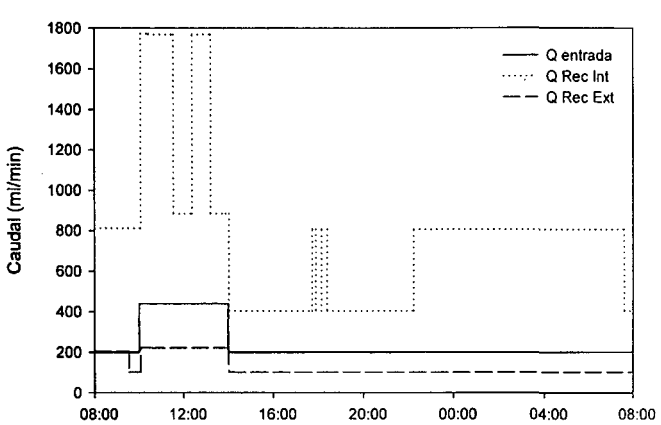
Tabla 5.53. Condiciones experimento 23

Incidencias de operación

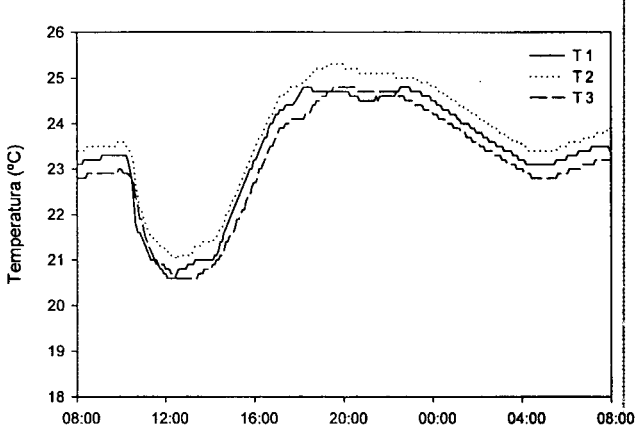
- Parada por escape de agua 18:06 a 18:07 horas.
- El control de oxígeno en los tres reactores funciona adecuadamente durante todo el experimento. Se mantienen correctamente las consignas fijadas en cada reactor, excepto algún periodo de muy baja carga en el reactor 3.
- No se disponen de medida de redox del reactor 3, por mal funcionamiento del electrodo.
- El control de pH funciona correctamente, excepto una perturbación sobre las 18:00 horas. Se adiciona un exceso de carbonato, lo que afecta al pH de los tres reactores.

Analizadores

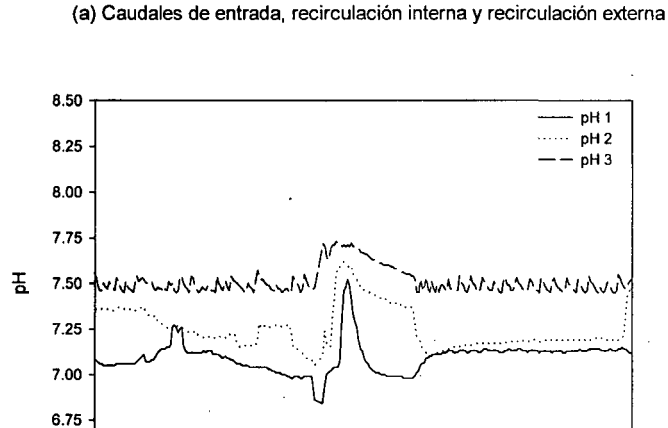
No se disponen datos de nitrato, nitrito ni amonio de 9:45 a 11:00 horas por calibración de los analizadores.



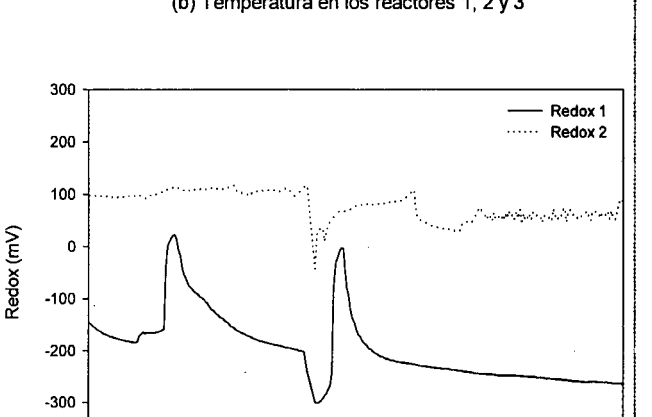
(a) Caudales de entrada, recirculación interna y recirculación externa



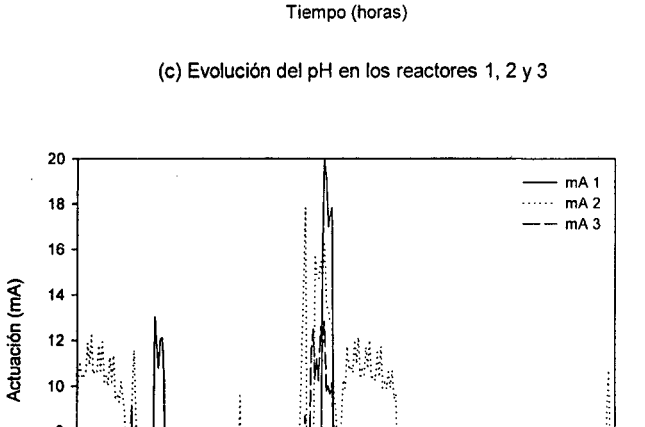
(b) Temperatura en los reactores 1, 2 y 3



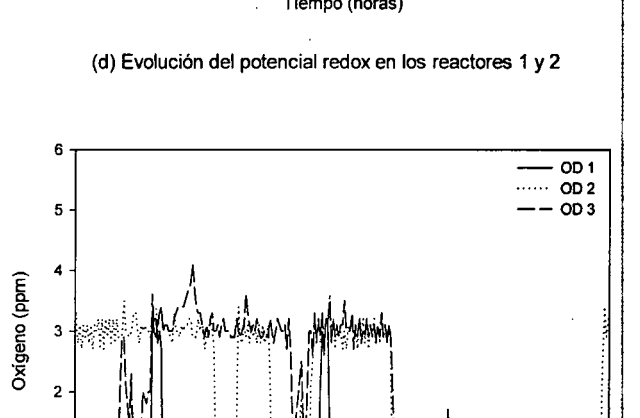
(c) Evolución del pH en los reactores 1, 2 y 3



(d) Evolución del potencial redox en los reactores 1 y 2

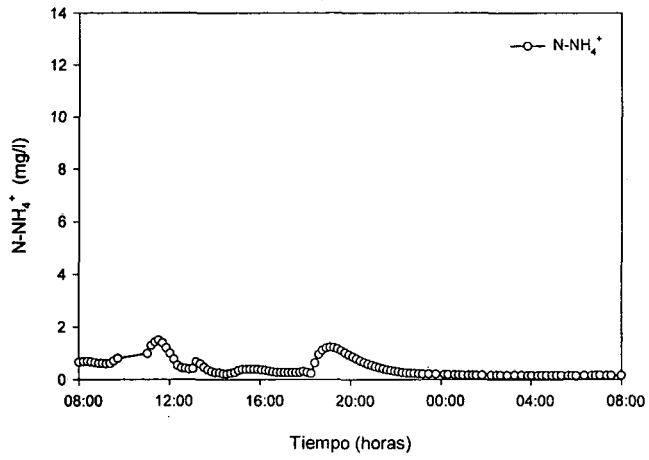


(e) Actuación sobre las válvulas de aireación de los reactores 1, 2 y 3

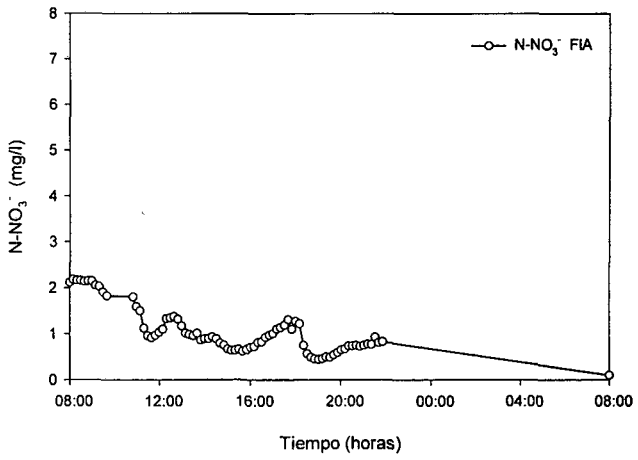


(f) Evolución del oxígeno disuelto en los reactores 1, 2 y 3

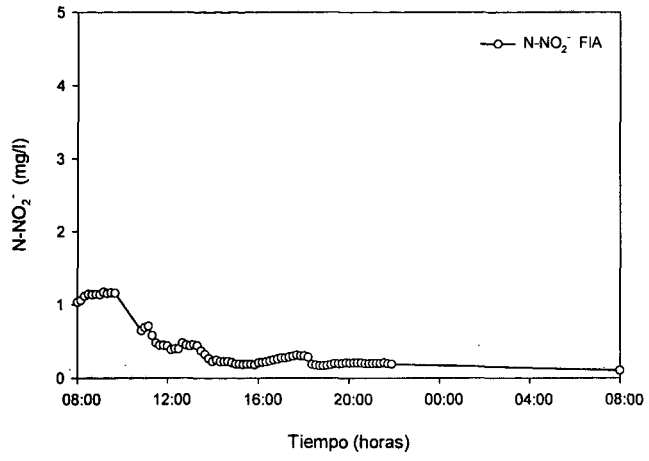
Fig. 5.43. Evolución de los parámetros del proceso. Experimento 23



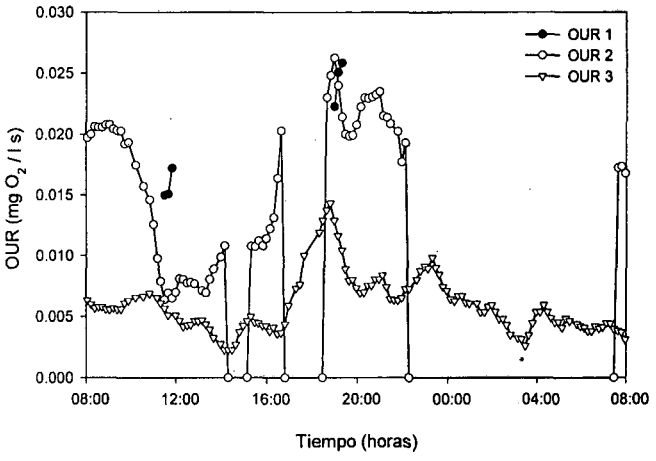
(g) Análisis del nitrógeno amoniacal "on-line" (CFA) de la salida



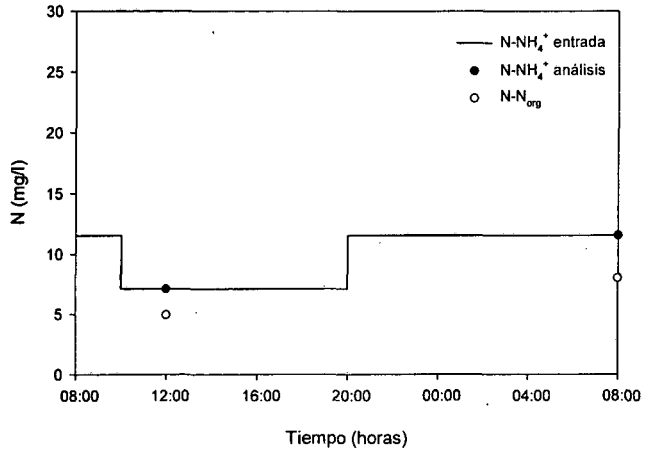
(h) Análisis de nitrato "on-line" (FIA) de la salida



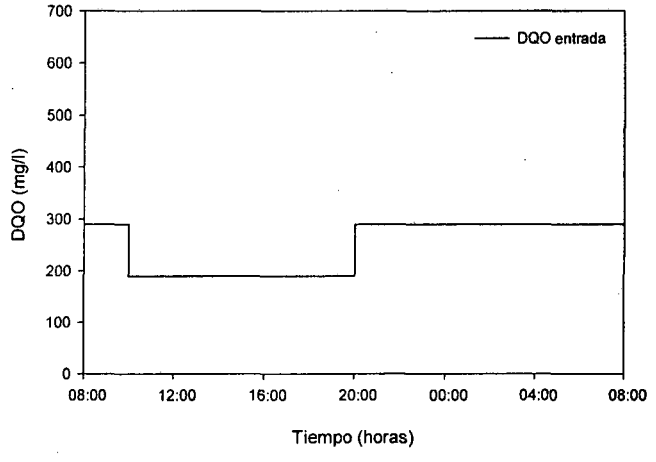
(i) Análisis de nitrito "on-line" (FIA) de la salida



(j) Velocidad de consumo de oxígeno (OUR) en los reactores aeróbicos



(k) Evolución de diferentes formas de nitrógeno en la entrada



(l) Evolución de la DQO de entrada de la planta

Fig. 5.43. Evolución de los parámetros del proceso. Experimento 23

Resultados

En la tabla 5.54 se muestran los resultados del experimento 23: nitrógeno no eliminado y porcentajes de eliminación en diferentes unidades.

Suma integrada de nitrógeno en la salida (g/d)	
$N\text{-}NH_4^+$	0.210
$N\text{-}NO_3^-$	0.397
$N\text{-}NO_2^-$	0.144
N total	0.751

Eliminación de nitrógeno	
Eliminación de N (g/d)	6.289
% eliminación de N	89.3
% $N\text{-}NH_4^+$ del N total salida	27.9
% $N\text{-}NO_3^-$ del N total salida	52.9
% $N\text{-}NO_2^-$ del N total salida	19.2
mg N eliminado / g SSV d	14.73
mg $N\text{-}NH_4^+$ salida / g SSV d	0.49
mg $N\text{-}NO_3^-$ salida / g SSV d	0.93
mg $N\text{-}NO_2^-$ salida / g SSV d	0.34

Tabla 5.54. Resultados experimento 23

Comentarios

Variables "in-line"

Caudales de recirculación. La relación de recirculación externa utilizada es 0.5, ya que no se detectan condiciones de alta carga. En la relación de recirculación interna se observan diferentes periodos con relación 2 y 4.

Temperatura. Se observan valores de 20.5 a 25 °C.

pH. En el reactor se mantiene en torno a la consigna con oscilaciones mayores que en anteriores experimentos, por la utilización de carbonato. En el reactor 2 aparecen valores en el margen entre 7.00 y 7.60. En el primer reactor, se observan valores de pH entre 6.80 y 7.50.

Redox. En el reactor 1 se observan valores entre 25 y –300 mV, lo que refleja las diferentes condiciones óxicas en el reactor. En el reactor 2 se observan valores entre -25 y 100 mV, dependiendo de las diferentes consignas de oxígeno. Con consignas de 3 mg/l se obtiene valores de redox alrededor de 100 mV, mientras que en condiciones anóxicas se llegan a valores negativos.

Oxígeno. Las medidas son correctas, aunque gráficamente no aparecen muy claras porque hay muchos cambios de consigna desde el sistema experto. También se aprecian las oscilaciones provocadas por la estimación de OUR.

Aireación. Se observa un consumo de aire muy reducido. Sólo aparece un consumo elevado en los dos periodos en los que se airea el primer reactor.

Variables analíticas

NH_4^+ salida. Se obtienen valores máximos muy pequeños, 1.5 mg/l sobre las 11:30 horas.

NO_3^- salida. El máximo aparece al principio del experimento, y es provocado por el amonio nitrificado en el anterior experimento, que no se ha eliminado por la falta de materia orgánica. El máximo es 2.2 mg/l sobre las 8:00 horas.

NO_2^- salida. Los valores máximos (1.2 mg/l) se presentan sobre las 9:30 horas. También provienen del anterior experimento.

OUR. El consumo no es elevado en el reactor 1 durante los pocos periodos aeróbicos. En el reactor 2 se observa un consumo en función de la carga de entrada. Sobre las 10:00 horas comienza a bajar al recibir una carga menor. Después de un cambio de anóxico a aeróbico, se incrementa mucho, alcanzando valores máximos de 0.026 mg O_2 / l s sobre las 19:00 horas. En el reactor 3 también se observan variaciones provocadas por los cambios de condiciones de consigna de oxígeno.

Valoración global

La eliminación de nitrógeno es de 89.3 %, valor elevado. La eliminación por unidad de biomasa es de 14.73 mg N / g SSV d, muy reducida porque la carga es muy pequeña. Las concentraciones de nitrógeno a la salida también son mínimas. Este experimento demuestra prácticamente que es posible obtener buenos resultados de eliminación utilizando los recursos necesarios, sin desperdiciar energía de aireación o recirculación cuando la eliminación deseada no lo requiere.

EXPERIMENTO 24. Efecto del sedimentador

Objetivos

Para este experimento se repiten las condiciones de entrada y de control de los experimentos 19, 20 y 21. La diferencia se encuentra en el tamaño del sedimentador, ya que se prueba la respuesta del sistema con un nuevo sedimentador de volumen 25 litros, en contraste con los 60 litros del anterior.

Condiciones experimentales

En la tabla 5.55 se muestran las condiciones experimentales de oxígeno, recirculación y entrada total de la planta piloto.

Estado óxico		
Reactor	*Estado*	*Consigna OD*
0	Anaeróbico	0 mg/l
1	Anóxico	0 mg/l / 3 mg/l
2	Aeróbico	0 mg/l / 3 mg/l
3	Aeróbico	1 mg/l / 3 mg/l

Recirculación	
R. Interna	Caudal RI / Caudal Entrada = 1 / 2 / 4
R. Externa	Caudal RE / Caudal Entrada = 0.5 / 1

Suma integrada de nitrógeno en la entrada (g/d)	
$N\text{-}NH_4^+$	7.264
$N\text{-}N_{org}$	5.049
$N\text{-}NO_3^-$	0.709
N total	13.022

Sólidos en suspensión (8:30 horas)	
Reactor	*SSV (mg/l)*
0	2670
1	2990
2	3060
3	3110
Total sistema	280.5 gramos
DSVI	410 ml

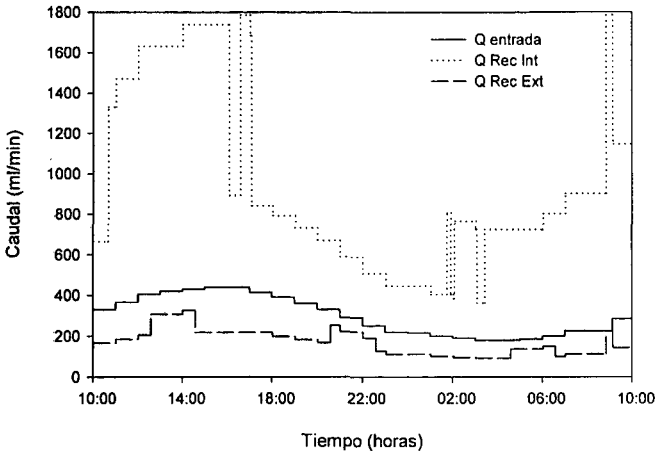
Tabla 5.55. Condiciones experimento 24

Incidencias de operación

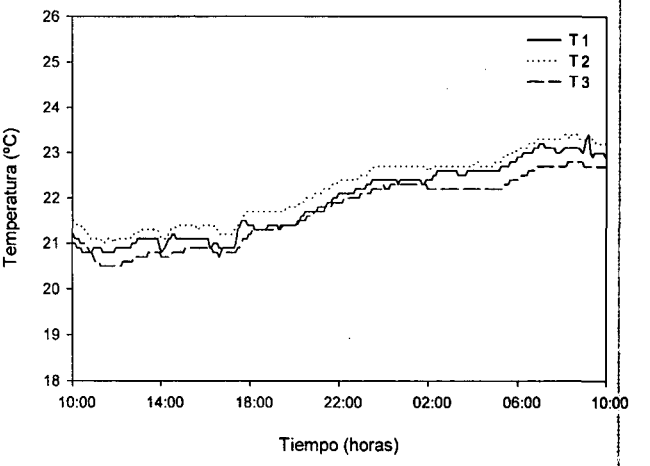
- Parada por alarma de funcionamiento de las bombas de 14:05 a 14:07 horas.
- La instalación del nuevo sedimentador provoca que la concentración de sólidos en la planta sea menor y más variable. Puede observarse como la concentración de sólidos en los reactores, sufre una disminución de 280 a 165 gramos de este experimento al del día siguiente. Esta variabilidad está producida por el peor rendimiento de sedimentación, unida a la menor capacidad de acumulación de lodos.
- La baja concentración de microorganismo en los reactores, provoca que el consumo de oxígeno sea muy reducido. Esto desfavorece al control de oxígeno, que sólo es correcto para el primer reactor. En los reactores 2 y 3 el consumo de aire es mínimo, por lo que el oxígeno está casi siempre por encima de la consigna, incluso con las válvulas de aireación cerradas.
- El control de pH funciona correctamente.

Analizadores

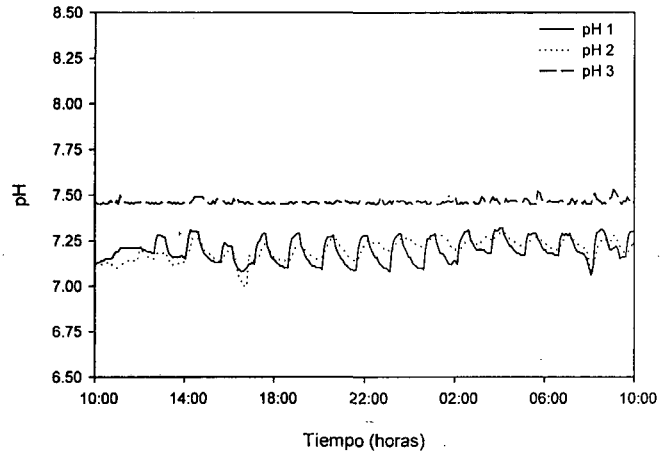
- No aparecen problemas con los analizadores durante el experimento.



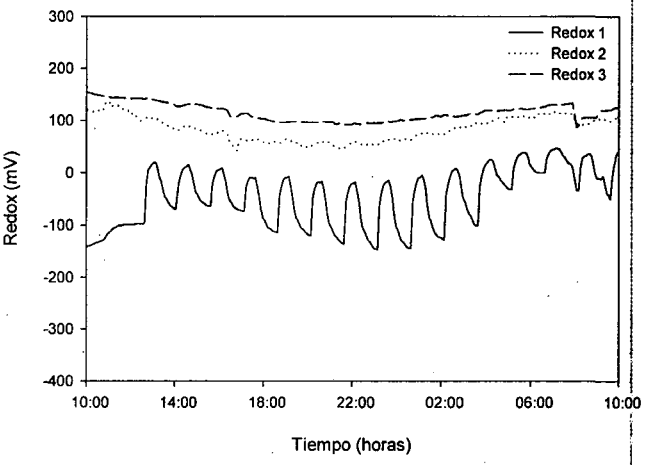
(a) Caudales de entrada, recirculación interna y recirculación externa



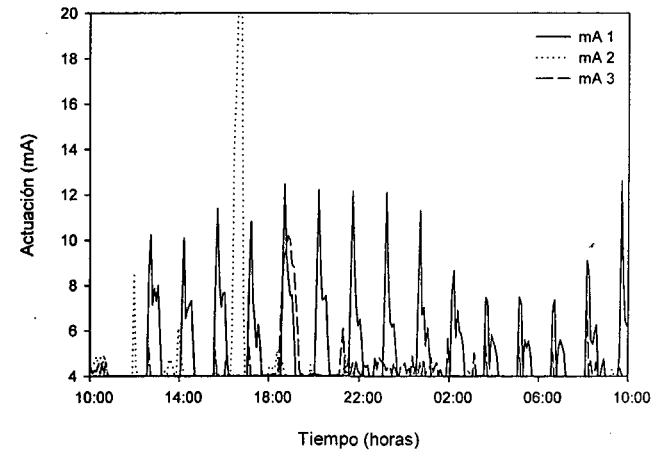
(b) Temperatura en los reactores 1, 2 y 3



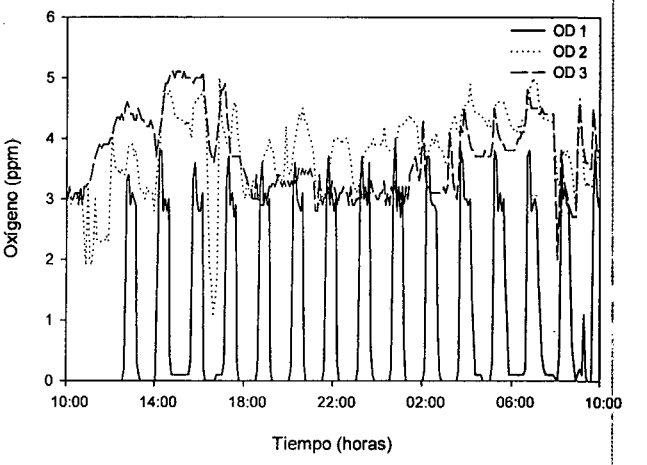
(c) Evolución del pH en los reactores 1, 2 y 3



(d) Evolución del potencial redox en los reactores 1, 2 y 3

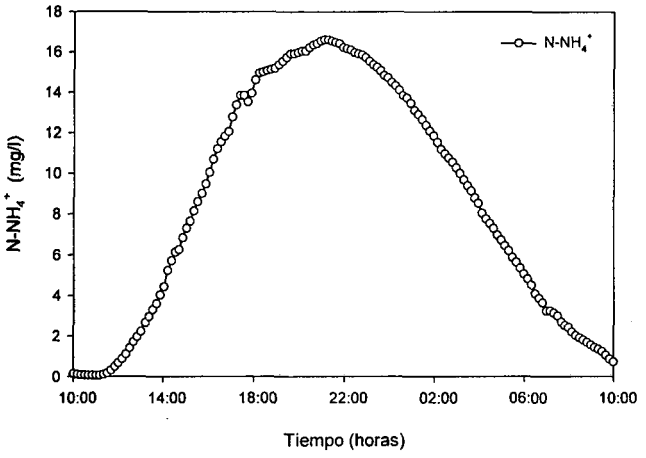


(e) Actuación sobre las válvulas de aireación de los reactores 1, 2 y 3

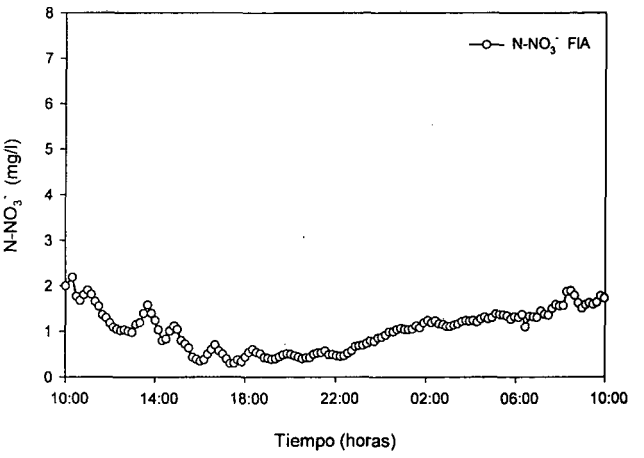


(f) Evolución del oxígeno disuelto en los reactores 1, 2 y 3

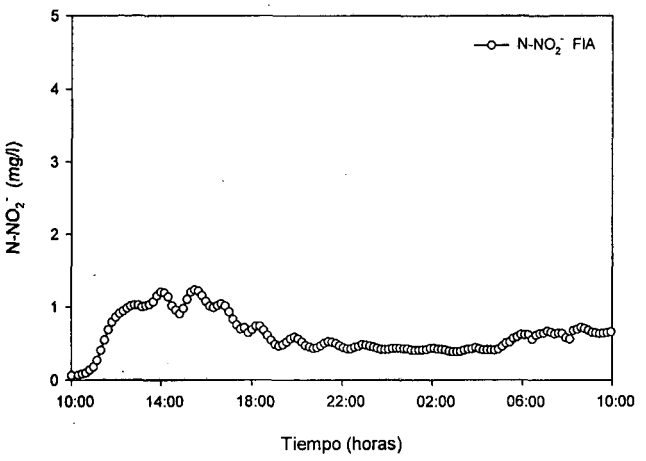
Fig. 5.44. Evolución de los parámetros del proceso. Experimento 24



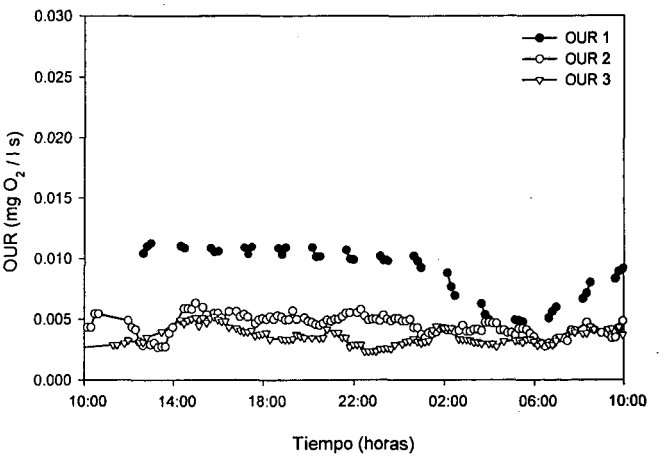
(g) Análisis del nitrógeno amoniacal "on-line" (CFA) de la salida



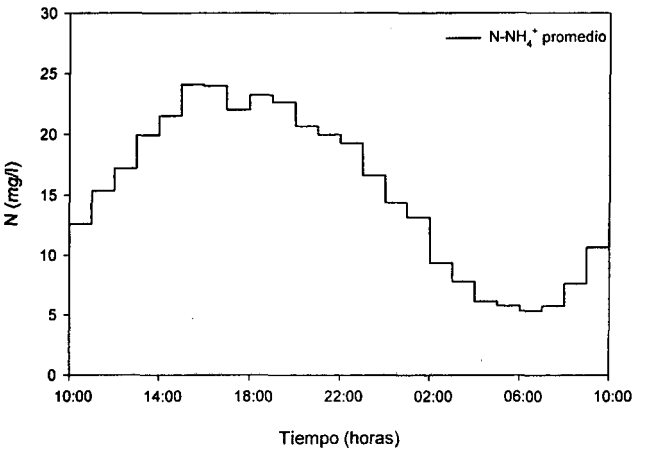
(h) Análisis de nitrato "on-line" (FIA) de la salida



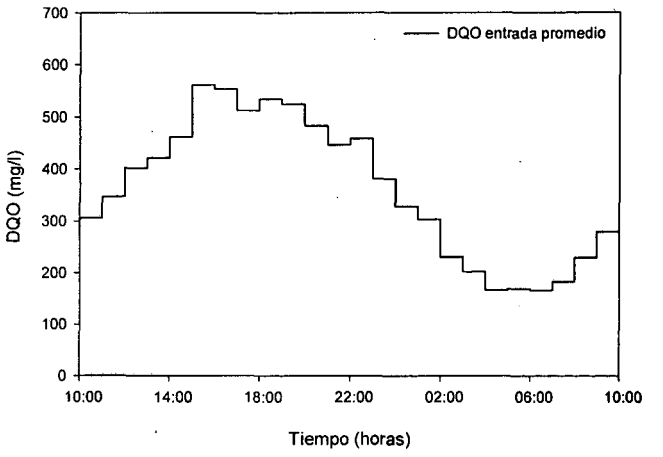
(i) Análisis de nitrito "on-line" (FIA) de la salida



(j) Velocidad de consumo de oxígeno (OUR) en los reactores aeróbicos



(k) Evolución del nitrógeno amoniacal en la entrada



(l) Evolución de la DQO de entrada de la planta

Fig. 5.44. Evolución de los parámetros del proceso. Experimento 24

Resultados

En la tabla 5.56 se muestran los resultados del experimento 24: nitrógeno no eliminado y porcentajes de eliminación en diferentes unidades.

Suma integrada de nitrógeno en la salida (g/d)	
$N\text{-}NH_4^+$	3.713
$N\text{-}NO_3^-$	0.420
$N\text{-}NO_2^-$	0.290
N total	4.423

Eliminación de nitrógeno	
Eliminación de N (g/d)	8.598
% eliminación de N	66.0
% $N\text{-}NH_4^+$ del N total salida	83.9
% $N\text{-}NO_3^-$ del N total salida	9.5
% $N\text{-}NO_2^-$ del N total salida	6.6
mg N eliminado / g SSV d	30.65
mg $N\text{-}NH_4^+$ salida / g SSV d	13.24
mg $N\text{-}NO_3^-$ salida / g SSV d	1.50
mg $N\text{-}NO_2^-$ salida / g SSV d	1.03

Tabla 5.56. Resultados experimento 24

Comentarios

Variables "in-line"

Caudales de recirculación. Las relaciones internas utilizadas son 0.5 y 1. En la relación de recirculación interna se observan diferentes periodos con relación 2 y 4.

Temperatura. Se observan valores de 20.5 a 23.5 °C.

pH. En el reactor se mantiene en torno a la consigna sin oscilaciones. En el reactor 2 aparecen valores en un estrecho margen entre 7.00 y 7.25. En el primer reactor, se observan valores de pH entre 7.00 y 7.25, con oscilaciones por las diferentes condiciones aeróbicas.

Redox. En el reactor 1 se observan valores entre 50 y –150 mV, con oscilaciones por el cambio de condiciones. En el reactor 2 se observan valores entre 50 y 150 mV. En el reactor 3 se miden valores entre 100 y 150 mV.

Oxígeno. Las medidas son mayores que las consignas en los reactores 2 y 3, ya que el control trabaja en la zona donde las válvulas no cierran totalmente. En el reactor 1 los valores son más aceptables.

Aireación. Se observa un consumo de aire muy reducido, incluso en los periodos en los que se airea el reactor 1.

Variables analíticas

NH_4^+ salida. Se obtienen valores máximos muy elevados, 16.6 mg/l sobre las 21:00 horas. Son tan elevados por la escasa nitrificación debido a los bajos sólidos en los reactores.

NO_3^- salida. El máximo es 2.2 mg/l sobre las 10:00 horas. No son muy elevados porque la nitrificación es bastante baja.

NO_2^- salida. Los valores máximos (1.2 mg/l) se presentan sobre las 15:30 horas.

OUR. El consumo es muy reducido en los tres reactores. El máximo de 0.011 mg O_2 / l s se alcanza sobre las 13:00 horas en el reactor 1. En los reactores 2 y 3 se mantiene valores sobre 0.05 mg O_2 / l s durante todo el periodo. Estos consumos están producidos por la baja concentración de microorganismos en los reactores.

Valoración global

La eliminación de nitrógeno es de 66.0 %, valor muy reducido. La eliminación por unidad de biomasa es de 30.65 mg N / g SSV d, muy elevado porque la concentración de microorganismos es escasa. También debe considerarse la alta concentración de amonio a la salida, que supera los límites legales. En conclusión, puede admitirse que la nueva situación de la planta no favorece al porcentaje de eliminación de nitrógeno, aunque proporcionalmente los resultados sean buenos.

EXPERIMENTO 25. Validación del efecto del sedimentador

Objetivos

Este experimento es una repetición del anterior, con las mismas condiciones de entrada y de control.

Condiciones experimentales

En la tabla 5.57 se muestran las condiciones experimentales de oxígeno, recirculación y entrada total de la planta piloto.

Estado óxico		
Reactor	*Estado*	*Consigna OD*
0	Anaeróbico	0 mg/l
1	Anóxico	0 mg/l / 3 mg/l
2	Aeróbico	0 mg/l / 3 mg/l
3	Aeróbico	1 mg/l / 3 mg/l

Recirculación	
R. Interna	Caudal RI / Caudal Entrada = 1 / 2 / 4
R. Externa	Caudal RE / Caudal Entrada = 0.5 / 1

Suma integrada de nitrógeno en la entrada (g/d)	
$N\text{-}NH_4^+$	7.264
$N\text{-}N_{org}$	5.049
$N\text{-}NO_3^-$	0.709
N total	13.022

Sólidos en suspensión (8:30 horas)	
Reactor	*SSV (mg/l)*
0	1790
1	1780
2	1790
3	1750
Total sistema	165.1 gramos
DSVI	365 ml

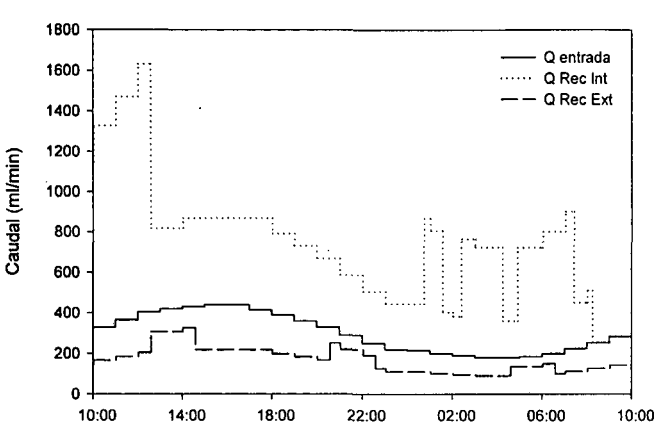
Tabla 5.57. Condiciones experimento 25

Incidencias de operación

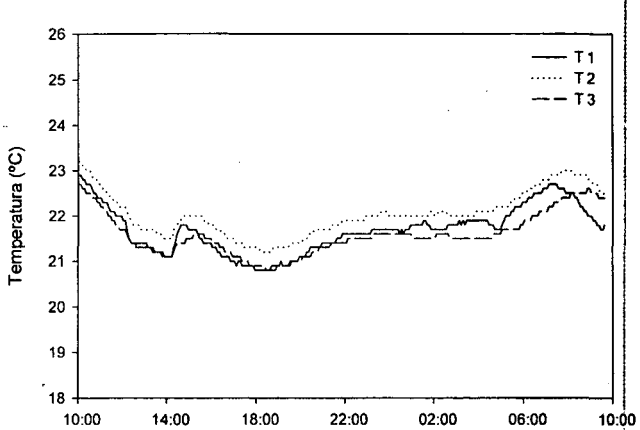
- Parada por alarma de funcionamiento de las bombas de 14:05 a 14:22 horas.
- La concentración de microorganismos en los reactores es bastante reducida y tiene un comportamiento inestable.
- La baja concentración de microorganismo en los reactores, provoca que el consumo de oxígeno sea muy reducido, lo que desfavorece su control.
- El control de pH funciona correctamente.

Analizadores

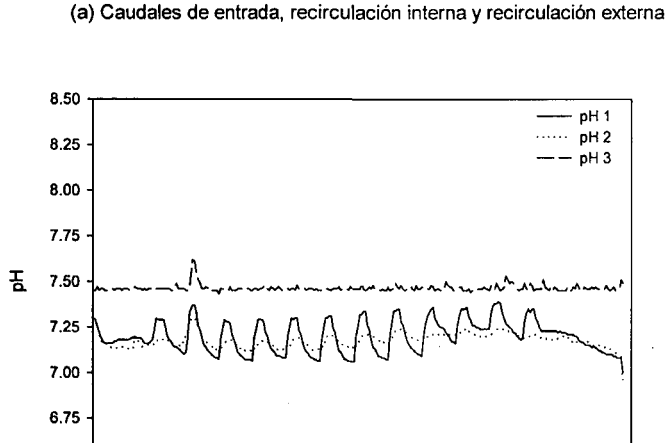
- No aparecen problemas con los analizadores durante el experimento.



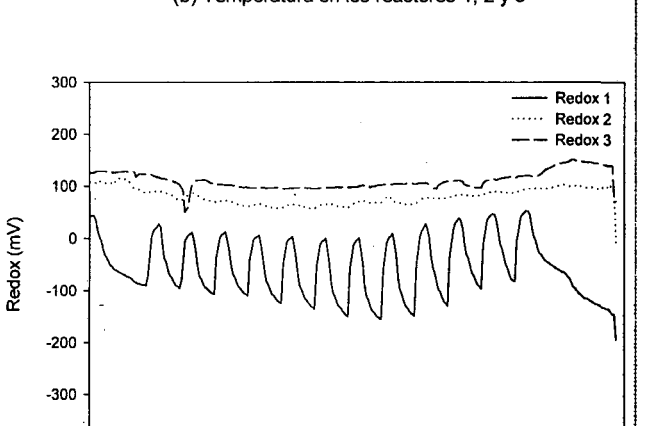
(a) Caudales de entrada, recirculación interna y recirculación externa



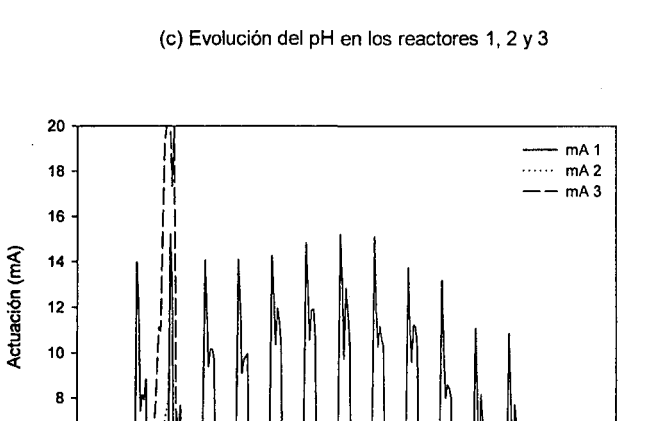
(b) Temperatura en los reactores 1, 2 y 3



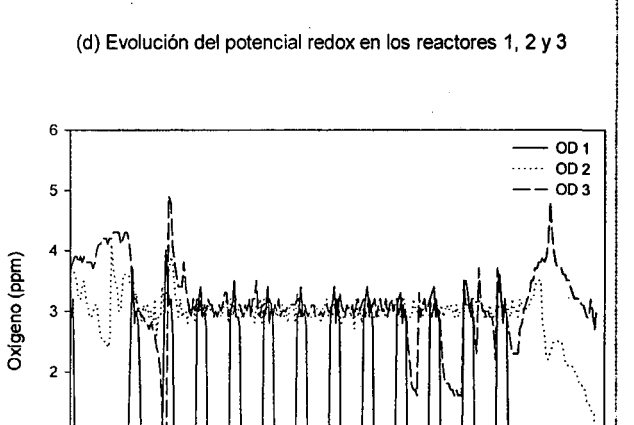
(c) Evolución del pH en los reactores 1, 2 y 3



(d) Evolución del potencial redox en los reactores 1, 2 y 3

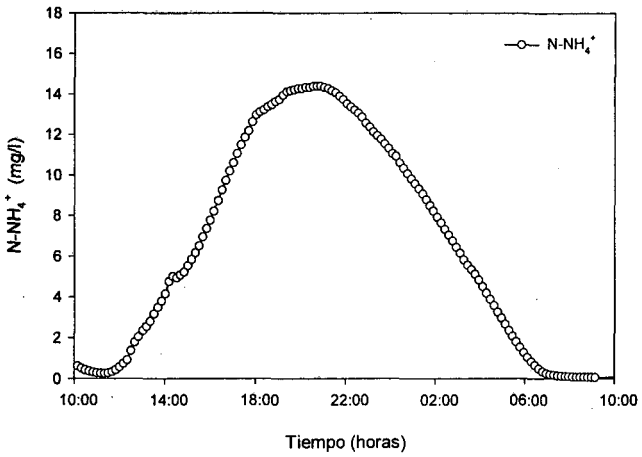


(e) Actuación sobre las válvulas de aireación de los reactores 1, 2 y 3

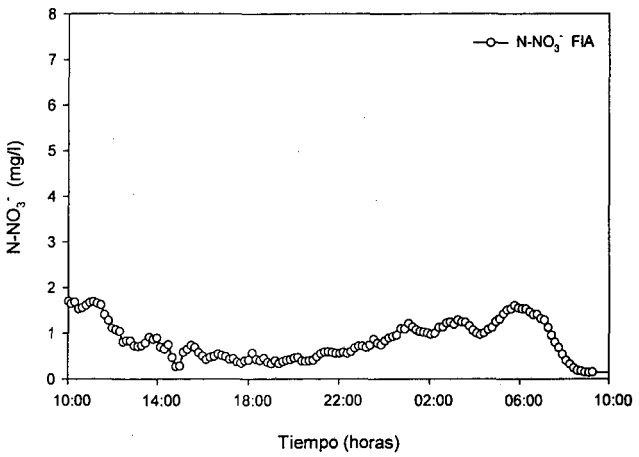


(f) Evolución del oxígeno disuelto en los reactores 1, 2 y 3

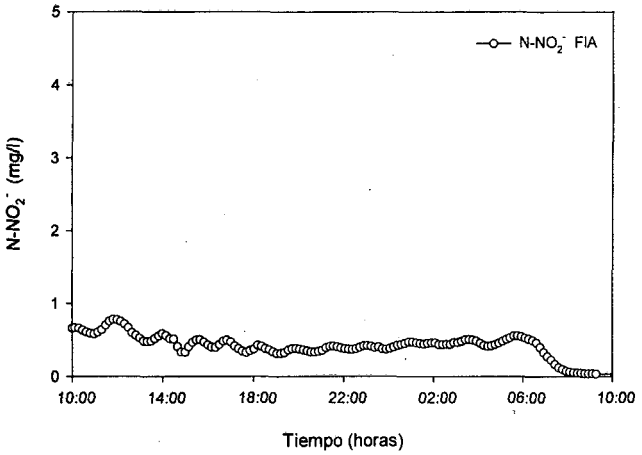
Fig. 5.45. Evolución de los parámetros del proceso. Experimento 25



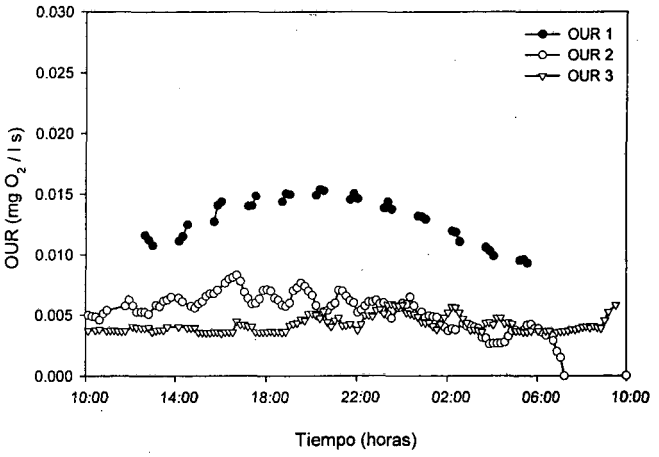
(g) Análisis del nitrógeno amoniacal "on-line" (CFA) de la salida



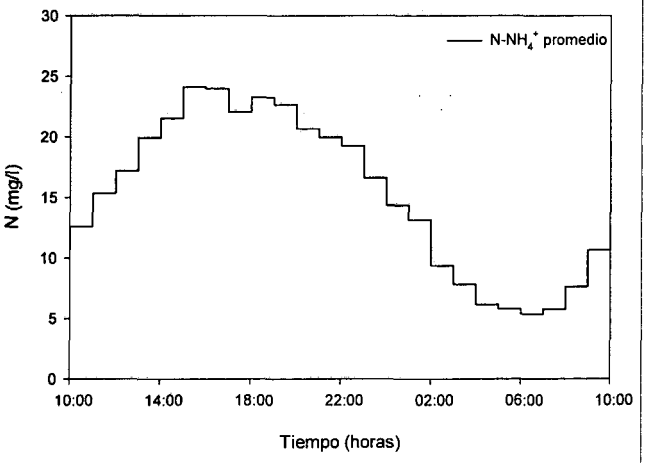
(h) Análisis de nitrato "on-line" (FIA) de la salida



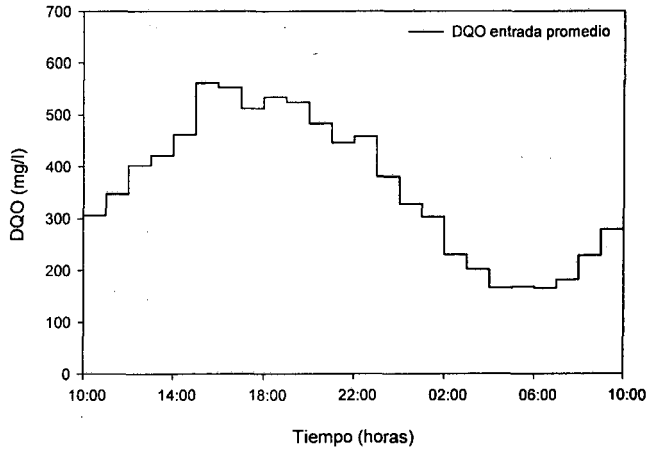
(i) Análisis de nitrito "on-line" (FIA) de la salida



(j) Velocidad de consumo de oxígeno (OUR) en los reactores aeróbicos



(k) Evolución del nitrógeno amoniacal en la entrada



(l) Evolución de la DQO de entrada de la planta

Fig. 5.45. Evolución de los parámetros del proceso. Experimento 25

Resultados

En la tabla 5.58 se muestran los resultados del experimento 25: nitrógeno no eliminado y porcentajes de eliminación en diferentes unidades.

Suma integrada de nitrógeno en la salida (g/d)	
$N\text{-}NH_4^+$	2.915
$N\text{-}NO_3^-$	0.331
$N\text{-}NO_2^-$	0.182
N total	3.428

Eliminación de nitrógeno	
Eliminación de N (g/d)	9.5942
% eliminación de N	73.7
% $N\text{-}NH_4^+$ del N total salida	85.0
% $N\text{-}NO_3^-$ del N total salida	9.7
% $N\text{-}NO_2^-$ del N total salida	5.3
mg N eliminado / g SSV d	58.12
mg $N\text{-}NH_4^+$ salida / g SSV d	17.66
mg $N\text{-}NO_3^-$ salida / g SSV d	2.01
mg $N\text{-}NO_2^-$ salida / g SSV d	1.10

Tabla 5.58. Resultados experimento 25

Comentarios

Variables "in-line"

Caudales de recirculación. Las relaciones internas utilizadas son 0.5 y 1. En la relación de recirculación interna se observan diferentes periodos con relación 2 y 4.

Temperatura. Se observan valores de 20.5 a 23.5 °C.

pH. En el reactor se mantiene en torno a la consigna sin oscilaciones. En el reactor 2 aparecen valores en un margen entre 7.10 y 7.25. En el primer reactor, se observan valores de pH entre 7.00 y 7.25, con oscilaciones por las diferentes condiciones aeróbicas.

Redox. En el reactor 1 se observan valores entre 50 y –150 mV, con oscilaciones por el cambio de condiciones. En el reactor 2 se observan valores alrededor de 100 mV. En el reactor 3 se registran valores alrededor de 150 mV.

Oxígeno. Se mantiene por encima de la consigna en los reactores 2 y 3 durante algunos periodos de baja carga. En el reactor 1 se mantiene correctamente.

Aireación. Se observa un consumo de aire muy reducido, aunque ligeramente superior al del anterior experimento. El consumo en el primer reactor también es superior al anterior.

Variables analíticas

NH_4^+ salida. Se obtienen valores máximos muy elevados, 14.4 mg/l sobre las 21:00 horas. Son tan elevados por la escasa nitrificación.

NO_3^- salida. El máximo es 1.7 mg/l sobre las 8:00 horas

NO_2^- salida. Los valores máximos (0.8 mg/l) se presentan sobre las 12:00 horas.

OUR. El consumo es muy reducido en los tres reactores, aunque ligeramente superior al del anterior experimento. El máximo de 0.015 mg O_2 / l s se alcanza sobre las 20:30 horas en el reactor 1. En los reactores 2 y 3 se mantiene valores sobre 0.05 mg O_2 / l s durante todo el periodo.

Valoración global

La eliminación de nitrógeno es de 73.7 %, valor reducido pero mejor al anterior. La eliminación por unidad de biomasa es de 58.12 mg N / g SSV d, demasiado elevado y, por tanto, algo sospechosa. La concentración de microorganismos varía a lo largo del experimento, debido al mal funcionamiento del sedimentador. Una posible explicación es la acumulación de biomasa en el sedimentador para caudales bajos de recirculación externa (correspondientes al inicio del experimento) y su posterior recirculación al aumentar la recirculación externa cuando se detecta el aumento de carga en el sistema. También es posible que el nitrógeno orgánico, que normalmente desaparece por completo, no se haya hidrolizado totalmente, lo que no estaríamos considerando cuando calculamos el rendimiento.

5.2.7 ANÁLISIS DE RESULTADOS

En primer lugar se representan los resultados de eliminación de nitrógeno obtenidos para los experimentos 1 a 23, realizados con la planta piloto en las mismas condiciones. Los experimentos 24 y 25 son los efectuados con el nuevo sedimentador de menor volumen y se desvían mucho del comportamiento de los anteriores, por lo que son tratados aparte.

La figura 5.46 muestra los resultados de porcentaje de eliminación en los experimentos 1 a 23. Estos valores permiten la comparación entre los experimentos, pero no se está considerando las diferentes concentraciones de sólidos en los reactores. Esta es una variación importante, que se debe tener en cuenta para obtener resultados coherentes.

En la figura 5.47 se muestran los valores analizados de sólidos volátiles en los reactores, utilizados para calcular la eliminación de nitrógeno por unidad de biomasa. En la representación de estos datos no hay una evolución de los sólidos en el tiempo, sino que sólo es indicativo de la variación entre cada experimento. Durante el periodo experimental con el sedimentador inicial, se obtienen valores entre 400 y 650 gramos de biomasa, pero después del cambio de sedimentador (experimentos 24 y 25) la biomasa disminuye en gran medida, provocado por la menor capacidad de separación del nuevo sedimentador.

En la figura 5.48 se representa la eliminación de nitrógeno en miligramos por gramo de SSV en los reactores y día. Este parámetro sí permite la comparación entre experimentos con diferentes condiciones de sólidos en los reactores.

Por último, se representa en la figura 5.49 los mg de nitrógeno en forma de amonio, nitrato o nitrito que aparece en la salida por gramo de SSV y día. Con estos parámetros se obtiene un valor comparativo, entre experimentos con diferente concentración de microorganismos, de la fracción de cada especie del nitrógeno que no se elimina.

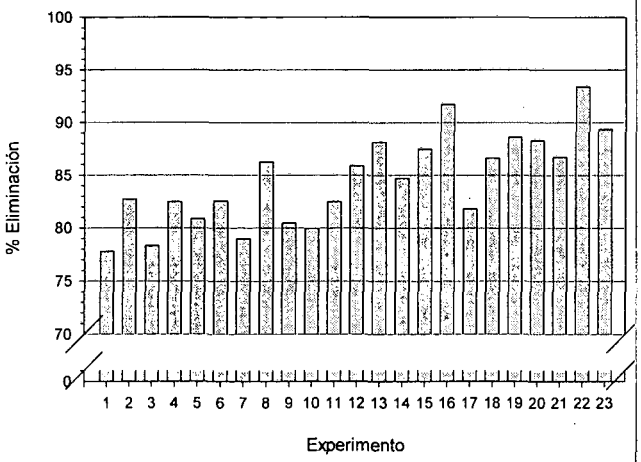


Fig.5.46. Porcentaje de eliminación de nitrógeno

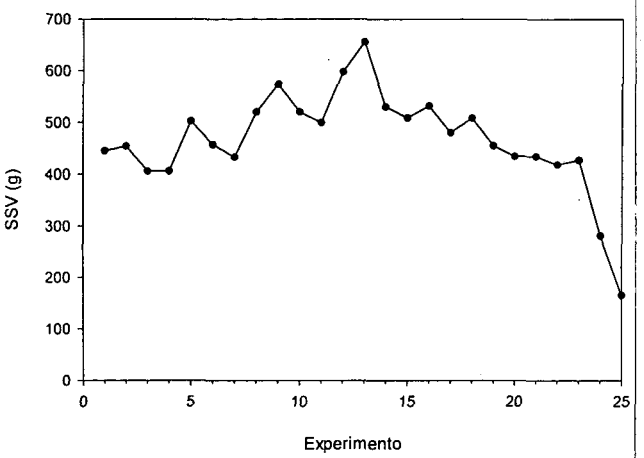


Fig.5.47. Sólidos volátiles totales en el sistema

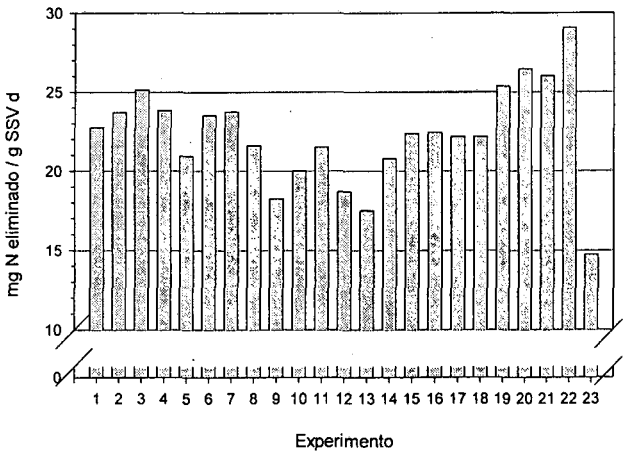


Fig.5.48. mg N eliminados / g SSV d

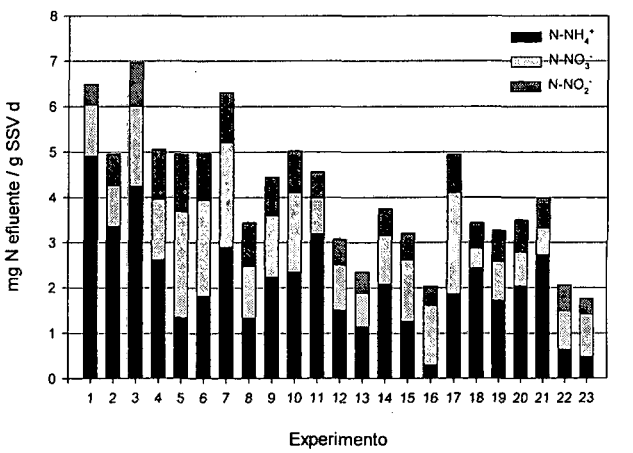


Fig.5.49. mg N en el efluente / g SSV d

5.2.7.1 EFECTO DE LA RELACIÓN DE RECIRCULACIÓN EXTERNA

En los experimentos 1 y 2 se estudia la respuesta de la planta piloto a la entrada variable en composición y caudal, comparando la eliminación obtenida utilizando una recirculación externa de caudal constante Q = 150 ml / min (experimento 1) con la obtenida a una relación de recirculación constante $Q_{RE}/Q_E = 0.5$ (experimento 2). El mantenimiento de la relación de recirculación constante debe permitir la recirculación de lodos adecuada para tratar el caudal de entrada de la planta, lo que favorece el mantenimiento de una adecuada relación alimento/microorganismos.

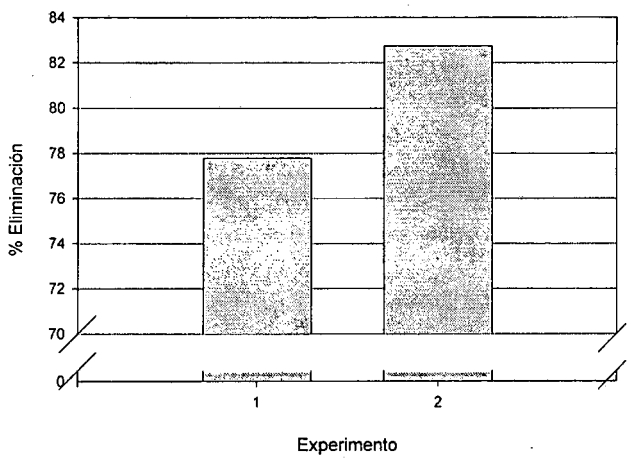


Fig. 5.50. Porcentaje de eliminación de nitrógeno

En el primer experimento se observa que, a pesar de ser las condiciones con menor control, la eliminación de nitrógeno es cercana al 78 % (figura 5.50). Si la entrada fuera constante la eliminación hubiera sido superior, pero el aumento de caudal y concentración del perfil de entrada, provoca que el sistema no sea capaz de transformar todo el nitrógeno de entrada durante el periodo de máxima carga. Vista en porcentaje, la eliminación es satisfactoria, pero como concentración de nitrógeno a la salida, la eliminación no es adecuada, ya que se obtienen niveles mayores de 12mg/l, lo que no cumple los límites legislados.

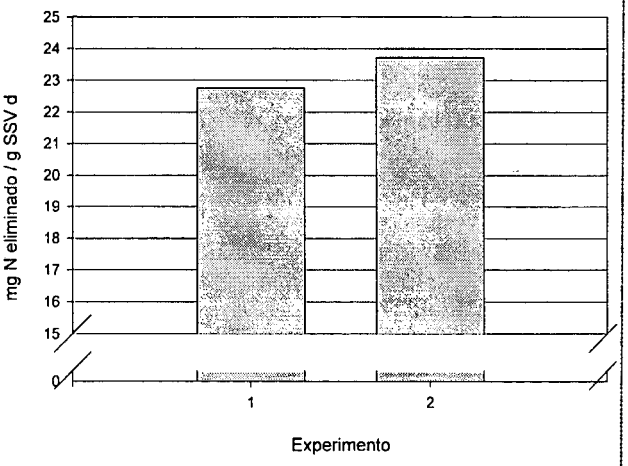


Fig. 5.51. mg N eliminados / g SSV d

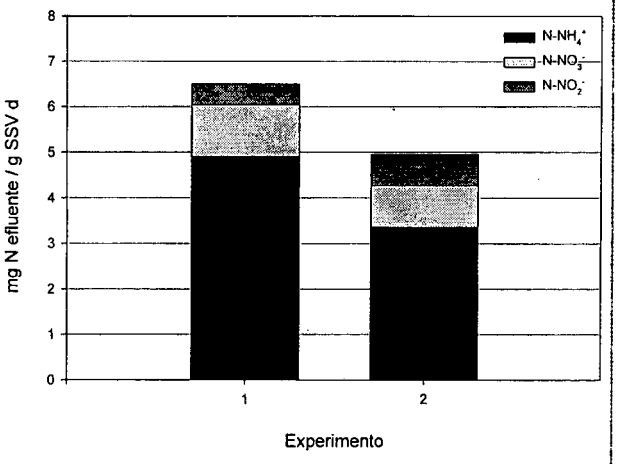


Fig. 5.52. mg N en el efluente / g SSV d

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
1	Anóxico	Aeróbico 3	Aeróbico 3	Q=150 (ml/min)	2
2	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
1	445.2	2.186	0.503	0.204	2.893	10.129	77.78	75.57	17.39	7.04	22.75	4.91	1.13	0.46
2	454.3	1.532	0.410	0.307	2.249	10.773	82.73	68.12	18.21	13.67	23.71	3.37	0.90	0.68

Tabla 5.59. Condiciones experimentales y resultados de los experimentos 1 y 2

En el segundo experimento se introduce la estrategia de control de la relación de recirculación constante, lo que supone una ligera mejora en la eliminación de nitrógeno, que se acerca en este caso al 83 %. También se presenta una mejora en los niveles de vertido, que disminuyen a 10 mg/l de nitrógeno amoniacal.

Si se calcula la mejora de eliminación en función de los mg N eliminados por gramo SSV y día (figura 5.51), se obtiene una mejora del 4.2 %.

El nitrógeno a la salida, calculado en forma de mg de N que quedan por unidad de biomasa y día (figura 5.52), disminuye en torno al 24 %. La composición del nitrógeno de salida también varía, aumentando el nitrito en detrimento del amonio, lo que significa que la nitrificación se ve ligeramente mejorada.

5.2.7.2 EFECTO DE LOS CICLOS ANÓXICOS - AERÓBICOS EN EL REACTOR 1

En los experimentos 3, 4, 5, 6 y 7 se estudia la respuesta del sistema a un cambio cíclico de condiciones aeróbicas y anóxicas en el reactor 1, para comprobar si se puede aprovechar su exceso de capacidad desnitrificadora para nitrificar. En los experimentos se utiliza diferente relación de tiempo anóxico/aeróbico y diferente duración del ciclo total. En todos ellos se mantiene una consigna de oxígeno en los reactores 2 y 3 de 3 mg/l, una relación de recirculación interna de 2 y una relación de recirculación externa de 0.5.

En la figura 5.53 se representa para estos experimentos el porcentaje de eliminación de nitrógeno. En las figuras 5.54 y 5.55 se representa el nitrógeno eliminado y el nitrógeno en el efluente, calculados por unidad de biomasa en el sistema.

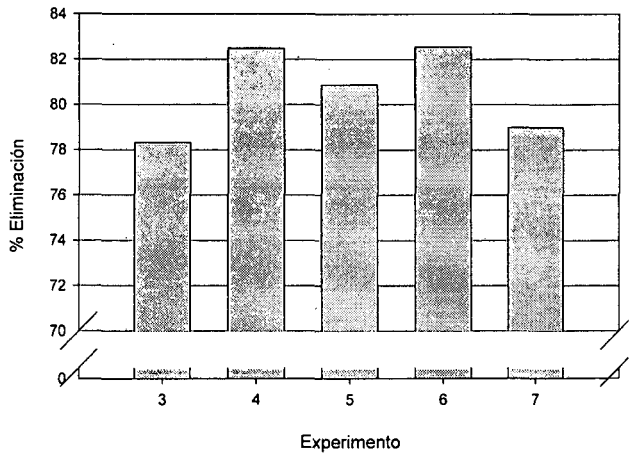


Fig. 5.53. Porcentaje de eliminación de nitrógeno

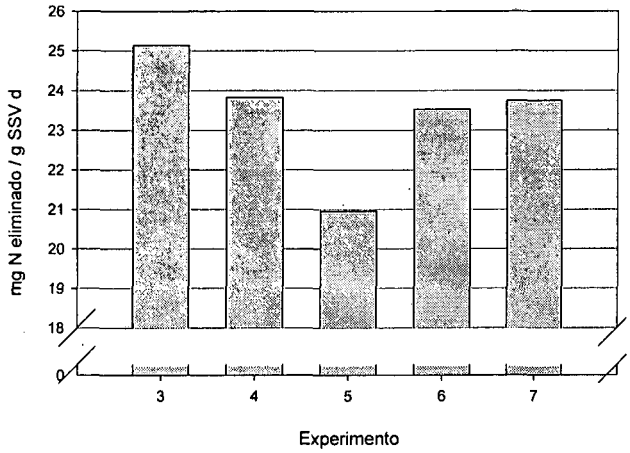


Fig. 5.54. mg N eliminados / g SSV d

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
3	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2
4	20'aer-40'anox	Aeróbico 3	Aeróbico 3	0.5	2
5	30'aer-30'anox	Aeróbico 3	Aeróbico 3	0.5	2
6	30'aer-60'anox	Aeróbico 3	Aeróbico 3	0.5	2
7	10'aer-20'anox	Aeróbico 3	Aeróbico 3	0.5	2

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
3	405.7	1.727	0.711	0.386	2.824	10.199	78.32	61.17	25.16	13.67	25.14	4.26	1.75	0.95
4	450.9	1.187	0.599	0.496	2.282	10.741	82.48	52.03	26.24	21.73	23.82	2.63	1.33	1.10
5	502.6	0.684	1.172	0.634	2.490	10.532	80.88	27.45	47.07	25.48	20.96	1.36	2.33	1.26
6	457.0	0.837	0.966	0.468	2.271	10.751	82.56	36.87	42.52	20.61	23.53	1.83	2.11	1.02
7	433.1	1.261	0.998	0.478	2.737	10.285	78.98	46.07	36.46	17.48	23.75	2.91	2.30	1.10

Tabla 5.60. Condiciones experimentales y resultados de los experimentos 3 a 7

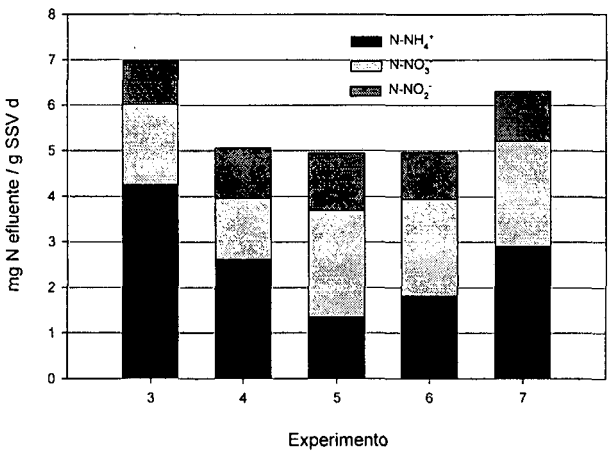


Fig. 5.55. mg N en el efluente / g SSV d

El experimento 3 es una repetición de las condiciones del número 2, pero se realiza para tener una referencia actualizada para el resto de experimentos de la serie. En él se observa una eliminación de nitrógeno del 78.3 %, pero teniendo en cuenta la biomasa del sistema, la eliminación sube a 25.1 mg N/ g SSV d. El nivel de vertido máximo de amonio se mantiene sobre los 10 mg/l.

En el número 4 la eliminación de nitrógeno es del 82.5 %, y, considerando eliminación por unidad de biomasa, es de 23.8 mg N/ g SSV d. Es una eliminación inferior, pero debe considerarse que la biomasa medida en el experimento 3 era inferior a la normal, por lo que puede producir resultados diferentes. Se observa una mejora en los niveles de vertido por unidad de biomasa, además de variar su composición, ya que la presencia de amonio en la salida es un 10% inferior. También se observa una disminución a 8 mg/l de la concentración máxima de amonio en el efluente. Los resultados de este experimento deben considerarse cuidadosamente, ya que durante las últimas horas la planta estuvo parada, y las concentraciones utilizadas durante ese periodo fueron estimadas siguiendo las tendencias de otros experimentos.

En el número 5, la eliminación de nitrógeno baja al 80.9 %. Considerando eliminación por unidad de biomasa es de 21 mg N/ g SSV d, valor inferior al anterior. Si se observa la composición del nitrógeno en el efluente, se aprecia la disminución del amonio, pero también el aumento del nitrato y del nitrito. En las condiciones de este experimento, la nitrificación mejora mucho, pero en detrimento de la desnitrificación. Como balance global, el rendimiento no es tan satisfactorio. Existe otro factor que también afecta negativamente a la valoración de este experimento, los problemas de "rising" en el sedimentador. La pobre desnitrificación, debida al excesivo porcentaje de condiciones aeróbicas en el reactor 1, provoca altos niveles de nitrato en la salida, lo que induce procesos de desnitrificación en el sedimentador secundario, con la consiguiente pérdida de lodos en el efluente. Este problema desaconseja la utilización de esta duración de ciclos, por lo que en experimentos posteriores se utiliza una relación de condiciones anóxicas y aeróbicas 2 a 1.

En el experimento 6 la eliminación de nitrógeno aumenta al 82.6 %, la máxima de esta serie de experimentos. Considerando eliminación por unidad de biomasa es de 23.5 mg N/ g SSV d, valor superior al anterior. Con esta secuencia, la nitrificación mejora, sin que afecte tanto a la desnitrificación, por lo que el balance global es mejor al anterior. En la figura 5.55, puede observarse como el nitrógeno total de salida por unidad de biomasa es casi igual al del número 5, pero la composición varía. En el nº 6 el porcentaje de amonio es mayor, porque la nitrificación no está tan favorecida, pero esto tiene efectos positivos, porque la desnitrificación se ve mejorada. Si se compara al experimento 4, se puede observar que la eliminación por unidad de biomasa tiene un valor muy parecido, y que el nitrógeno total en el efluente es también muy similar. La diferencia entre los dos experimentos es la diferente fracción de cada una de las especies. Puede observarse como en el nº 6 aparecen 0.8 mg/l menos de nitrógeno amoniacal, transformados en nitrato. Los niveles de nitrato en este experimento no produjeron los problemas de "rising" que aparecieron en el nº 5, por lo que las condiciones utilizadas parecen las más favorables de las estudiadas.

Por último, en el experimento 7 la eliminación de nitrógeno es del 79.0 %. Considerando eliminación por unidad de biomasa es de 23.75 mg N / g SSV d. Los niveles de nitrógeno en el efluente son los mayores obtenidos aplicando los ciclos de aireación. La duración de los ciclos aeróbicos, 10 minutos, no proporciona tiempo suficiente para que los microorganismos nitrificantes se adapten a las nuevas condiciones, pero si tiene un efecto inhibitorio de la desnitrificación, por lo que los resultados no se ven excesivamente mejorados respecto al experimento de referencia 3.

En resumen global de esta serie de experimentos, se puede observar como la secuencia más positiva de las probadas es la que utiliza un ciclo aeróbico de 30 minutos seguido de uno anóxico de 60 minutos, correspondiente al experimento 6. Con ella se obtiene una eliminación del 82.6 %, consumiendo 25.6 mg N/ g SSV d, y vertiendo alrededor de 5 mg N / g SSV d con una composición del 37 % de nitrógeno amoniacal.

Al aplicar esta estrategia de operación, se observa como aparecen algunos problemas que deben ser evitados al implementarla como estrategia de control. En primer lugar, debe prevenirse la aparición de cantidades de nitrato excesivas en el

efluente, lo que produce problemas de "rising" en el sedimentador, y en consecuencia, pérdidas de sólidos e incumplimiento de la legislación en este sentido.

Otro problema es la poca rentabilidad económica de mantener los ciclos de aireación las 24 horas. En horario de baja carga, es suficiente la configuración normal para tratar la carga introducida, por lo que se está malgastando energía innecesariamente.

La aireación del primer reactor también provoca una disminución del consumo de oxígeno en los reactores 2 y 3, lo que produce un funcionamiento incorrecto de su control de oxígeno, aunque éste es un problema resoluble con una mejora del sistema de aireación.

Considerando los hechos anteriores y los resultados experimentales, se propone la utilización de esta estrategia de operación, con ciclos de 30 minutos aeróbico y 60 anóxico, cuando aparezcan condiciones de alta carga en el influente.

Variación de pH y redox

Para este conjunto de experimentos se puede estudiar la diferente respuesta del potencial redox y del pH durante los ciclos aeróbicos / anóxicos. En la figura 5.56 se representan los valores que toma el potencial redox para el periodo de 12:00 a 15:00 horas para los 5 experimentos.

En la figura puede observarse la disminución del potencial redox desde que empieza un periodo anóxico. Esta disminución tiene una pendiente inicial aproximadamente igual en los cuatro experimentos cíclicos. La disminución es inicialmente muy rápida, pero la velocidad de variación va disminuyendo progresivamente. Puede observarse que se requiere un tiempo elevado para alcanzar las condiciones correspondientes a un sistema anóxico como en el experimento sin aireación cíclica. La duración del periodo anóxico en los experimentos cíclicos es la variable clave para obtener condiciones anóxicas, con potencial redox bajo, lo que corresponde a un sistema desnitrificante funcionando adecuadamente. En este sentido, los experimentos más adecuados son el nº 4 y el nº 6, correspondientes a secuencias 20'ae-40'an y 30'ae-60'an. En el experimento 5 (30'ae-30'an) se observa que el redox no baja de –75 mV, por lo que no se está eliminado el nitrato acumulado en el proceso de nitrificación. Esto también ocurre en el experimento 7, a pesar que la relación anóxico/aeróbico es 2 a 1, el sistema no tiene tiempo suficiente para eliminar el oxígeno aportado en el periodo aeróbico y consumir el nitrato producido, lo que se refleja finalmente en los resultados de eliminación de nitrógeno.

En la figura 5.57 se representa la variación de redox, pH y oxígeno disuelto durante dos horas para los cuatro experimentos cíclicos.

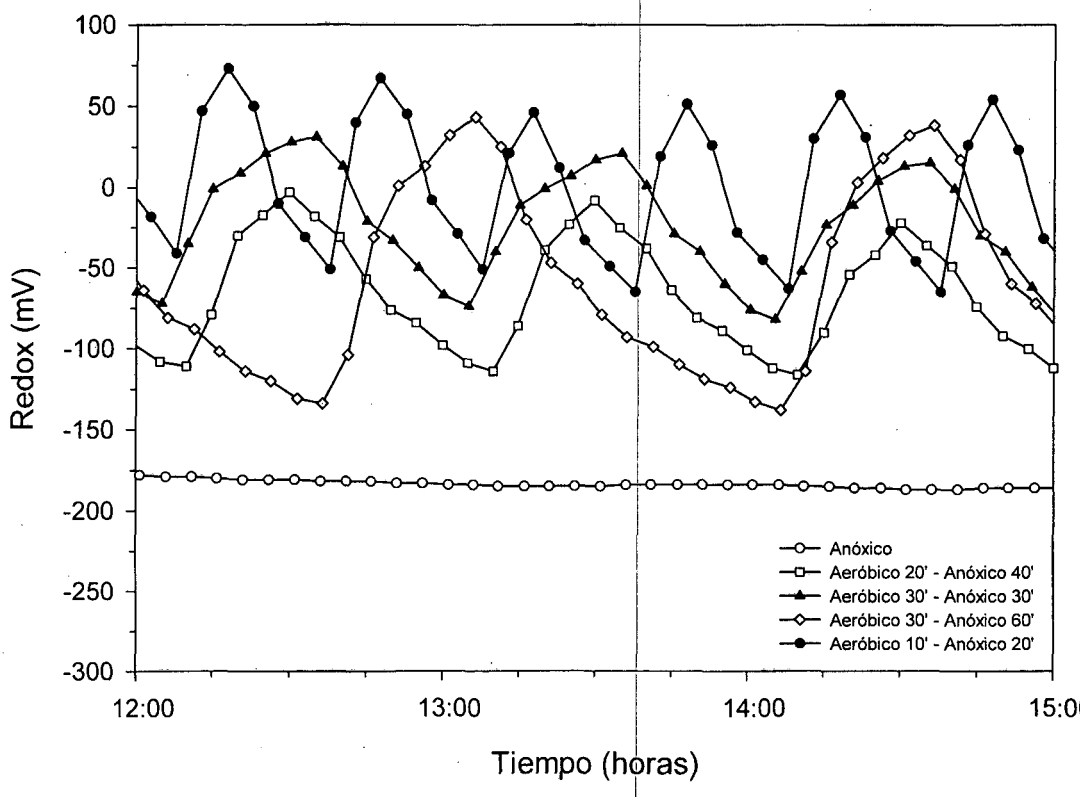


Fig.5.56. Variación del potencial redox para diferentes ciclos de aireación

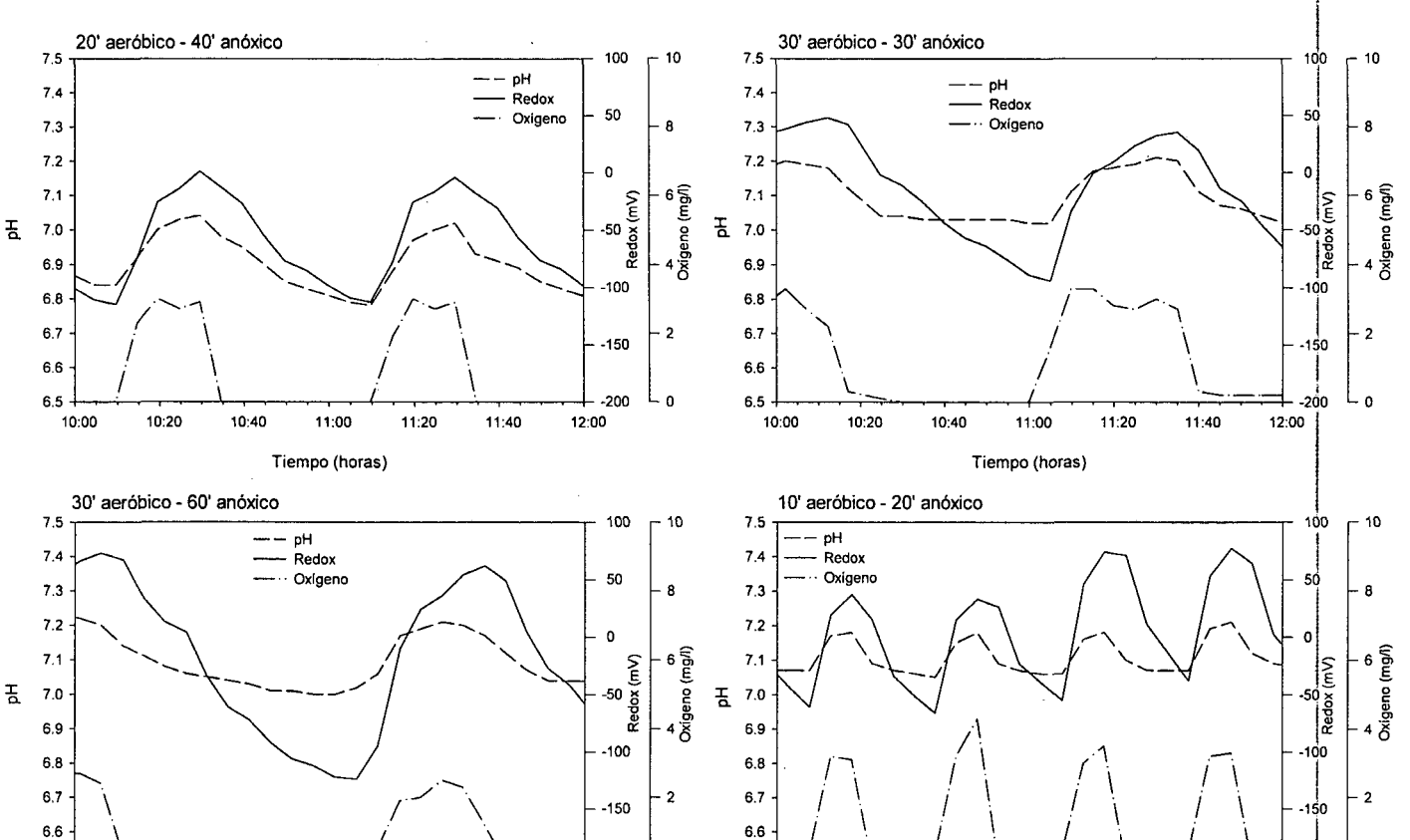


Fig. 5.57. Variación de pH, redox y oxígeno disuelto durante los ciclos de aireación

En la figura 5.57 puede observarse la relación entre las medidas de redox y pH con el oxígeno disuelto presente en el reactor en el diferente periodo del ciclo. El cambio de condiciones anóxicas a aeróbicas siempre induce un aumento del potencial redox y un aumento del pH. El aumento del redox viene provocado por la capacidad oxidante del oxígeno, pero el cambio de pH está provocado por la acumulación y arrastre de CO_2 en el medio. Si el cambio de pH fuera provocado mayoritariamente por la nitrificación y la desnitrificación, los resultados deberían ser los contrarios. La nitrificación en condiciones aeróbicas va acompañada de una disminución del pH y la desnitrificación provoca un incremento del pH. La explicación a estos cambios puede estar en el desplazamiento del CO_2 presente en el medio. En condiciones anóxicas, los microorganismos también pueden consumir la materia orgánica hasta la formación de CO_2. La acumulación de este CO_2 produce la bajada de pH, porque se encuentra en forma de ácido carbónico, H_2CO_3. Cuando se recuperan las condiciones aeróbicas, el aire introducido en el sistema produce el arrastre del CO_2 acumulado, lo que tiene como consecuencia la subida del pH.

Todos los cambios en las medidas de pH están muy influenciados por la alcalinidad disponible. Esta alcalinidad proviene del influente de la planta y del control de pH en el tercer reactor, en el que se utiliza bicarbonato. Si la alcalinidad es suficiente, los cambios en el pH se ven amortiguados. En los casos estudiados, la variación máxima producida es de alrededor de 0.4 unidades de pH. Estos cambios mayores se producen cuando la duración del periodo anóxico es más elevada. También se observa que en las condiciones de los experimentos, los valores de pH tienden hacia un valor estable para periodos anóxicos de larga duración.

La variación observada de pH también tiene influencia sobre los valores de redox, aunque la mayor variación de este parámetro está provocada por la concentración de oxígeno presente en el medio.

5.2.7.3 REPRODUCIBILIDAD DE LA RESPUESTA

En esta sección se estudia la reproducibilidad de la respuesta de la planta piloto frente a una misma entrada. En los experimentos 2, 3, 8 y 9 se han utilizado las mismas estrategias de operación (ver tabla 5.61), por lo que teóricamente la respuesta debería ser muy similar. La diferencia en los experimentos se encuentra en la concentración y estado de los microorganismos en el sistema. Debe considerarse que desde la realización del experimento 2 al 9 transcurre un mes y medio, por lo que la población de microorganismos puede haber variado en cantidad y proporción de nitrificantes.

En las figuras 5.58, 5.59 y 5.60 se representa el porcentaje de eliminación de nitrógeno, el nitrógeno eliminado y el nitrógeno que queda en el efluente por unidad de biomasa en el sistema.

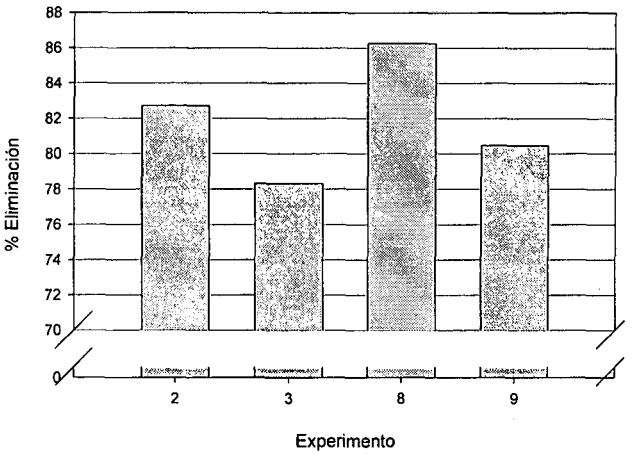


Fig. 5.58. Porcentaje de eliminación de nitrógeno

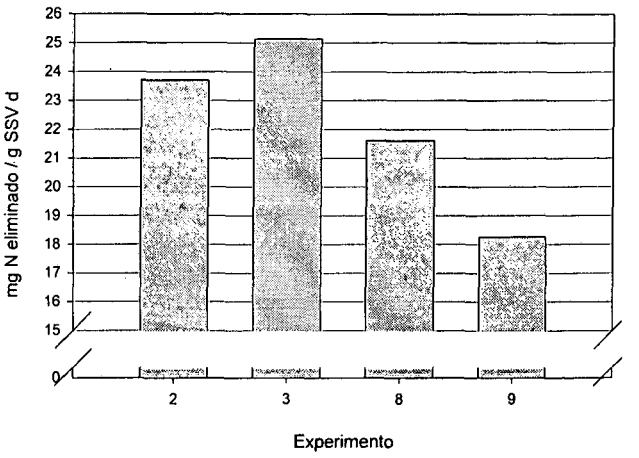


Fig. 5.59. mg N eliminados / g SSV d

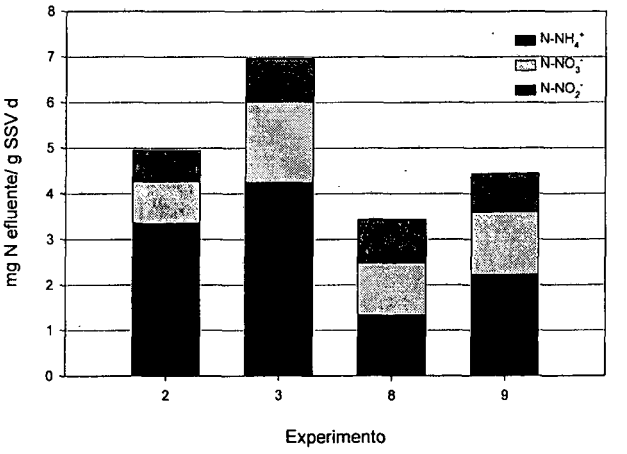


Fig. 5.60. mg N en el efluente / g SSV d

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
2	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2
3	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2
8	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2
9	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
2	454.3	1.532	0.410	0.307	2.249	10.773	82.73	68.12	18.21	13.67	23.71	3.37	0.90	0.68
3	405.7	1.727	0.711	0.386	2.824	10.199	78.32	61.17	25.16	13.67	25.14	4.26	1.75	0.95
8	519.8	0.702	0.588	0.499	1.789	11.233	86.26	39.22	32.89	27.89	21.61	1.35	1.13	0.96
9	574.0	1.283	0.779	0.480	2.542	10.480	80.48	50.49	30.63	18.88	18.26	2.24	1.36	0.84

Tabla 5.61. Condiciones experimentales y resultados de los experimentos 2, 3, 8 y 9

Puede observarse la variabilidad de la respuesta de la planta piloto, independientemente de las unidades utilizadas en la comparación.

Los resultados del experimento 8 son los que más se desvían. En este caso, se produjo un problema con el control de pH del reactor 3. Se dosificó un exceso de bicarbonato por el funcionamiento incorrecto de la sonda de pH del tercer reactor, lo que provocó unos valores de pH bastante superiores a los habituales en el resto de reactores. Este hecho puede haber favorecido a la nitrificación, lo que explicaría el porcentaje de eliminación del 86, bastante mayor al resto de experimentos. En los resultados por unidad de biomasa, también se observa éste es el mejor experimento.

En la comparación del resto de experimentos, se debe tener en cuenta que aparece una importante variación en la concentración inicial de microorganismos en los reactores para cada experimento, que produce variaciones de 405 a 574 gramos totales. Esta importante variación muestra los diferentes estados en los que se encuentra el sedimentador, un parámetro muy importante en el comportamiento de la planta piloto. Es difícil valorar la concentración de sólidos en el sedimentador sin perturbar el sistema, por lo que no se contabiliza su contribución en el cómputo de los sólidos volátiles en el sistema. Esto induce un error en los resultados experimentales, por lo que la comparación de experimentos, aún con las mismas condiciones, pero separados en el tiempo, proporciona resultados no muy repetitivos.

Otro factor que afecta a la eliminación es la temperatura en los reactores. En los experimentos 2 y 8 se mantiene alrededor de 22 °C, mientras que en el 3 y 9 está en torno a 20 °C. Puede observarse que el porcentaje de eliminación se relaciona con los valores de temperatura: los experimentos 2 y 8 son los que muestran una mayor eliminación.

Para evitar la influencia de estos problemas se realiza como primer experimento de cada serie uno de referencia con el que comparar correctamente los resultados del resto de experimentos de la serie.

5.2.7.4 EFECTO DE LA CONSIGNA DE OXÍGENO EN LOS REACTORES 2 Y 3

En los experimentos 9, 10 y 11 se estudia la influencia de la consigna de oxígeno disuelto en los reactores aeróbicos en la nitrificación y, en consecuencia, en la eliminación de nitrógeno. En el experimento 9 se utilizan 3 mg/l, en el nº 10, 5 mg/l y en el nº 11, 1 mg/l. El resto de condiciones experimentales se mantiene constante (tabla 5.62).

Para estos experimentos se representa en las figuras 5.61, 5.62 y 5.63 el porcentaje de eliminación de nitrógeno, el nitrógeno eliminado y el nitrógeno del efluente por unidad de biomasa en el sistema.

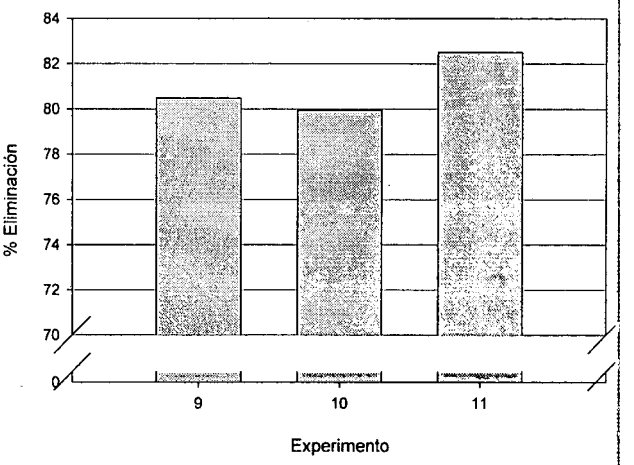


Fig. 5.61. Porcentaje de eliminación de nitrógeno

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
9	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2
10	Anóxico	Aeróbico 5	Aeróbico 5	0.5	2
11	Anóxico	Aeróbico 1	Aeróbico 1	0.5	2

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
9	574.0	1.283	0.779	0.480	2.542	10.480	80.48	50.49	30.63	18.88	18.26	2.24	1.36	0.84
10	519.8	1.226	0.906	0.478	2.609	10.413	79.96	46.97	34.71	18.32	20.03	2.36	1.74	0.92
11	499.4	1.598	0.394	0.284	2.277	10.745	82.51	70.20	17.31	12.49	21.52	3.20	0.79	0.57

Tabla 5.62. Condiciones experimentales y resultados de los experimentos 9 a 11

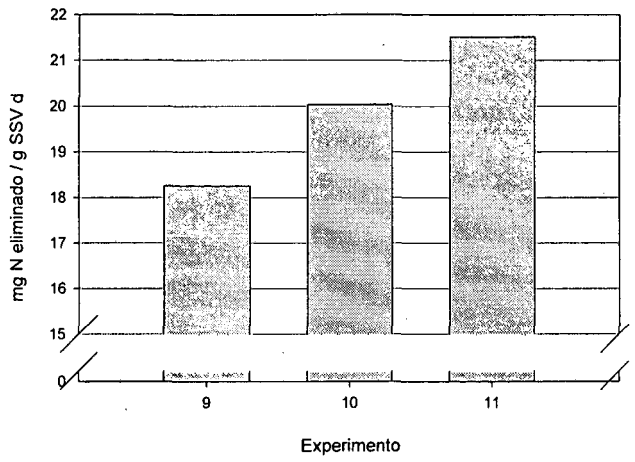


Fig. 5.62. mg N eliminados / g SSV d

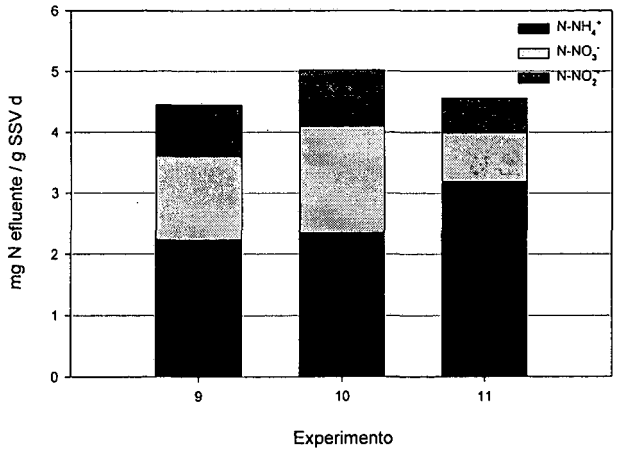


Fig. 5.63. mg N en el efluente / g SSV d

En la figura 5.61 se observa un porcentaje de eliminación prácticamente igual para consignas de 3 y 5 mg/l de oxígeno en los reactores aeróbicos. En el experimento con consigna de 1 mg/l, la eliminación es inesperadamente superior. Estos resultados del experimento 11, son debidos a que la nitrificación no se ve muy favorecida, pero si que se mejora mucho la desnitrificación. Estos resultados se observan en la figura 5.63, donde aparece una importante contribución del amonio al nitrógeno que no se elimina, unido al menor valor de nitrato y nitrito. En los perfiles de salida de nitrato y nitrito en el experimento 11 (figura 5.31) se aprecian concentraciones muy reducidas de salida, lo que favorece el porcentaje de eliminación. A pesar de los valores obtenidos, la nitrificación no es destacable, ya que se obtienen valores máximos de salida de amonio de 8.5 mg/l.

En la figura 5.63 se observa que los niveles de nitrógeno en la salida son mejores utilizando la concentración de oxígeno disuelto de 3 mg/l. En comparación con el experimento de 5 mg/l, la cantidad de amonio en el efluente es prácticamente la misma, pero la de nitrato y nitrito es menor para 3 mg/l, lo que demuestra los problemas de desnitrificación para el experimento de 5 mg/l.

Estos problemas están provocados por entrada de oxígeno en el reactor 1 mediante la recirculación interna. Esto se detecta especialmente para concentraciones de 5 mg/l de oxígeno en el reactor 3. Este hecho debe tenerse en cuenta al introducir esquemas de operación que aumenten la recirculación interna teniendo concentraciones de oxígeno elevadas en el reactor 3.

En cuanto a la eliminación por unidad de biomasa (figura 5.62), se observa un aumento en cada uno de los experimentos. Esto puede ser debido a los problemas de acumulación de biomasa en el sedimentador, ya que la biomasa medida en los reactores disminuye un 15 % del experimento 9 al 11.

Como resultado global de estos experimentos, se observa que la nitrificación mejora para un aumento de la consigna de oxígeno en los reactores aeróbicos, aunque a partir de 3 mg/l no se obtienen mejoras representativas. Por otro lado, la desnitrificación se ve favorecida en condiciones de baja concentración de oxígeno en los reactores aeróbicos. Con una consigna de 1 mg/l, se obtienen resultados especialmente favorables, lo que puede estar provocado por la menor recirculación de oxígeno desde el reactor 3 al reactor 1, así como por la posible nitrificación y desnitrificación simultánea en los reactores aeróbicos.

5.2.7.5 EFECTO DE LA RECIRCULACIÓN INTERNA

En los experimentos 12, 13 y 14 se estudia el efecto de la recirculación interna en la desnitrificación y, en consecuencia, en la eliminación global de nitrógeno. El experimento 12 se utiliza como referencia y en los experimentos 13 y 14 se introduce la estrategia de control que varia la relación de recirculación interna en función de la concentración de nitrato más nitrito a la salida. Las condiciones de cada experimento se muestran en la tabla 5.63.

En las figuras 5.64, 5.65 y 5.66 se representa el porcentaje de eliminación, la eliminación y el nitrógeno en el efluente por unidad de biomasa en los reactores.

En los tres experimentos se observa una alta variabilidad en los sólidos totales del sistema, ya que aparecen valores de 530 a 655 gramos de biomasa total en los reactores. Esto provoca resultados poco reproducibles en la planta piloto. A pesar de la variabilidad, puede observarse como el porcentaje de eliminación aumenta del experimento 12 al 13.

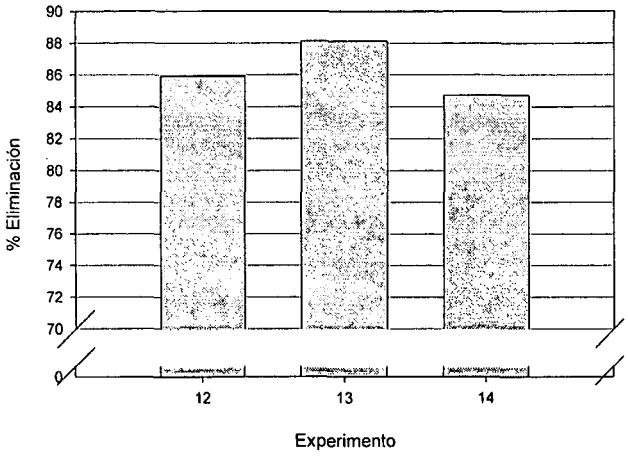


Fig. 5.64. Porcentaje de eliminación de nitrógeno

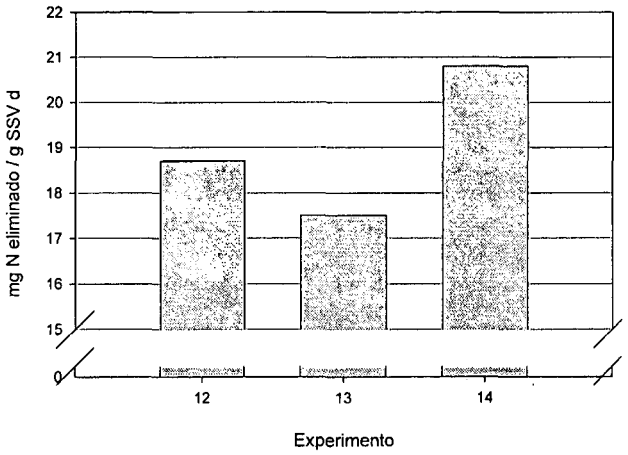


Fig. 5.65. mg N eliminados / g SSV d

Este aumento de eliminación no se observa si se calcula la eliminación por unidad de biomasa (figura 5.65), pero el nitrógeno que aparece en el efluente es menor (figura 5.66), tanto en forma de amonio como nitrato y nitrito.

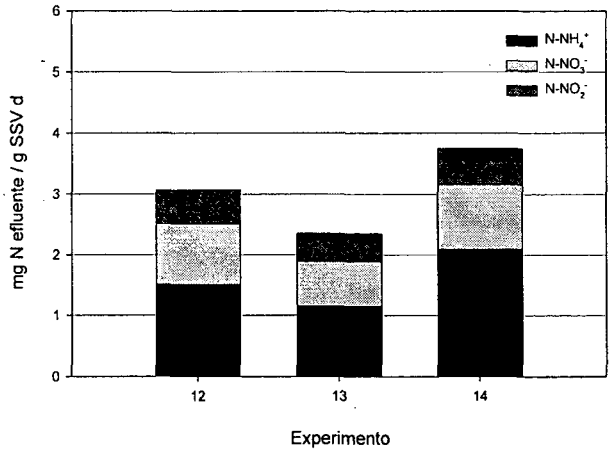


Fig. 5.66. mg N en el efluente / g SSV d

Para el experimento 14, repetición de las condiciones del anterior, no se obtienen los mismos resultados. La eliminación de nitrógeno es elevada (84.7 %), aunque inferior a las anteriores. Por unidad de biomasa, la eliminación es de 20.8 mgN/g SSV d, bastante superior a las anteriores. La baja concentración de sólidos en los reactores provoca ese valor de eliminación por unidad de biomasa.

Los resultados de eliminación obtenidos muestran la variabilidad de la respuesta de la planta piloto, debido al comportamiento no ideal del sedimentador.

En los perfiles de nitrato y nitrito del efluente (figuras 5.33 y 5.34) si se observa la correcta actuación del control de la recirculación interna en función de la concentración de $N\text{-}NO_x$ en el efluente. Puede observarse como cuando la concentración supera los 2 mg/l, la relación de recirculación interna varía, con lo que se consigue mantener la concentración de salida de $N\text{-}NO_x$ en el efluente, al contrario que en el experimento 12, donde la concentración de nitrato y nitrito continúa

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
12	Anóxico	Aeróbico 3	Aeróbico 3	0.5	2
13	Anóxico	Aeróbico 3	Aeróbico 3 ó 1	0.5	1 ó 2 ó 4
14	Anóxico	Aeróbico 3	Aeróbico 3 ó 1	0.5	1 ó 2 ó 4

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
12	598.2	0.907	0.594	0.334	1.835	11.187	85.91	49.42	32.35	18.23	18.70	1.52	0.99	0.56
13	655.7	0.762	0.479	0.303	1.544	11.478	88.14	49.33	31.05	19.61	17.51	1.16	0.73	0.46
14	530.3	1.115	0.564	0.312	1.991	11.031	84.71	56.00	28.33	15.67	20.80	2.10	1.06	0.59

Tabla 5.63. Condiciones experimentales y resultados de los experimentos 12 a 14

aumentando. También se observa como durante algunos periodos de los experimentos 13 y 14, la relación de recirculación interna toma un valor de 1, ya que la concentración de $N\text{-}NO_x$ en el efluente disminuye por debajo del valor límite. Esta relación de recirculación permite un ahorro energético en periodos donde no es necesaria tanta recirculación interna.

5.2.7.6 EFECTO DE LA RELACIÓN DE RECIRCULACIÓN EXTERNA VARIABLE

En los experimentos 15, 16 y 17 se estudia el efecto de la relación de recirculación externa variable en la eliminación de nitrógeno en la planta piloto. En los experimentos 16 y 17 se incluye un aumento de la recirculación externa en determinados periodos para hacer frente al aumento de carga en la planta.

Este incremento se determina en función de los valores de OUR en el reactor 2 y de la salida de nitrógeno amoniacal. En el nº 17 también se modifica la consigna de oxígeno disuelto en el reactor 3 cuando la relación de recirculación interna es 4. De este modo se evita el aporte excesivo de oxígeno al reactor anóxico, ya que este es un factor inhibitorio de la desnitrificación.

En las figuras 5.67, 5.68 y 5.69 se representan los valores de eliminación para los tres experimentos.

Se muestran los resultados del experimento 16, aunque debe considerarse que no son comparables al resto de experimentos, ya que han sido calculados considerando la misma entrada que el resto. Durante dos horas la planta estuvo parada debido a un problema de operación, por lo que no ha habido entrada a la planta durante ese periodo. Los resultados de este experimento demuestran que la eliminación obtenible en la planta puede ser casi total dependiendo de las condiciones experimentales. Una parada de dos horas permite el consumo casi total del amonio (ver figura 5.36), pero no así de nitrato ni nitrito. Esto puede deberse a la falta de DQO para que los microorganismos heterótrofos puedan desnitrificar, pero globalmente los resultados son de eliminación casi total.

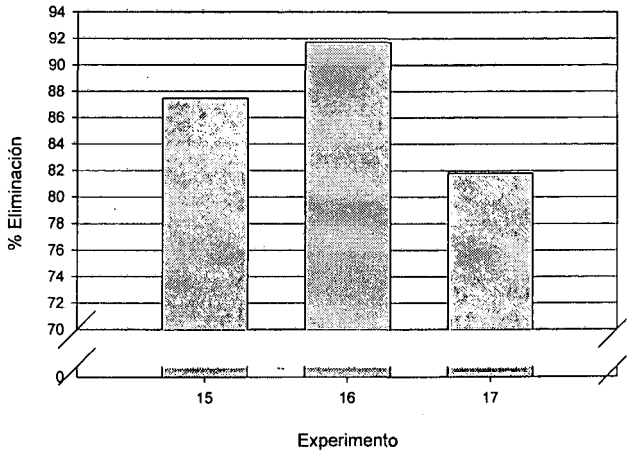


Fig. 5.67. Porcentaje de eliminación de nitrógeno

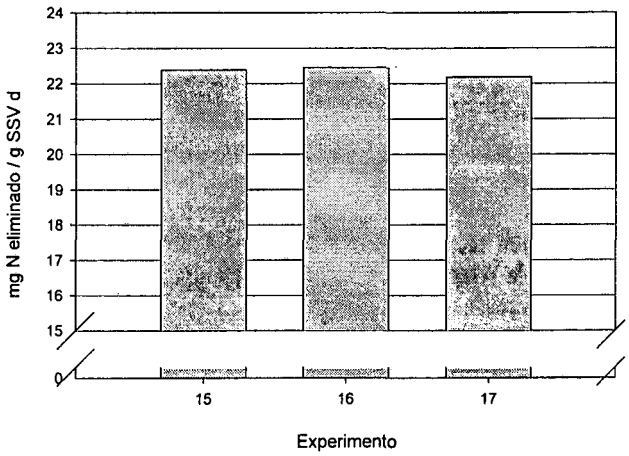


Fig. 5.68. mg N eliminados / g SSV d

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
15	Anóxico	Aeróbico 3	Aeróbico 3 ó 1	0.5	1 ó 2 ó 4
16	Anóxico	Aeróbico 3	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4
17	Anóxico	Aeróbico 3	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
15	508.9	0.652	0.681	0.298	1.632	11.391	87.47	39.97	41.76	18.27	22.38	1.28	1.34	0.59
16	532.2	0.164	0.697	0.217	1.078	11.944	91.72	15.18	64.65	20.17	22.45	0.31	1.31	0.41
17	480.4	0.902	1.065	0.401	2.368	10.654	81.81	38.09	44.96	16.94	22.18	1.88	2.22	0.84

Tabla 5.64. Condiciones experimentales y resultados de los experimentos 15 a 17

esas condiciones la eliminación ha mejorado. También se obtienen rendimientos de eliminación por unidad de biomasa bastante mejores en los experimentos 19 y 20.

En cuanto al nitrógeno en el efluente, se observa una cantidad total igual, pero con diferente composición. En los experimentos 19 y 20 la proporción de nitrato y nitrito es superior, lo que indica que la nitrificación está más favorecida, así como que la desnitrificación no se ve excesivamente perjudicada como consecuencia de los ciclos de aireación en el reactor 1.

En los perfiles de salida (figuras 5.38, 5.39 y 5.40) se puede observar la menor concentración de amonio en el efluente en los experimentos 19 y 20, pero unido a un ligero aumento en las concentraciones de nitrato y nitrito que se producen oscilatoriamente, coincidiendo con los ciclos de aireación en los que la desnitrificación no se produce.

Como valoración global, se observa que la eliminación mejora, lo que unido a la optimización de los costes de depuración, puede considerarse un resultado muy positivo.

5.2.7.8 COMPORTAMIENTO DEL SISTEMA ANTE DIFERENTES ENTRADAS

En los experimentos 21, 22 y 23 se utilizan todas las estrategias de control implementadas. En el nº 21 se repite la entrada, pero en el nº 22 y el nº 23 se utiliza otro tipo de entrada, constante. En el nº 22 se introducen dos caudales diferentes durante dos periodos, pero la cantidad de nitrógeno adicionado mediante las bombas dosificadoras se mantiene constante. De este modo se producen dos periodos de diferente caudal pero con igual carga. Los caudales y la dosificación se han calculado de tal modo que el caudal total y el nitrógeno total introducido durante el experimento es igual al de los experimentos anteriores. En el nº 23 se simula una condición de baja carga de nitrógeno y de DQO. Se introducen dos caudales diferentes durante dos periodos, pero la cantidad de nitrógeno adicionado mediante las bombas dosificadoras se mantiene constante. De este modo se producen dos periodos de diferente caudal pero con igual carga.

Los resultados de eliminación se muestran en las figuras 5.74, 5.75 y 5.76.

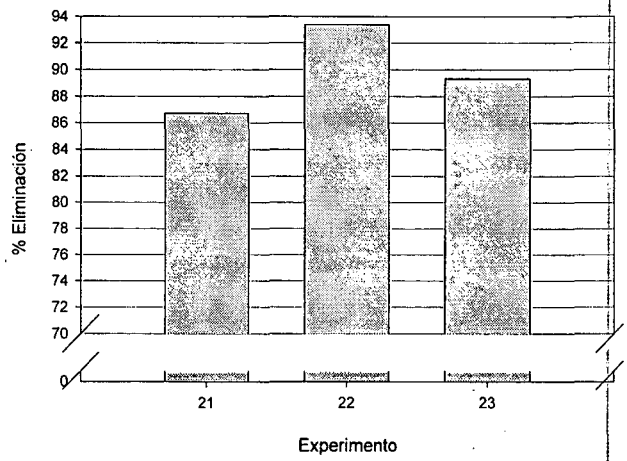


Fig. 5.74. Porcentaje de eliminación de nitrógeno

En el experimento 21 se repite la buena eliminación de los experimentos 19 y 20, considerando la concentración de biomasa. La eliminación de nitrógeno es de 86.7 % y por unidad de biomasa es de 26.01 mg N /g SSV d, rendimiento de eliminación elevado. Las concentraciones de nitrógeno amoniacal a la salida son ligeramente altas, pero las de nitrato y nitrito son reducidas, por lo que globalmente la eliminación es correcta.

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
21	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4
22	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4
23	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
21	434.2	1.188	0.255	0.285	1.728	11.294	86.73	68.74	14.75	16.51	26.01	2.74	0.59	0.66
22	418.3	0.270	0.351	0.239	0.861	12.161	93.39	31.39	40.80	27.81	29.07	0.65	0.84	0.57
23	427.0	0.209	0.397	0.144	0.751	6.289	89.33	27.88	52.91	19.21	14.73	0.49	0.93	0.34

Tabla 5.66. Condiciones experimentales y resultados de los experimentos 21 a 23

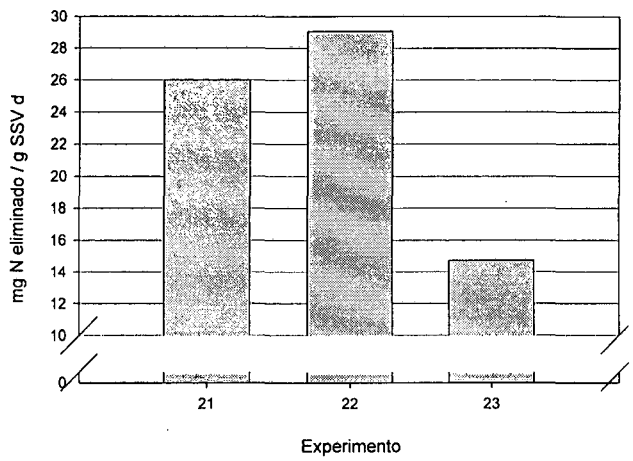


Fig. 5.75. mg N eliminados / g SSV d

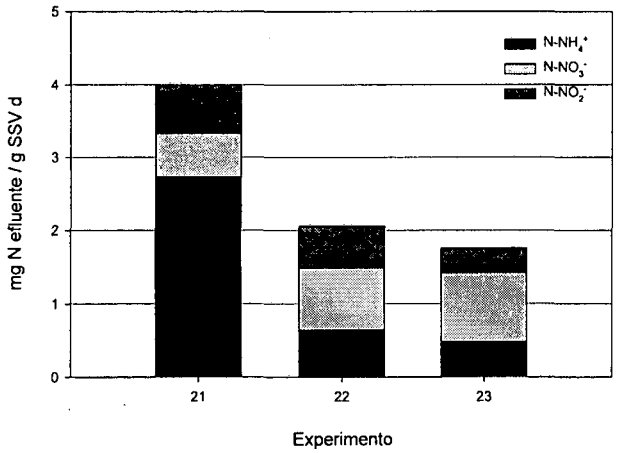


Fig. 5.76. mg N en el efluente / g SSV d

En el experimento 22 se obtienen los mejores valores de eliminación de nitrógeno de todos los experimentos. El porcentaje de eliminación es del 93.4 %, un valor muy elevado. La eliminación por unidad de biomasa es de 29.07 mg N /g SSV d, el mayor rendimiento de eliminación obtenido. Las concentraciones de nitrógeno a la salida también son muy reducidas (ver figura 5.42), por lo que la carga vertida por el efluente es también mínima (figura 5.76). Teniendo en cuenta que la carga total es igual a la de anteriores experimentos, los resultados demuestran la importancia de la adecuada distribución durante todo el periodo para obtener eliminaciones casi totales. Estos resultados hacen pensar que la disponibilidad de un depósito de homogeneización sería una mejora muy importante para el proceso. Esta es una opción posible para caudales reducidos, pero no es una opción demasiado económica para una planta de tratamiento de aguas residuales urbanas de tamaño medio.

También debe considerarse que la eliminación podría haber sido aún superior, manteniendo durante las 24 horas condiciones aeróbicas en los reactores 2 y 3, y una relación de recirculación máxima. Estas condiciones hubieran proporcionado una eliminación superior, pero el coste económico hubiera sido mucho mayor.

Otra consideración que debe realizarse es que las estrategias de control desarrolladas para un tipo de entrada como la de los experimentos anteriores puede no ser adecuada para otro tipo de condiciones de entrada. Por ejemplo, en el experimento 22 se pasa de mantener condiciones anóxicas en el reactor 2 a una hora después tener que establecer condiciones aeróbicas para ese reactor y realizar secuencias de aireación en el reactor 1. En este caso, se deberían haber mantenido condiciones aeróbicas en el reactor 2, con lo que no se habría producido el aumento de la concentración de amonio en el efluente. Estos problemas también son consecuencia de las medidas disponibles de amonio, nitrato y nitrito son únicamente en el efluente. Si se hubiera dispuesto de la medida de la entrada podría haberse implementado alguna estrategia de control complementaria.

En el último experimentos la eliminación de nitrógeno es del 89.3 %. Se podría obtener una eliminación superior, pero los cambios de modo de operación están regulados por los mismos niveles en el efluente que en anteriores experimentos. Cambiando esos valores en el sistema experto, el porcentaje de eliminación podría ser muy mejorado. La eliminación por unidad de biomasa es de 14.73 mg N /g SSV d, muy reducida porque la carga es muy pequeña. Las concentraciones de nitrógeno a la salida también son mínimas (ver figura 5.43). En este experimento se demuestra prácticamente que es posible obtener buenos resultados de eliminación utilizando los recursos necesarios, sin desperdiciar energía de aireación o recirculación cuando la eliminación deseada no lo requiere.

5.2.7.9 EFECTO DEL SEDIMENTADOR

En los experimentos 24 y 25 se prueba la respuesta del sistema con el nuevo sedimentador de 25 litros de volumen, en contraste con los 60 litros del anterior. La entrada es el mismo perfil de alimentación utilizado en casi todos los experimentos, mientras que las condiciones utilizadas son las de máximo control (tabla 5.67). La respuesta se modifica notablemente, como consecuencia del diferente comportamiento de sedimentador.

En las figuras 5.77, 5.78 y 5.79 se representa el porcentaje de eliminación, la eliminación y el nitrógeno en el efluente por unidad de biomasa en los reactores.

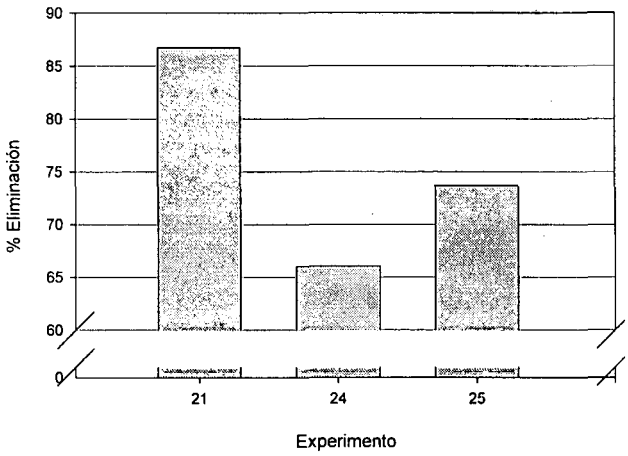


Fig. 5.74. Porcentaje de eliminación de nitrógeno

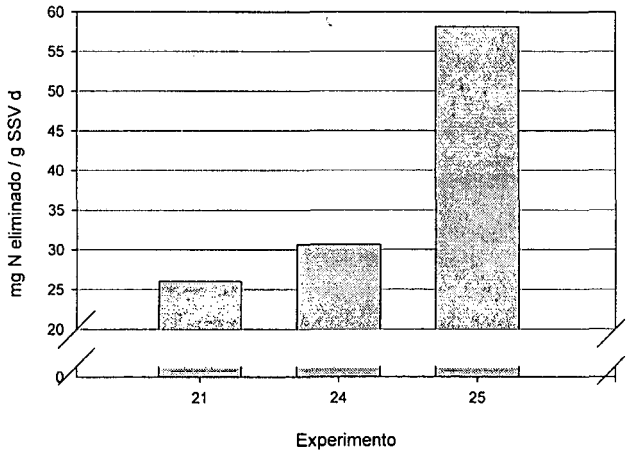


Fig. 5.75. mg N eliminados / g SSV d

En el experimento 24 la eliminación de nitrógeno es de 66.0 %, valor muy reducido. La eliminación por unidad de biomasa es de 30.65 mg N /g SSV d, muy elevada porque la concentración de microorganismos es escasa. También debe considerarse la alta concentración de amonio a la salida, que supera los límites legales (figura 5.64). En conclusión, puede admitirse que la nueva situación de la planta no favorece al porcentaje de eliminación de nitrógeno, aunque proporcionalmente los resultados sean buenos.

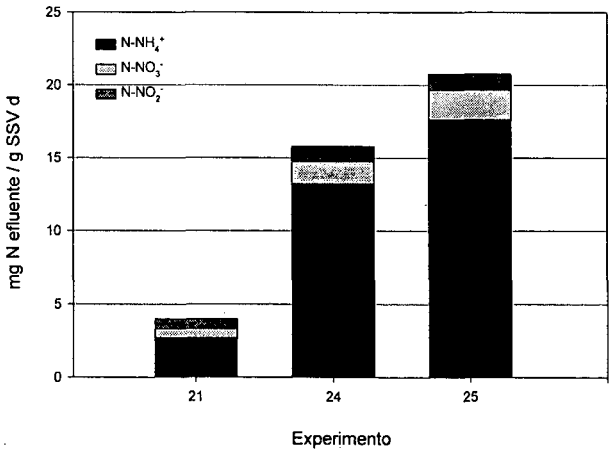


Fig. 5.76. mg N en el efluente / g SSV d

En el experimento 25 la eliminación de nitrógeno es de 73.7 %, valor reducido pero mejor al anterior. La eliminación por unidad de biomasa es de 58.12 mg N /g SSV d, demasiado elevado comparado con el resto de experimentos. La concentración de microorganismos puede haber variado a lo largo del experimento, debido al mal funcionamiento del sedimentador. Una posible explicación es la acumulación de biomasa en el sedimentador para caudales bajos de recirculación externa (correspondientes al inicio del experimento) y su posterior recirculación al aumentar la recirculación externa cuando se detecta el aumento de carga en el sistema. También es posible que el nitrógeno orgánico, que normalmente desaparece por completo, no se haya hidrolizado totalmente, lo que no estaríamos considerando cuando calculamos el rendimiento.

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
21	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4
24	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4
25	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
21	434.2	1.188	0.255	0.285	1.728	11.294	86.73	68.74	14.75	16.51	26.01	2.74	0.59	0.66
24	280.5	3.713	0.420	0.290	4.424	8.598	66.03	83.94	9.50	6.56	30.65	13.24	1.50	1.03
25	165.1	2.915	0.331	0.182	3.428	9.594	73.68	85.04	9.66	5.30	58.12	17.66	2.01	1.10

Tabla 5.67. Condiciones experimentales y resultados de los experimentos 21, 24 y 25

5.2.7.10 MEJORA GLOBAL OBTENIDA MEDIANTE EL SISTEMA DE CONTROL

En esta sección se compara el efecto global que tiene la implementación de todas las estrategias de control estudiadas. Para ello se comparan los resultados de los experimentos 1 y 20 (tabla 5.68), ambos con el mismo influente pero con diferente grado de control. En el primero se utiliza el mínimo número de estrategias probadas y en el experimento 20 se utilizan todas las estrategias estudiadas.

En la figura 5.77 se observa el porcentaje de eliminación global de nitrógeno. Se obtiene una mejora en la eliminación del 10.5 %.

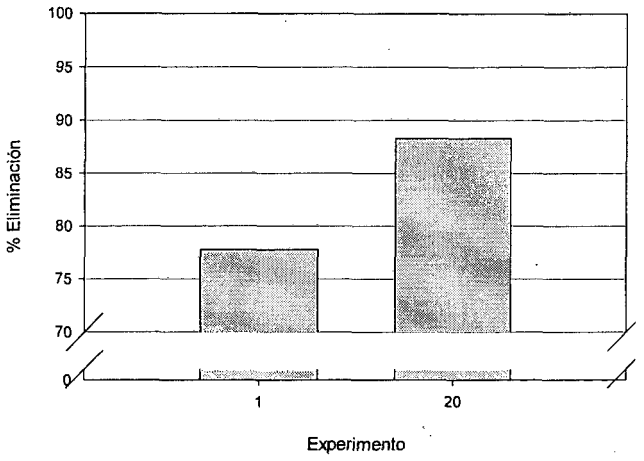


Fig. 5.77. Porcentaje de eliminación de nitrógeno

En la figura 5.78 se representa la eliminación de nitrógeno por unidad de biomasa y día, que en este caso supone una mejora de 3.7 mg N / g SSV d, lo que representa un porcentaje del 16.3 %.

Por último se representa el nitrógeno, en las diversas formas en que aparece en el efluente (figura 5.79). En este caso, se reduce un 46 % la salida de nitrógeno.

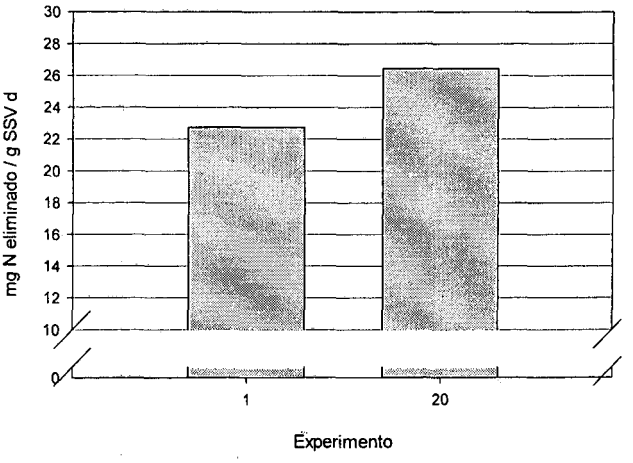


Fig. 5.78. mg N eliminados / g SSV d

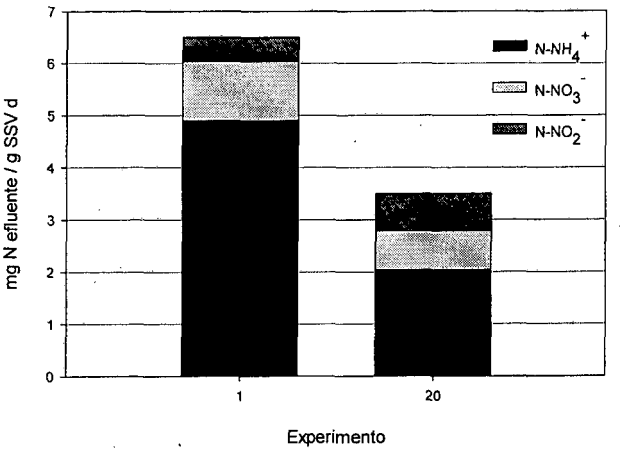


Fig. 5.79. mg N en el efluente / g SSV d

Experimento	Estado óxico (consigna mg O_2 / l)			Relación de recirculación	
	Reactor 1	Reactor 2	Reactor 3	Externa	Interna
1	Anóxico	Aeróbico 3	Aeróbico 3	Q=150 (ml/min)	2
20	Anox ó 30'aer-60'anox	Aer 3 ó Anox 0	Aeróbico 3 ó 1	0.5 ó 1	1 ó 2 ó 4

Experimento	SSV_T (g)	$N\text{-}NH_4^+{}_S$ (g)	$N\text{-}NO_3^-{}_S$ (g)	$N\text{-}NO_2^-{}_S$ (g)	$N_{TOTAL\ S}$ (g)	N eliminado (g)	% Eliminación	% $N\text{-}NH_4^+{}_S$	% $N\text{-}NO_3^-{}_S$	% $N\text{-}NO_2^-{}_S$	mg N / g SSV d	mg $N\text{-}NH_4^+{}_S$ / g SSV d	mg $N\text{-}NO_3^-{}_S$ / g SSV d	mg $N\text{-}NO_2^-{}_S$ / g SSV d
1	445.2	2.186	0.503	0.204	2.893	10.129	77.78	75.57	17.39	7.04	22.75	4.91	1.13	0.46
20	434.6	0.893	0.321	0.310	1.524	11.499	88.30	58.61	21.07	20.32	26.46	2.05	0.74	0.71

Tabla 5.68. Condiciones experimentales y resultados de los experimentos 1 y 20

Como puede observarse en los resultados de eliminación, las estrategias de control tienen un efecto global muy positivo. Aparte de esta lectura superficial, debe considerarse que la mejora obtenida se ha realizado con el mínimo coste energético posible, optimizando el funcionamiento de la planta mediante las reglas de operación implementadas en tiempo real. Además de esta mejora global, si se observan los perfiles de las diferentes formas de nitrógeno a la salida de la planta (figuras 5.21 y 5.40), se observa la menor concentración de nitrógeno amoniacal en el experimento 20 (menor a 6 mg/l) respecto al experimento 1 (superior a 12 mg/l), lo que puede ser un factor de vital importancia para cumplir los límites legislados (apartado 1.1).

5.2.7.11 ELIMINACIÓN DE FÓSFORO

Respecto a la eliminación de fósforo por la planta piloto, ésta ha sido bastante constante, se ha visto poco influenciada por las distintas estrategias de control utilizadas.

En general, los valores de $P\text{-}PO_4^{3-}$ de la entrada en forma de perfil (con una concentración variable entre 3 y 8 mg/l), han sido eliminados, obteniéndose resultados inferiores a 2 mg/l.

El único proceso que puede afectar a la eliminación de fósforo, con la configuración de operación de la planta piloto, sería la entrada de nitrato en el reactor anaeróbico. Este problema de operación sólo puede ocurrir cuando no se está desnitrificando correctamente, como en el problema de falta de DQO del experimento 17, pero sólo afectaría si el problema es de larga duración.

En los experimentos realizados, sólo se producen concentraciones de fosfato elevadas cuando existen problemas con los lodos del sedimentador. Los lodos pueden retener fosfato adherido a los flóculo, que en caso de ser removidos se desprende, produciendo valores elevados de fosfato en el efluente de la planta, tal como se puede observar en algunos experimentos.

6 IMPLEMENTACIÓN Y VALIDACIÓN EN PLANTA REAL. EDAR GRANOLLERS

6 IMPLEMENTACIÓN Y VALIDACIÓN EN PLANTA REAL. EDAR GRANOLLERS

Además de la implementación del SE en la planta piloto del "Departament d 'Enginyeria Química de la Universitat Autònoma de Barcelona" se está implementando otro prototipo, en colaboración con el "Laboratori d'Enginyeria Química i Ambiental de la Universitat de Girona". En este prototipo se utilizan las mismas técnicas descritas en el desarrollo del SE de la planta piloto, pero el desarrollo se realiza utilizado la versión 5.0 de G2 [Gensym 1997].

Este prototipo se ha implementado en una planta real, la EDAR de Granollers, que vierte en el cauce del río Besós. Esta planta dispone de pretratamiento, tratamiento primario y tratamiento secundario, utilizando un proceso de lodos activos para eliminar materia orgánica y sólidos en suspensión en el agua residual producida por cerca de 130.000 habitantes equivalentes. Actualmente se está empezando a implementar la eliminación de nitrógeno. El esquema del tratamiento de aguas residuales se muestra en la figura 6.1.

En la planta se realiza una exhaustiva caracterización del influente, proveniente de vertidos urbanos e industriales, incluyendo datos cualitativos y cuantitativos. Los datos cuantitativos son proporcionados por sensores "on-line"

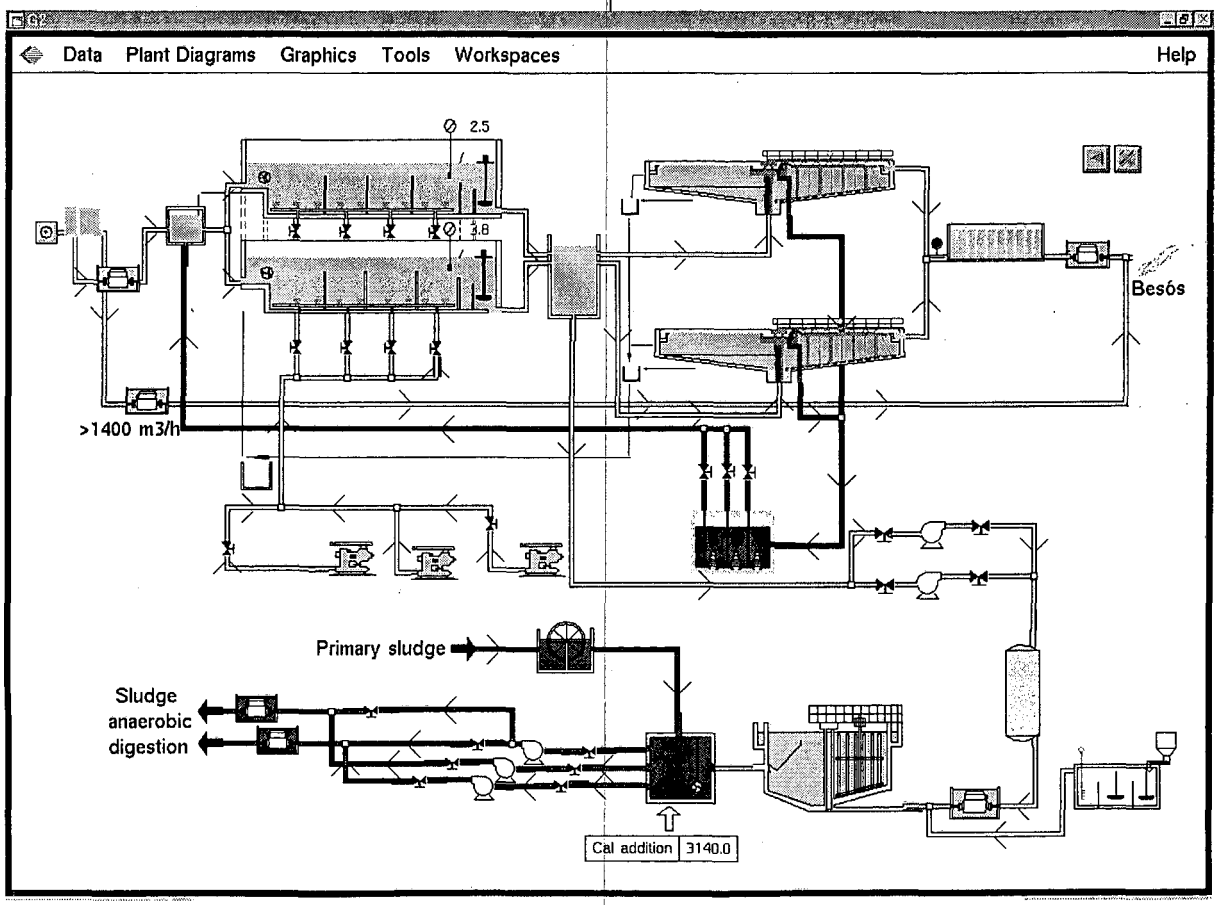


Fig. 6.1. Esquema de tratamiento de la EDAR de Granollers

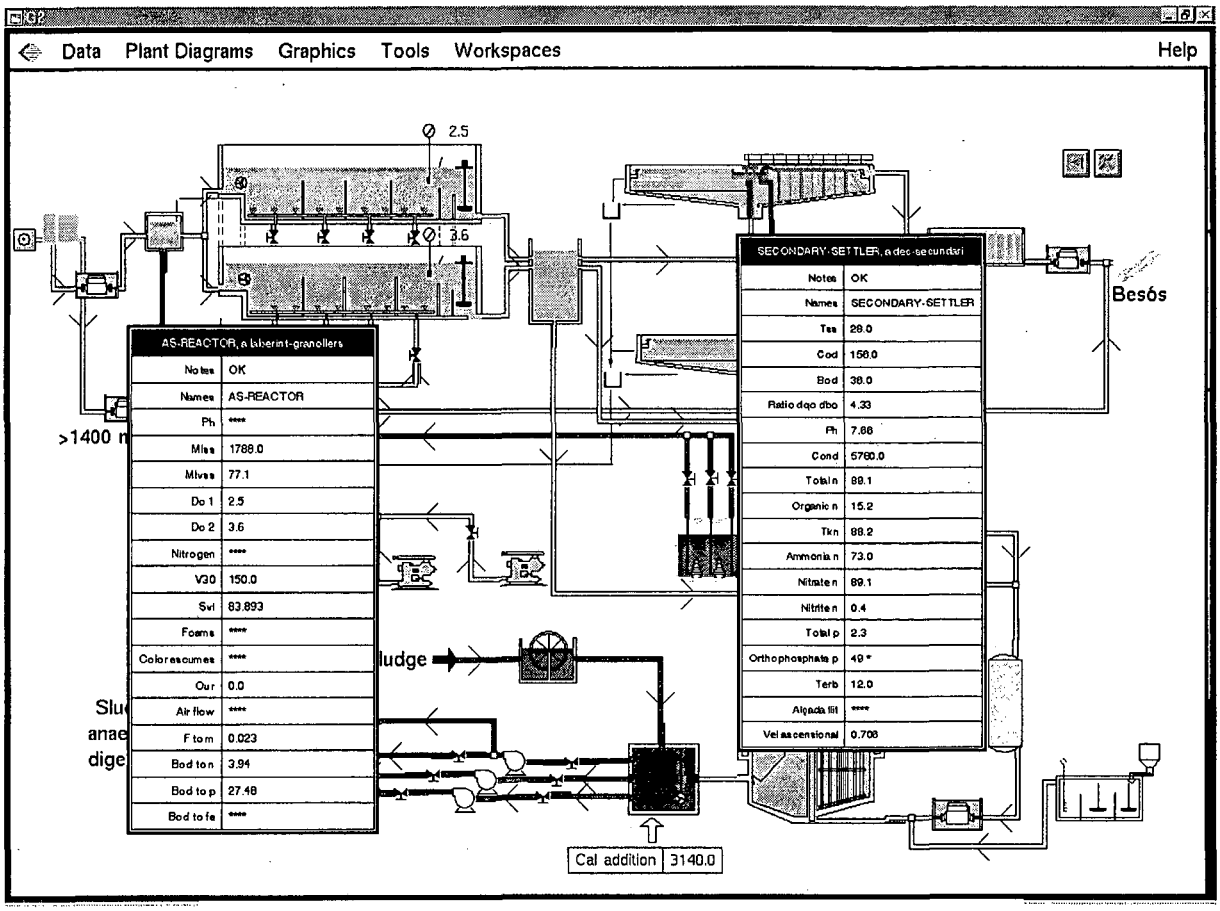


Fig. 6.2. Variables analíticas del proceso

(principalmente caudales, pH, conductividad y oxígeno disuelto) y por determinaciones analíticas de muestras tomadas diariamente en la planta (DQO, nutrientes, aceites y biomasa). También se calculan parámetros globales como el tiempo de residencia celular, SVI o F/M. Los datos cualitativos, medidos con una frecuencia inferior, incluyen observaciones microscópicas de la biomasa (características del flóculo, identificación y contaje de microfauna y bacterias filamentosas) y observaciones del proceso por los operadores, como la presencia y color de espuma y lodo, o la apariencia del sobrenadante del sedimentador y del efluente. En la figura 6.2 se pueden observar las tablas de atributos del reactor y del sedimentador, con las variables analíticas anteriormente comentadas.

En este caso, el sistema desarrollado se utiliza, en una primera fase, para la monitorización, la detección de problemas de funcionamiento y para dar sugerencias de operación a los operadores. Se sugieren acciones de control pero no se realizan automáticamente. El esquema de supervisión implementado se muestra en la figura 6.3.

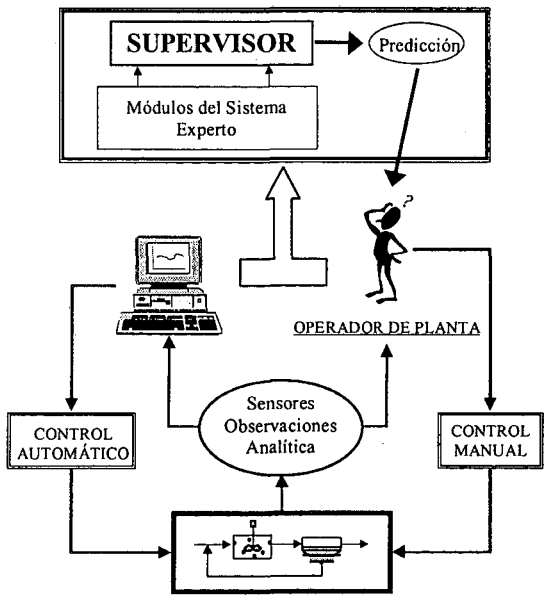


Fig. 6.3. Esquema supervisor implementado

6.1 SISTEMA DE ADQUISICIÓN DE DATOS

Se ha desarrollado un sistema de adquisición de datos para el SE. La información generada por los sensores y el estado de todos los equipos distribuidos en la EDAR es monitorizado por una red de PLCs. Estos PLCs están conectados a un PLC principal mediante un bus de comunicaciones de campo. El PLC principal está a su vez conectado al ordenador del SE mediante una interfaz RS-232-C (figura 6.4). En el ordenador un programa servidor de datos está a cargo de la comunicación con el PLC principal vía RS-232 utilizando un conjunto de mensajes predefinidos.

El servidor de datos (figura 6.5) está construido utilizando G2-GATEWAY, una herramienta para desarrollar sistemas de adquisición externos para G2 [Gensym 1997]. Éste es el nombre de la herramienta equivalente a GSI que se utiliza en la versión 5.0 de G2.

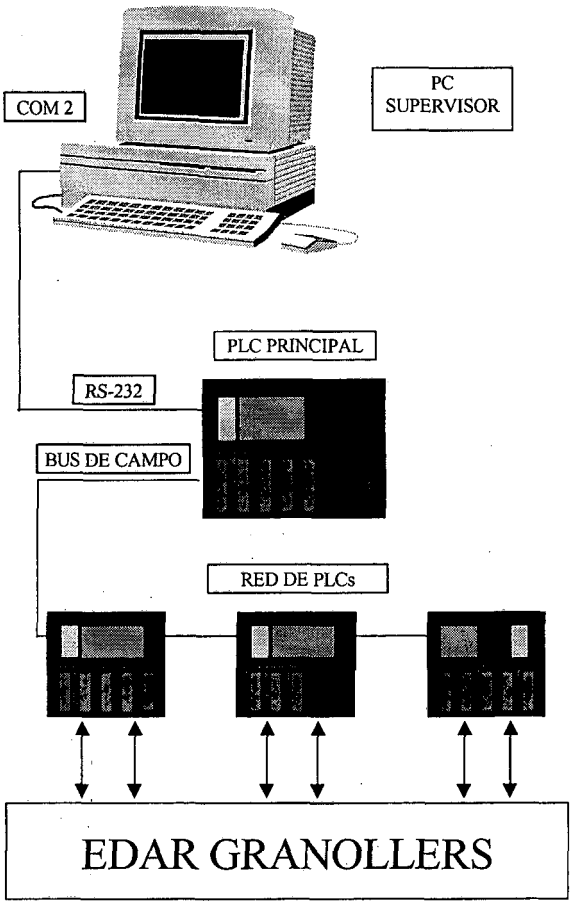


Fig. 6.4. Sistema de adquisición de datos de la

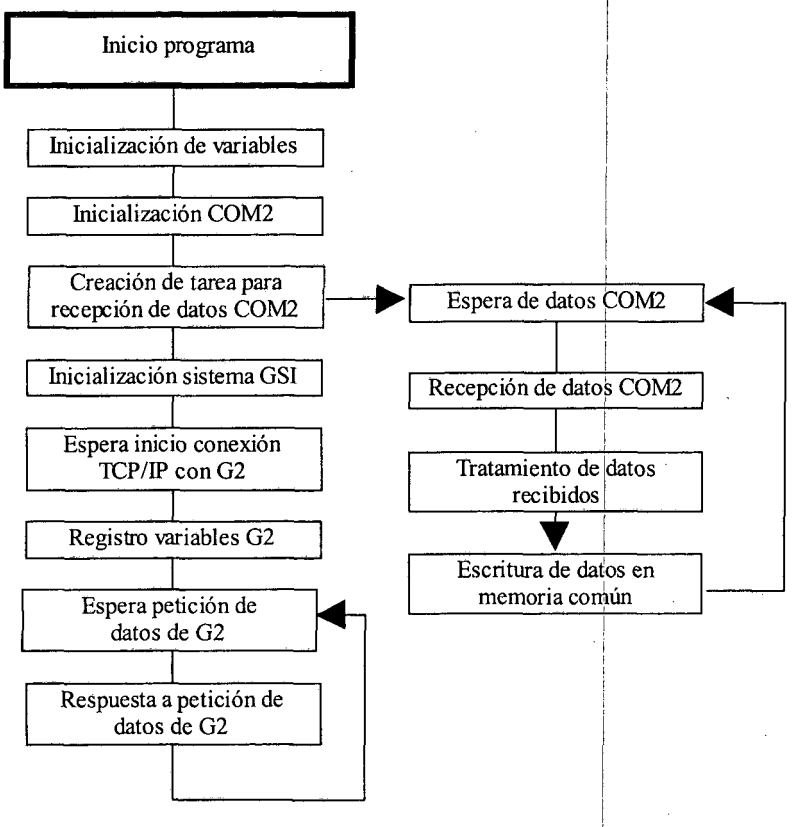


Fig. 6.5. Esquema de funcionamiento del servidor de datos

EDAR de Granollers El servidor, desarrollado en Microsoft Visual C++, es capaz de recibir una petición de datos desde G2 en cualquier momento y responder inmediatamente. Esta comunicación se establece vía TCP/IP. Esta arquitectura permite tener diferentes servidores de datos en cualquier ordenador conectado vía TCP/IP con el ordenador que ejecuta G2. En nuestro caso el servidor de datos y G2 se ejecutan en el mismo ordenador.

El funcionamiento del servidor se esquematiza en la figura 6.5. En primer lugar, el programa inicializa las variables del programa y el puerto de comunicaciones. Después de la inicialización, el programa crea una subproceso ("thread") que se encarga de la recepción de datos desde el PLC principal. Este PLC está configurado para adquirir las señales de todos los PLCs distribuidos, y posteriormente codificar esta información y enviarla mediante la salida RS-232 de que dispone.

La información enviada por el PLC es recibida en el puerto serie del PC. En la configuración utilizada es el PLC quien inicia la comunicación cada cierto tiempo prefijado. El PC siempre está a la espera de los datos enviados por el PLC, de aquí la necesidad de disponer de un proceso dedicado únicamente a ello. Una vez recibidos los datos, éstos son descodificados y posteriormente introducidos en la memoria común del programa, lo que los hace disponibles al servidor que trata las peticiones de G2. El subproceso, una vez realizado el tratamiento de datos, vuelve a situarse en espera de nuevos datos, repitiendo el procedimiento cada vez que se reciben datos.

El proceso principal inicializa el sistema GSI y se sitúa en espera de una petición de conexión por parte de G2. Cuando esto sucede, G2 envía las órdenes necesarias para el registro de las variables utilizadas en la base de conocimiento. Una vez registradas, G2 puede realizar una petición de datos en cualquier momento. El programa se mantiene siempre en el punto de espera de petición de datos, hasta la petición de desconexión desde G2.

En cuanto a las variables "off-line" analíticas, éstas pueden ser introducidas manualmente desde el SE o leídas a partir de ficheros de datos generados por el servicio de análisis de la EDAR.

En la figura 6.6 se muestra una pantalla de ejemplo de la monitorización gráfica de los valores "on-line" adquiridos.

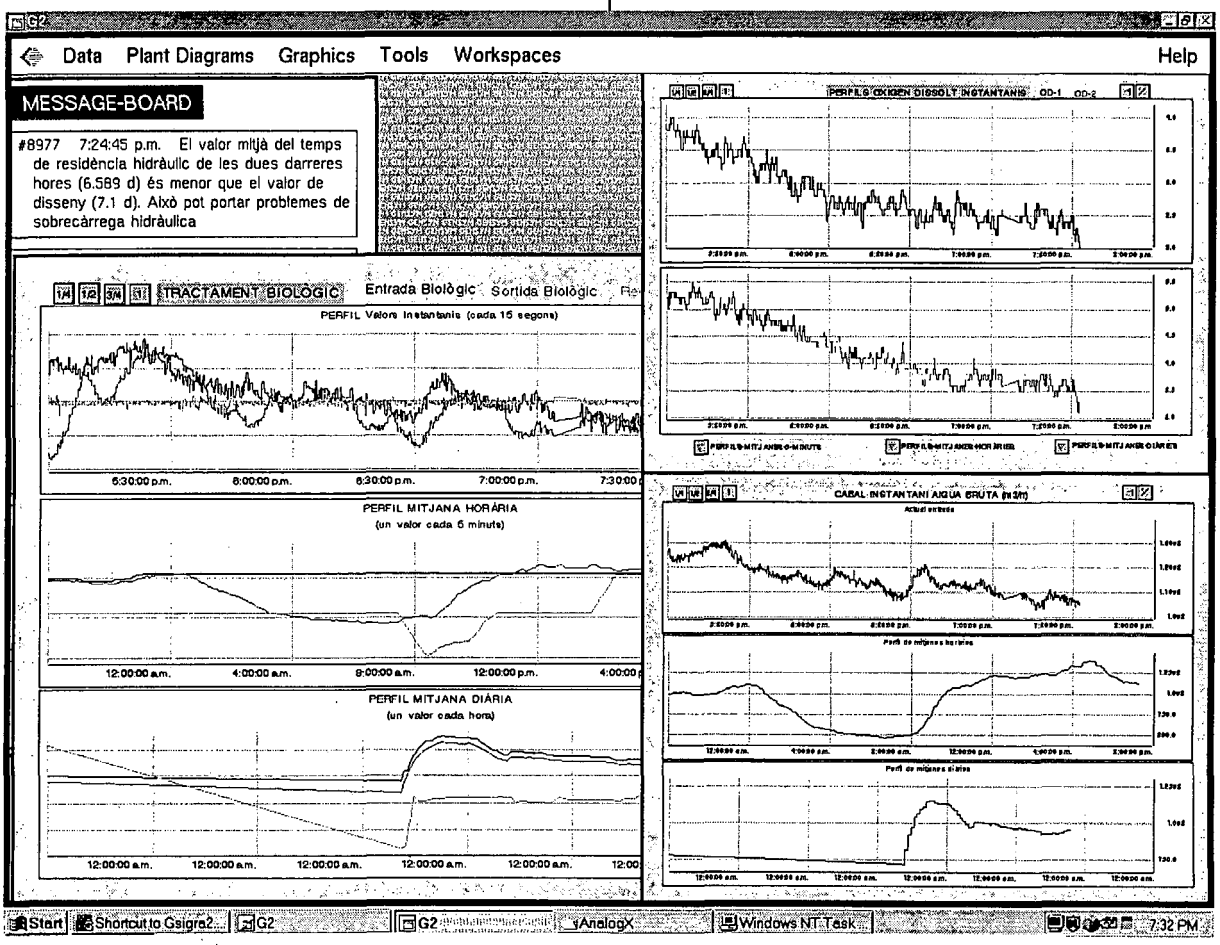


Fig. 6.6. Pantalla de monitorización de valores "on-line" de la EDAR de Granollers

6.2 SISTEMA EXPERTO

Para la construcción de SE, se ha extraído conocimiento general y específico de la EDAR a partir de una exhaustiva búsqueda bibliográfica, entrevistas con los responsables y los operadores de la planta y de una clasificación automática de la base de datos histórica. Esta información ha sido reunida, estructurada y sintetizada en un conjunto de árboles de decisión lógica para la diagnosis y la actuación.

Una vez validado por los expertos del proceso, en nuestro caso el director de la planta, estos árboles lógicos fueron codificados mediante un conjunto de reglas heurísticas que conforman la base de conocimiento del SE en tiempo real.

Utilizando como base los módulos desarrollados en la aplicación de la planta piloto, se ha implementado un nuevo sistema que contiene las características específicas (proceso y conocimiento) de la EDAR de Granollers.

La base de conocimiento cubre los problemas de operación para la sedimentación primaria y para el tratamiento secundario, incluyendo problemas no biológicos (días de lluvia, tormentas, problemas mecánicos y eléctricos, sobrecarga, baja carga, problemas de aireación y caudales no compensados) y problemas con origen biológico ("bulking", "foaming", "rising", "pinpoint", "bulking" viscoso y crecimiento disperso).

En la figura 6.7 se presenta una pantalla donde se pueden observar los módulos principales desarrollados en el sistema supervisor.

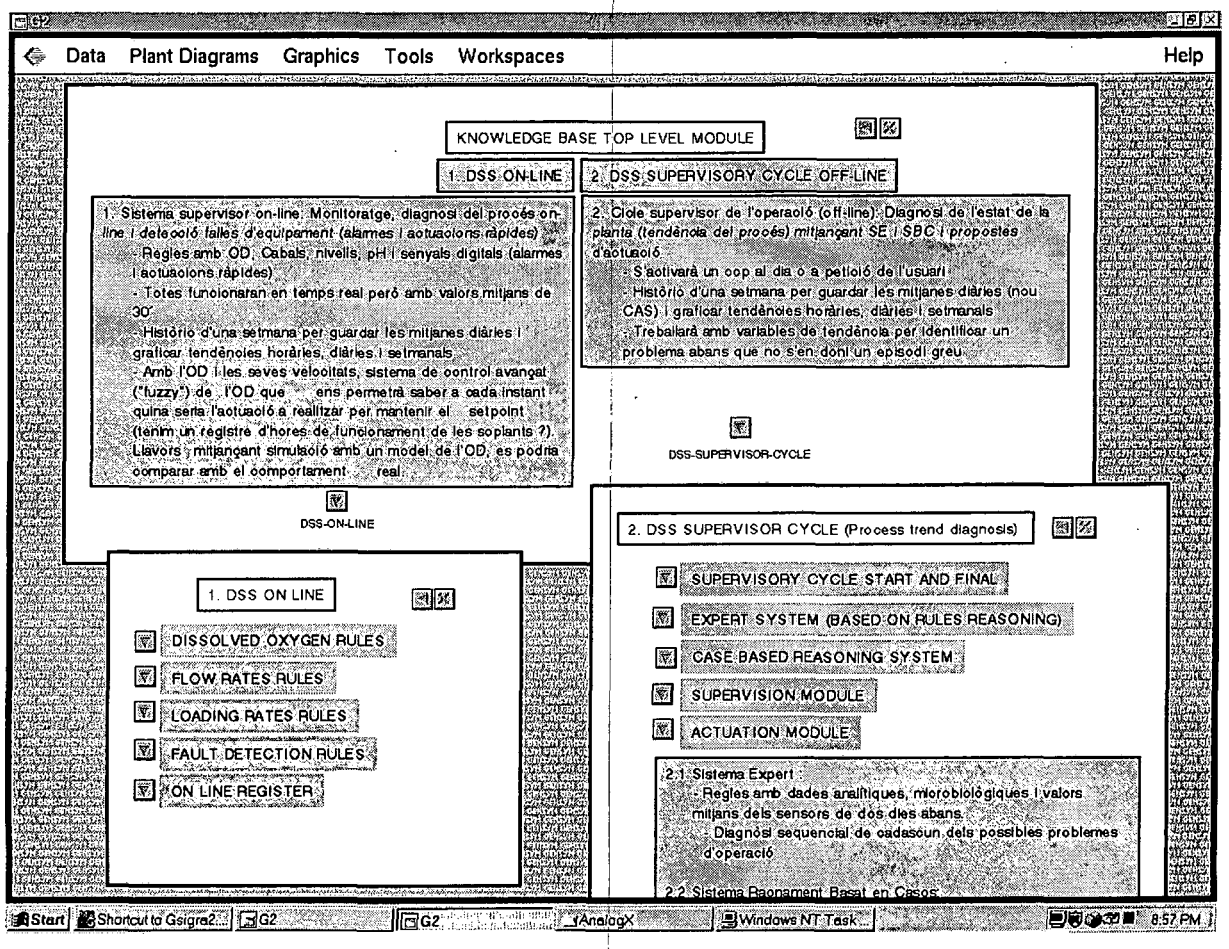


Fig. 6.7. Pantalla con los módulos principales de supervisión del sistema

6.3 APLICACIÓN

Para observar el funcionamiento del SE, a continuación se muestra su actuación frente a un caso especifico, cuyas características aparecen en la tabla 6.1.

El SE puede inferir la situación del EDAR siguiendo diferentes caminos de los arboles de decisión lógica que constituyen su base de conocimiento. En el caso estudiado, la diagnosis sigue los caminos basados en los valores de algunas variables numéricas relacionadas con el estado global del proceso (baja relación F/M y alto θ_C) y en información cualitativa (referida a la abundante presencia de espumas en la superficie del bioreactor, la proliferación de las bacterias filamentosas y en unos flóculos ligeramente dispersos). Tal como se muestra en la figura 6.8, el SE activa una alarma intermedia señalando el riesgo de la proliferación de los microorganismos filamentosos.

"On-line"	Influente 18532 m^3/d $Conductividad_{Influente}$: 3150 µS/cm Caudal de purga: 926 m^3/d
Analítica	$DQO_{Influente}$: 864 mg/l $Amonio_{Influente}$: 84 mg/l $DQO_{Primario}$: 543 mg/l $DQO_{Efluente}$: 128 mg/l $Amonio_{Efluente}$: 58 mg/l $Nitrato_{Efluente}$: 1 mg/l Biomasa: 1423 mg/l
Parámetros globales	θ_C: 9.6 days F/M: 0.22 Kg/Kg·d SVI: 117 ml/g
Microscopio	Apariencia flóculo: ligeramente disperso Biodiversidad: 6 Especie predominante: ciliados adheridos Filamentosas: abundante Filamentosa predominante: *Nocardia* *Nocardia*: abundante
Observaciones	Espuma bioreactor: abundante Turbidez sobrenadante: no

Tabla 6.1. Datos relevantes para el caso estudiado

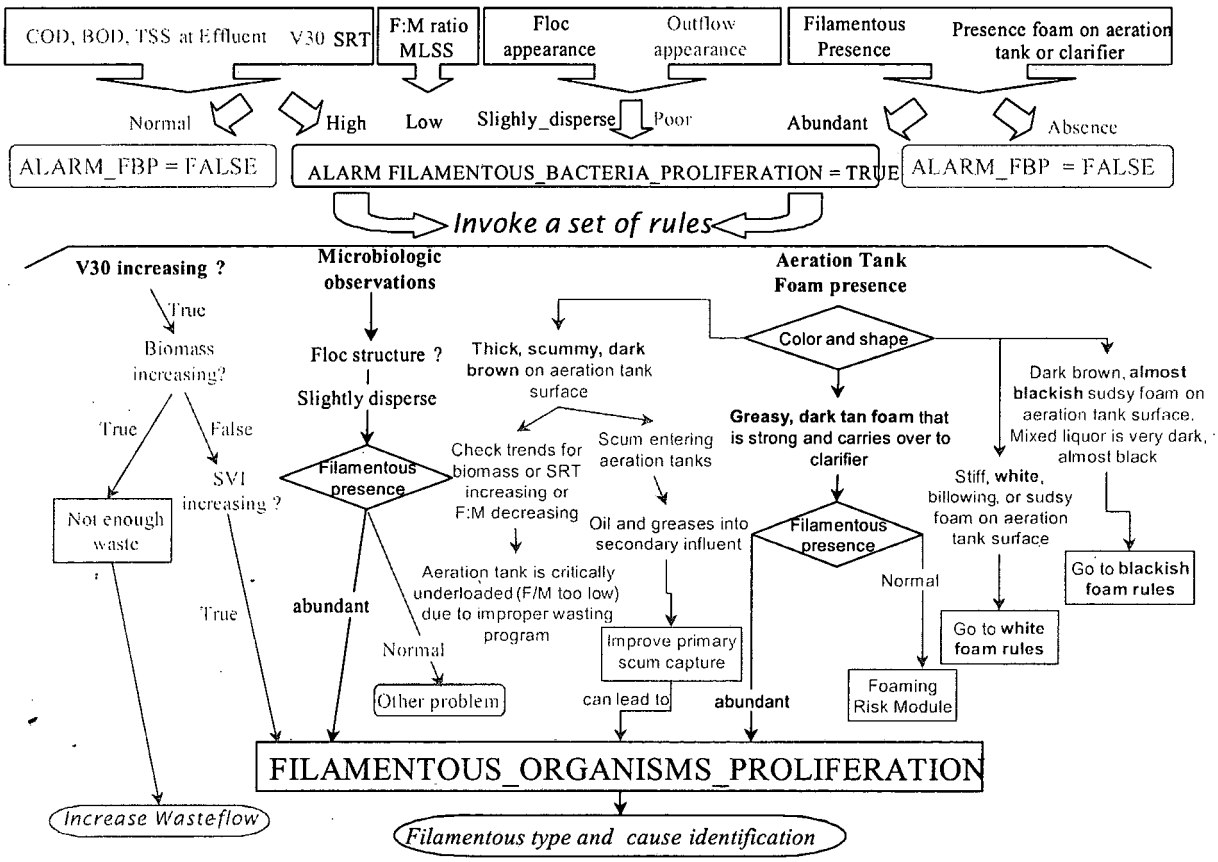


Fig. 6.8. Arbol de decisión para la prolifereación de bacterias filamentosas

En contraste, otras posibles condiciones que requieren variables cuantitativas o cualitativas relacionadas con la calidad del agua del influente y del efluente no infieren en ningún camino de diagnosis (por ejemplo valores normales de otros parámetros analíticos como SST, DQO, DBO, V_{30} o apariencia del efluente). Cuando esta alarma intermedia es activada, el motor de inferencia del SE invoca diferentes rutas de diagnosis a través de las metareglas para inferir la situación (caminos en negrita en la figura 6.8), mientras que otras reglas no son invocadas (caminos en gris en la figura). Cuando la situación es identificada como proliferación de organismos filamentosos, el SE activa un conjunto similar de reglas, correspondientes a la identificación de filamentosas.

Con la información detallada en la tabla 6.1 el SE detecta una presencia abundante de *Nocardia* y una normal de *Microthrix Parvicella,* infiriendo la situación de "foaming" causada por estas bacterias filamentosas. Entonces, el motor de inferencia de SE llama a las reglas que deben determinar las causas de la proliferación de estos microorganismos. Es importante remarcar que el SE esta todavía utilizando la misma información recogida rutinariamente en la planta, con la mínima información adicional requerida al usuario a través de la interfaz del SE.

Si se determina la causa correcta, se recomienda una actuación específica. Si la causa no es satisfactoriamente determinada, se recomienda una acción no específica.

Centrándose en el caso de estudio, el sistema es capaz de deducir las causas del "foaming" (edad del lodo elevada, o un rápido descenso en la relación F/M) y entonces, puede activar las reglas de actuación específicas correspondientes.

Supervisión y predicción del proceso

La integración de las diferentes informaciones y soluciones proporcionadas por el SE se realiza mediante el módulo supervisor. Este módulo está realizado mediante un conjunto de reglas que debe procesar la información recibida de diferentes submódulos evaluando los posibles conflictos. Además, el módulo supervisor sugiere un plan de actuación resultado de integrar las recomendaciones de los diferentes submódulos.

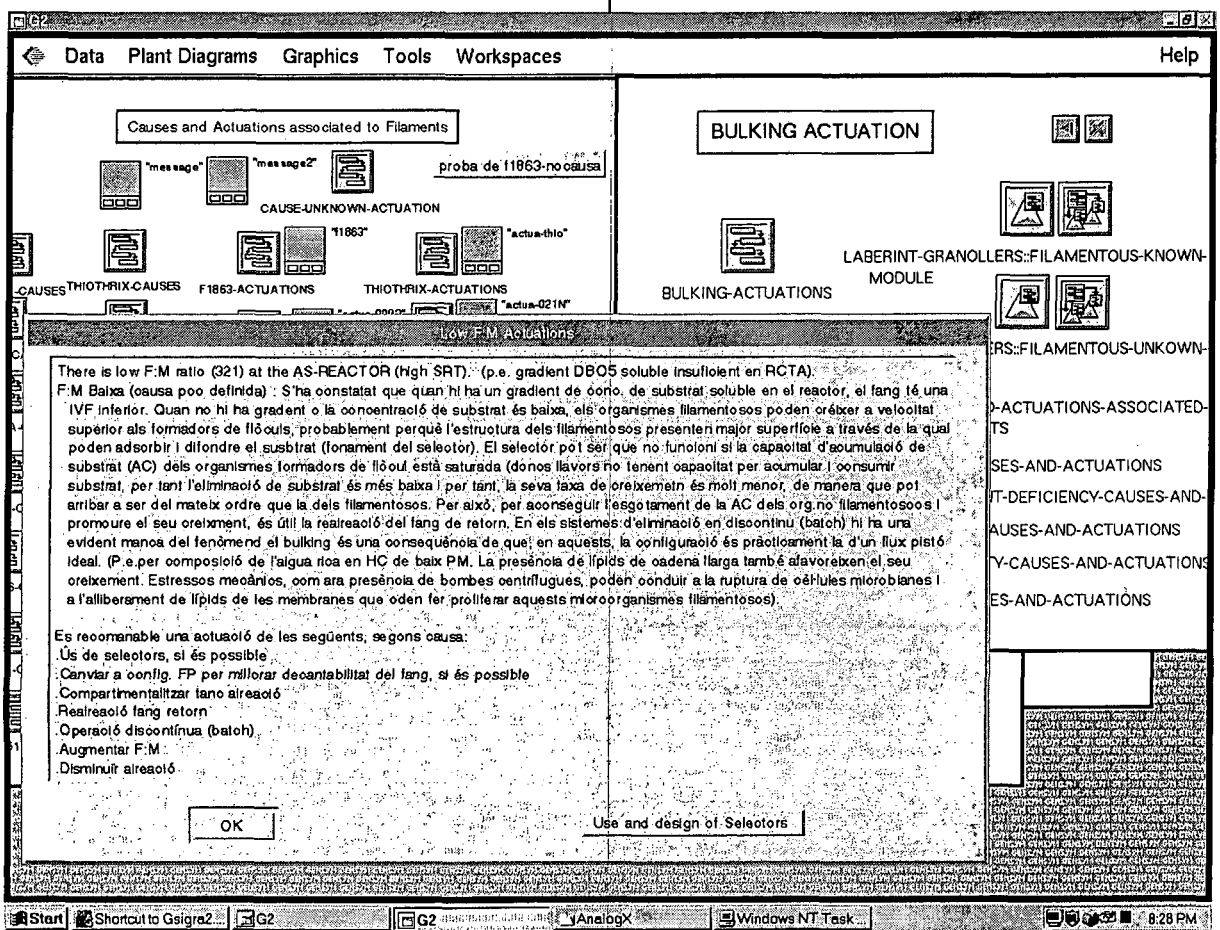


Fig. 6.9. Diagnóstico de un situación de proliferación de filamentos por baja carga

En el caso de estudio la situación inferida es la siguiente (figura 6.9):

- "Foaming" causado por *Nocardia:* la proliferación de bacterias filamentosas, como *Nocardia* y *Microthrix Parvicella,* induce la aparición de espumas de origen biológico en la superficie del reactor que se puede extender al sedimentador secundario y en algunas ocasiones puede provocar pérdida de biomasa no intencionada, reduciendo su concentración y causando sobrecarga en el bioreactor.

La solución propuesta es la siguiente estrategia de actuación:

- Incrementar el caudal de purga alrededor del 10 % cada día hasta que el proceso se acerca a los parámetros de control normales.
- Eliminar la espuma físicamente de los reactores y del sedimentador.
- Comprobar las tendencias de la población de *Nocardia* cada día.
- Si el "foaming" persiste considerar la utilización del efecto selector y la adición de cloro en la recirculación de lodos. Mandar un mensaje a los operadores recomendando como eliminar físicamente espumas y como añadir cloro (dosis, localización y prueba).

7 CONCLUSIONES

7 CONCLUSIONES

Como resultados de esta tesis se han desarrollado y optimizado un conjunto de metodologías que han permitido la construcción de un sistema de control supervisor para la gestión y control de Estaciones Depuradoras de Aguas Residuales.

El presente trabajo se inicia en el verano de 1993, cuando se implementa un sistema de adquisición de datos a una planta EDAR real con toda su instrumentación de tipo analógica distribuida por la planta.

Al intentar realizar aplicaciones de control "experto" y predictivo en tiempo real se encontró la problemática de las EDAR. Estas plantas tienen que verter el efluente a un cauce natural, por lo que no se pueden probar estrategias de operación o control que no garanticen el cumplimiento de la legislación.

Para poder demostrar la aplicabilidad de este tipo de técnicas, así como incrementar el conocimiento del proceso y desarrollar nuevas técnicas de control, se decide la construcción de una planta piloto.

PLANTA PILOTO

El trabajo desarrollado en la presente tesis se inicia con la construcción de la planta piloto. En las bases de la misma, figuran la necesidad de que esté altamente instrumentalizada y la posibilidad de tener una gran versatilidad que permita disponer de diferentes configuraciones implementables automáticamente.

El primer resultado fue la planta con tres reactores de 18 o 28 litros, que permite disponer de tres sistemas aeróbico/anóxico, con un sedimentador de 60 litros totales. Posteriormente, se introduce un cuarto reactor anaerobio para permitir la eliminación simultánea de carbono, nitrógeno y fósforo. También se dispone de un segundo sedimentador con menor volumen total.

Gracias a disponer de una planta que trabaja en continuo (1400 días de operación sobre 1500 días totales) y al "gran volumen" de trabajo, se planteó la necesidad de disponer de alimento en continuo. Por motivos de situación de la planta dentro de la universidad y por el caudal tratado (450 litros por día) se decidió la definición de un alimento que pudiera suministrarse en composición y caudal variable, y tuviera la mínima intervención manual.

Gracias a la complejidad del alimento definido se ha mantenido durante los cuatro años de trabajo una gran diversidad de microorganismos dentro del sistema de lodos activos.

Una de las mayores ventajas de este sistema experimental es el gran nivel de instrumentación de que dispone. Esta instrumentación permite el seguimiento de los parámetros normales (pH, T, OD, redox) de todos los reactores y la regulación de las principales actuaciones del sistema (velocidad de agitación, caudales de bombeo, caudal de aireación).

Una vez construido el sistema experimental se inicia el desarrollo del sistema de control y adquisición de datos distribuido (SCADA).

SCADA

El sistema de control y adquisición de datos ha sido diseñado siguiendo un esquema distribuido. El sistema consta de un PLC y diversos controladores de sondas, supervisados por un ordenador de proceso, que permite la distribución de diferentes tareas entre esos elementos.

El PLC se encarga de controlar el funcionamiento hidráulico y del sistema de alimentación automático de la planta piloto, siendo supervisado por el ordenador de proceso. Éste puede cambiar las consignas de funcionamiento del PLC, con lo que se obtiene un sistema configurable automáticamente.

Los controladores de sondas se encargan de efectuar las medidas del proceso, que son utilizadas por el ordenador de control para la adquisición de datos del proceso y para su monitorización gráfica. Estos controladores también pueden realizar control ON/OFF de las variables, como por ejemplo del pH, característica que ha sido utilizado en algunos experimentos.

El ordenador del proceso se encarga de la supervisión de los anteriores elementos y del control de otros parámetros del proceso, como por ejemplo el control de oxígeno en los reactores o los caudales de recirculación.

Este sistema desarrollado e implementado en la planta piloto, ha permitido el funcionamiento autónomo de un proceso de depuración biológica con eliminación de nutrientes. La fiabilidad del sistema se ha demostrado durante la operación de la planta durante más de cuatro años.

La metodología utilizada en la implementación de este sistema, ha permitido construir con un esfuerzo muy inferior otras aplicaciones de control, como la desarrollada para el control del sistema de monitorización analítica.

SISTEMA DE CONTROL DE LOS ANALIZADORES

Para la obtención de información analítica sobre el proceso de eliminación de nitrógeno, se ha realizado un programa que controla dos analizadores, que permiten realizar medidas de amonio, nitrato y nitrito. Estos dos analizadores, son alimentados por muestra libre de biomasa mediante un sistema automático de toma de muestras, también controlado por el mismo ordenador.

El programa desarrollado se encarga de monitorizar las concentraciones de amonio, nitrato y nitrito de los puntos de la planta preseleccionados, obteniendo la máxima frecuencia de adquisición de esos parámetros del proceso permitida por los analizadores. Esto permite la óptima utilización del equipo disponible, con lo que se mejora la información sobre el proceso.

Con los datos proporcionados por el sistema de monitorización de la planta piloto y el sistema de control de los analizadores, es posible optimizar el funcionamiento de la planta en tiempo real. Para realizar esta tarea se ha desarrollado el sistema de control experto supervisor.

SISTEMA EXPERTO

La implementación del SE hace posible el control del rendimiento del proceso en conjunto, mediante la relación entre diferentes parámetros.

La implementación del SE en la planta piloto ha supuesto la transformación de un sistema de control clásico con un comportamiento fijo en un sistema adaptable a los diferentes problemas que pueden aparecer en una EDAR. La capacidad para tratar estos problemas es útil para controlar situaciones anormales y para mantener las restricciones legales en el efluente. Además, es capaz de evitar la aparición de algunas situaciones que pueden provocar problemas a largo plazo y provocar dificultades para mantener las restricciones en el efluente para algunos parámetros.

Las acciones correctivas del control experto no son implementables en un sistema de control clásico, por lo que hay una importante mejora respecto al control clásico. Además, es posible implementar todos los algoritmos de control numérico, por lo que hay un beneficio neto de posibilidades para resolver los diferentes problemas de una EDAR.

La arquitectura distribuida multimódulo implementada mejora previos desarrollos de sistemas con bases de conocimiento monolíticas. Esta arquitectura permite el desarrollo de módulos independientes y reutilizables, que pueden ser realizados y/o utilizados por diferentes desarrolladores.

Una consecuencia de realizar un sistema modular y basado en el diseño orientado a objeto es que los módulos desarrollados pueden reutilizarse en otros sistemas expertos, adaptándose con mínimos cambios, como se puede comprobar en la aplicación en la EDAR de Granollers. Otra importante característica es el menor esfuerzo de mantenimiento respecto a versiones previas. Las tareas realizadas por diferentes módulos son fácilmente validables porque son independientes, por lo que esta tarea se simplifica. La única dificultad añadida en el esquema implementado es la integración con el módulo supervisor, que debe ser inspeccionada en detalle para evitar errores en la coordinación.

El principal éxito de este prototipo es un marco versátil capaz de tratar con diferentes configuraciones de la planta, basado en el paradigma de la programación orientada a objeto y en el razonamiento basado en reglas. La característica "on-line" es una importante innovación de este sistema, particularmente para la monitorización de datos y el control supervisor.

En el sistema desarrollado, se pueden implementar diferentes estrategias de control clásico y experto, así como configuraciones de planta para el sistema de lodos activos con eliminación de materia orgánica, nitrógeno y fósforo.

El mencionado sistema se ha validado funcionando durante un periodo ininterrumpido de 30 meses. Es destacable el cambio del modo de operación de la planta que ha supuesto la implementación del sistema de control, pasando de una situación de gestión semiautomática (1996) inicial a que el sistema de gestión desarrollado sea una necesidad ineludible en el funcionamiento diario de la planta.

En este periodo, los objetivos operacionales de la planta fueron cambiados periódicamente, para optimizar diferentes partes del proceso. El SE ha mostrado un excelente rendimiento en la operación de la planta piloto. El sistema desarrollado detecta y controla todos los problemas y operaciones especiales de la planta, como por ejemplo fallo de la bomba, problemas de alimentación, funcionamiento incorrecto de las sondas, mantenimiento del equipo y control de los analizadores. Esto ha permitido una disminución en la atención necesaria para el mantenimiento de la planta piloto en unas condiciones adecuadas de funcionamiento.

VALIDACIÓN EXPERIMENTAL

Para la validación de todo el sistema supervisor se han realizado una serie de experimentos para optimizar la eliminación de nitrógeno en condiciones dinámicas.

Esta optimización se ha basado en dos objetivos principales, la obtención de la eliminación de nitrógeno prefijada y la utilización de los mínimos recursos posibles, manteniendo, a su vez, una adecuada eliminación de materia orgánica y fósforo.

En la aplicación de las estrategias de operación implementadas, se ha detectado que la validez de su aplicación depende mucho de la situación en que se encuentre la planta en ese momento. Se ha demostrado que la utilización de las reglas expertas para la aplicación de una u otra estrategia, permite utilizarlas correctamente, con lo que se mejora el rendimiento del proceso.

Utilizando distintas estrategias de operación se ha obtenido una mejora global en la eliminación de nitrógeno. Esta mejora ha sido superior al 16 % en cuanto a eliminación por unidad de biomasa, y se ha obtenido una disminución en el vertido de nitrógeno superior al 46 %.

Además de las mejoras de eliminación, ha mejorado la fiabilidad del proceso, ya que han sido implementadas las acciones correctoras posibles cuando se ha detectado algún problema de operación.

Como conclusión final, se puede afirmar que el sistema de control supervisor desarrollado es una herramienta válida para el control y gestión en tiempo real de plantas depuradoras, que permite la mejora de los procesos de depuración biológica, en cuanto a fiabilidad de operación, mejora del rendimiento de eliminación y reducción del coste energético necesario para obtener un determinado grado de depuración. Se ha demostrado que el sistema es fácilmente implementable en una EDAR real, mejorando la visión general del proceso y de los problemas de operación al operador experimentado, así como el aprendizaje a los operadores con poca preparación.

8 NOMENCLATURA

CONTROL

BC	Base Conocimiento
DDC	Digital Direct Control
DDDC	Distributed Digital Direct Control
GSI	Gensym Standard Interface
HDSE	Herramientas de desarrollo de sistemas expertos
HTML	HyperText Mark-up Language
IAD	Inteligencia Artificial Distribuida
KB	Knowledge Base
KBES	Knowledge Base Expert System
KBS	Knowledge Based System
PERL	Practical Extraction and Report Language
PID	Proporcional Integral Derivado
PLC	Programmable Logic Controller
RTES	Real Time Expert System
SBC	Sistema Basado en el Conocimiento
SCADA	Sistema de Control y Adquisición de Datos
SE	Sistema experto
SEBC	Sistema Experto Basado en el Conocimiento
SSC	Supervisory Setpoint Control
TCP/IP	Transfer Control Protocol / Internet Protocol

PROCESO

AGV	Ácidos Grasos Volátiles
CFA	Análisis de Flujo Continuo
COT	Carbono Orgánico Total
DBO	Demanda Bioquímica de Oxígeno
DQO	Demanda Química de Oxígeno
DSVI	Diluted Sludge Volume Index
DTO	Demanda Total de Oxígeno
EDAR	Estación Depuradora de Aguas Residuales
FIA	Análisis por inyección en flujo
IVFD	Índice Volumétrico de Fangos Diluidos
N/D	Nitrificación/Desnitrificación
NKT	Nitrógeno Kheldal Total
NT	Nitrógeno Total
OAF	Microorganismos Acumuladores de Fósforo
OD	Oxígeno Disuelto
ORP	Oxidation-Reduction Potential
OUR	Oxygen Uptake Rate
PHA	Polihidroxialcanos
PHB	Poli-ß-hidroxibutirato
Q_E	Caudal de entrada a la planta
Q_{RE}	Caudal de recirculación externa
Q_{RI}	Caudal de recirculación interna
Q_{RI}/Q_E	Relación de recirculación interna
Q_{RE}/Q_E	Relación de recirculación externa
SBI	Índice biótico de fangos activos
SBR	Sequential Batch Reactor
SRT	Sludge Retention Time
SSV	Sólidos en Suspensión Volátiles
SST	Sólidos en Suspensión Totals
SVI	Sludge Volumen Index
TCA	Ciclo del Ácido Tricarboxílico
θc	Tiempo de residencia celular
$[\ ]_E$	Concentración del parámetro en la entrada
EPA	Environmental Protection Agency
IAWQ	International Association on Water Quality

9 BIBLIOGRAFÍA

9 BIBLIOGRAFÍA

Andrews JF. Andrews JF, editors. Dynamics and Control of the Activated Sludge Process. Lancaster, Pennsylvania: Technomic Publishing Company, Inc. 1992; 8, Knowledge-Based (Expert) System for the Activated Sludge Process. p. 231-43.

Andrews JF. Andrews JF, editors.Dynamics and Control of the Activated Sludge Process. Lancaster, Pennsylvania: Technomic Publishing Company, Inc. 1992; 7, Application of Artificial Intelligence (AI) Techniques in Activated Sludge Process Design, Operations, and Control. pp. 207-29.

Anthonisen AC, Loehr RC, Prakasan TBS, and Srineth EG. Inhibition of nitrification by ammonia and nitrous acid. *J. Wat. Poll. Con. Fed.* 48. pp.835-852.(1976).

Antoniou P. Effect of temperature and pH on the effective maximum specific growth rate of nitrifying bacteria. *Wat. Res.* 24. pp.97 (1990).

APHA.Standard methods. 18th ed. American Publishers Health Association.1993

Aspegren H, Andersson B, Nyberg U, and Jansen Jl. Model and sensor based optimization of nitrogen removal at klagshamn wastewater treatment plant. *Wat. Sci. Tech.* 26. 5-6. pp.1315-1323.(1992)

Aström KJ and McAvoy TJ. Intelligent control. *J. Proc. Cont.* 2. 3. pp.115-127.(1992)

Aynsley M, Hofland A, Morris AJ, Montague GA, and Di Massimo C. Blanch HW, Bungay HR, Cooney CL, et al, editors. Bioprocess Design and Control. Berlin Heildelberg: Springer-Verlag, 1993; 1, Artificial Intelligence and the Supervision of Bioprocesses. (Real-Time Knowledge-Based Systems and Neural Networks). pp 1-27.

Baeza J, Ferreira EC, and Lafuente J. Real-Time Expert Control of a Pilot WWTP with Nitrogen and Phosphorus Removal.Mota, M. and Ferreira, E.C. editors. Departamento de Engenharia Biológica, Universidade do Minho; 1998 Jul; Braga, Portugal. 1998; 321 p. BIOTEC'98. Book of abstracts.

Baeza J. Diseño y diseño de un sistema de control supervisor y adquisición de datos: Aplicación a una planta piloto depuradora de aguas residuales urbanas con criterios de nitrificación/desnitrificación. *Tesis de Licenciatura.* Universitat Autònoma de Barcelona. Bellaterra. 1996.

Baeza J, Gabriel D, and Lafuente J. An expert supervisory system for a pilot WWTP. *Environmental Modelling & Software* . 14. pp.383-390.(1999)

Balslev P, Lynggaard-Jensen A, and Nickelsen C. Nutrient sensor based real-time on-line process control of a wastewater treatment plant using recirculation. *Wat. Sci. Tech.* 33. 1. pp.183-192.(1996)

Bañares-Alcántara, R.; Ponton, J.W. Artificial Intelligence Techniques in Chemical Engineering Process Design. *Applications of Artificial Intelligence.* VII. pp 581-607.(1992)

Barker PS and Dold PL. Denitrification behaviour in biological excess phosphorus removal activated sludge systems. *Wat. Res.* 30. 4. pp.769-780.(1996)

Barnett MW, Patry GG, and Hiraoka M. Andrews JF, editors.Dynamics and control of the activated sludge process. Lancaster, Los Angeles: Technomic Publ.Co. 1992;Knowledge-based (expert) systems for the activated sludge process. p. 231-43.

Beck, M.B. Identification, estimation and control of biological waste-water treatment processes. *IEEE* Proceedings. 133, 254-264.(1986)

Belanche LA, Valdés JJ, Comas J, R-Roda I, and Poch M. Towards a model of input-output behaviour of wastewater treatment plants using soft computing techniques. *Environmental Modelling & Software* . 14. pp.409-419.(1999)

Benefield; Randall. Biological process design for wastewater treatment. Prentince Hall Inc. N.J.1980.

Berk SG and Gunderson JH.Wastewater organisms.A color atlas. Florida: Lewis.1993.

Béjar J, Cortés U, Sànchez M, Gimeno JM, and Poch M. Applying domain knowledge to the discovery of operating situations in wastewater treatment plants. H.Adeli, IEEE Computer Society; Los Alamitos, California,U.S.A. 1997; p 360.

Bel M de, Stokes L, Upton J, and Watts J. Applications of a respirometry based toxicity monitor. *Wat. Sci. Tech.* 33. 1. pp.289-296.(1996)

Bond AH and Gasser L. A subject-indexed bibliography of distributed artificial intelligence. 1992; 1260 p.

Borer J.AnonymousInstrumentation and contol for the process industries. London and New York: Elsevier applied science publishers, 1985; 7, Transmission of measured data. pp 123-52.

Brdjanovic D, Slamet A, Van Loosdrecht MCM, Hooijmans CM, Alaerts GJ, and Heijnen JJ. Impact of excessive aeration on biological phosphorus removal from wastewater. *Wat. Res.* 32. 1. pp.200-208.(1998)

Brenner A. Use of computers for process design analysis and control: sequencing batch reactor application. *Wat. Sci. Tech.* 35. 1. pp.95-104.(1997)

Brett S, Guy J, Morse GK, and Lester JN. Brett S, editor.Phosphorus removal and recovery technologies. London, Great Britain: Selper Publications; 1997;

Brouwer H, Bloemen M, Klapwijk B, and Spanjers H. Feedforward control of nitrification by manipulating the aerobic volume in activated sludge plants. *Wat. Sci. Tech.* 38. 3. pp.245-254.(1998)

Bundgaard E, Nielsen MK, and Henze M. Process development by full-scale on-line tests and documentation. *Wat. Sci. Tech.* 33. 1. pp.281-287.(1996)

Burrell PC, Keller J, and Blackall L. Microbiology of a nitrite-oxidizing bioreactor. *Applied and Environmental Microbiology.* May. pp.1878-1883.(1998)

Cabezut Boo O and Sànchez Aguilar A. A Knowledge Base for Wastewater Treatment Plants: Case of an Activated-Sludge Facility. *Expert Systems with Applications.* 14. 1.(1998)

Cabezut Boo O and Sánchez-Aguilar A. Towards an ontology of waste water treatment plants: the identification phase. *Environmental Modelling & Software* . 14. pp.401-408.(1999)

Campbell, J. El libro del RS232. Anaya.1988.

Campmajó Riera, C.. Desenvolupament d'un sistema expert pel control en temps real d'un procés biotecnològic. *Tesis de Licenciatura.* Universitat Autònoma de Barcelona. Bellaterra. 1991.

Cerf, V.G. Redes. *Investigación y Ciencia.* Noviembre, 16-26.(1991).

Cao S and Rhinehart RR. A sel-tuning filter. *J. Proc. Cont.* 7. 2. pp.139-148.(1997)

Capodaglio AG, Jones HV, Novotny V, and Feng X. Sludge bulking analysis and forecasting: application of system identification and artificial neural computing technologies. *Wat. Res.* 25. 10. pp.1217-1224.(1991)

Carlsson H, Aspegren H, and Hilmer A. Interactions between wastewater quality and phosphorus release in the anaerobic reactor of the EBPR process. *Wat. Res.* 30. 6. pp.1517-1527.(1996)

Carrera J. Depuración biológica de nutrientes en aguas residuales urbanas: Estudio de los parámetros operacionales de la eliminación de fósforo. *Tesis de Licenciatura.* Universidad Autónoma de Barcelona. Bellaterra.1997

Cartensen J, Harremoës P, and Strube R. Software sensors based on the grey-box modelling approach. *Wat. Sci. Tech.* 33. 1. pp.117-126.(1996)

Chang WC, Ouyang CF, Chiang WL, and Hou CW. Sludge pre-recycle control of dynamic enhanced biological phosphorus removal system: an application of on-line fuzzy controller. *Wat. Res.* 32. 3. pp.727-736.(1998)

Charpentier J, Martin G, Wacheux H, and Gilles P. ORP regulation and activated sludge:15 years

of experience. *Wat. Sci. Tech.* 38. 3. pp.197-208.(1998)

Chuang SH, Ouyang CF, Yuang HC, and You SJ. Evaluation of phosphorus removal in anaerobic-anoxic-aerobic system - via polyhydroxyalkanoates measurements. *Wat. Sci. Tech.* 38. 1. pp.107-114.(1998)

Cohen A, Janssen G, Brewster SD et al. Application of computational intelligence for on-line control of a sequencing batch reactor (SBR) at Morrinsville sewage treatment plant. *Wat. Sci. Tech.* 35. 10. pp.63-71.(1997)

Comeau Y, Hall KJ, Hancock REW, and Oldham WK. Biochemical model for enhanced biological phosphorus removal. *Wat. Res.* 20. pp.1511-1521.(1986)

Connell B.AnonymousProcess instrumentation applications manual. New York: McGraw-Hill, 1996; 2, How instruments are selected. p. 17-34.

Copp JB and Dold PL. Confirming the nitrate-to-oxygen conversion factor for denitrification. *Wat. Res.* 32. 4. pp.1296-1304.(1998)

Çakici A and Bayramoglu M. An approach to controlling sludge age in the activated sludge process. *Wat. Res.* 29. 4. pp.1093-1097.(1995)

Demoulin G, Goronszy MC, Wutscher K, and Forsthuber ECO-current nitrification /denitrification and biological P-removal in cyclic activated sludge plants by redox controlled cycle operation. *Wat. Sci. Tech.* 35. 1. pp.215-224.(1997)

Delwiche C.C. The nitrogen cycle. *Scientific Am.* 223. 3. pp 137-146. (1970)

Dochain D and Perrier M. Control design for nonlinear wastewater treatment processes. *Wat. Sci. Tech.* 28. 11-12. pp.283-293.(1993)

EPA. Ambient water quality criteria fo ammonia-1984. Washington, DC. U.S. EPA. 1985; EPA/440/5-85/001.

EPA. Ambient water quality criteria for ammonia (saltwater) - 1989. Washington, D.C. U.S. EPA. 1989; EPA/440/5-88/004.

EPA.Manual of Nitrogen Control. EPA/625/R-93/010 ed. Washington, U.S.A. U.S. Environmental Protection Agency; 1993;

Folger, H.S. Elements of Chemical Reaction Engineering. Englewood Cliffs,N.J.1990

Forty TR. Nutrient control in Okanagan Valley lakes. *Water Quality International.* November/December. pp.51-54.(1998)

Froment, G.F.; Bischoff, K.B. Chemical Reactor Analysis and Design. Wiley. New York, USA.1990.

Fu CS and Poch M. System indentification and real-time pattern recognition by neural networks for an activated sludge process. *Environment international.* 21. pp.57-69.(1995)

Fuente MJ, Vega P, Zarrop M, and Poch M. Fault detection in a real wastewater plant using parameter estimation techniques. *Control Eng. Practice.* 4. pp.1089-1098.(1996)

Gabriel D, Baeza J, Valero F, and Lafuente J. A novel FIA configuration for the simultaneous determination of nitrate and nitrite and its use for monitoring an urban waste water treatmment plant based on N/D criteria. *Analytica Chimica Acta.* 359. pp.173-183.(1998)

Gabriel D. Disseny i automatització d'un sistema de monitorització de nitrats, nitrits i amoni mitjançant tècniques d'anàlisi en flux: Aplicació a una planta pilot depuradora d'aigües residuals urbanes amb criteris de Nitrificació/Desnitrificació. *Tesis de Licenciatura.* Universitat Autònoma de Barcelona. Bellaterra. 1996.

Gacot, R.; Smith, B. Tapping into Sockets. *Byte.* March, *261*-268.(1992).

Gall RAB and Patry GG. Patry GG, Chapman D, editors.Dynamic modeling and expert systems in wastewater engineering. Chelsea: Lewis, 1989;Knowledge-based system for the diagnosis of an activated sludge plant. p. 193-240.

Gensym. G2 reference manual version 4.0. Gensym corporation: Cambridge; 1995;

Gensym. G2-Standard Interface. User's manual version 3.2. Cabridge, MA: Gensym corporation; 1995;

Gensym. G2-Gateway, bridge developer's guide version 5.0. Cabridge, MA: Gensym corporation; 1997;

Gensym. G2 reference manual version 5.0. Gensym corporation: Cambridge; 1997;

Gernaey K, Vanrolleghem P, and Verstraete W. On-line estimation of Nitrosomonas kinetic parameters in activated sludge samples using titration in-sensor-experiments. *Wat. Res.* 32. 1. pp.71-80.(1998)

Giroux ÉY, Spanjers H, Patry GG, and Takàcs I. Dynamic modelling for operational design of a respirometer. *Wat. Sci. Tech.* 33. 1. pp.297-309.(1996)

Givens, S.W.; Gelman, S.R.; Brown, E.V.; Leslie Grady, C.P.; Skedsvold, D.A. Biological process design and pilot testing for a carbon oxidation, nitrification, and denitrification system. *Environmental Progress.* 10,2, 133-146.(1991)

Grijspeerdt K and Verstraete W. A sensor for the secondary clarifier based on image analysis. *Wat. Sci. Tech.* 33. 1. pp.61-70.(1996)

Guang D, Wiley DE, Hlavacek M, and Fane G. On-line automatic sampling for real time monitoring of wastewaters. *Wat. Res.* 30. 11. pp.2651-2654.(1996)

Harremoës P, Haarbo A, Winther-Nielsen M, and Thirsing C. Six years of pilot plant studies for design of treatment plants for nutrient removal. *Wat. Sci. Tech.* 38. 1. pp.219-226.(1998)

Harremoës P and Sinkjaer O. Kinetic interpretation of nitrogen removal in pilot scale experiments. *Wat. Res.* 29. 3. pp.899-905.(1995)

Häck M and Köhne M. Estimation of wastewater process parameters using neural networks. *Wat. Sci. Tech.* 33. 1. pp.101-115.(1996)

Hitzmann B, Lübbert A, and Schügerl K. An Expert System Approach for the Control of a Bioprocess. I: Knowledge Representation and Processing. *Biotechnol. Bioeng.* 39. pp.33-43.(1992)

Hoen K, Schuhen M, and Köhne M. Control of nitrogen removal in wastewater treatment plants with predenitrification, depending on the actual purification capacity. *Wat. Sci. Tech.* 33. 1. pp.223-236.(1996)

Horan NJ and Eccles CR. The potential for expert systems in the operation and control of activates sludge plants. *Process Biochemestry.* June. pp.81-85.(1986)

IDEC. *CLIP*: Computer Logic Input Program. User's Manual. IDEC.1992.

Isaacs S and Temmink H. Experiences with automatic N and P measurements of an activated sluge process in a research environment. *Wat. Sci. Tech.* 33. 1. pp.165-173.(1996)

Jenkins D, Richard MG, and Daigger GT.Manual on the causes and control of activated sludge bulking and foaming. Michigan: Lewis; 1993;

Jennings NR. Cooperation in industrial multi-agent systems. New Jersey, NJ. World Scientific Pub. Co. 1994; World Scientific Series in Computer Science 43

Jennings NR and Mamdani EH. Using joint responsibility to coordinate collaborattive problem solving in dynamic environments. MIT Press; San José, California. 1992; 269 p.

Johansen NH, Anderen JS, and la Cour Jansen J. Optimum operation of a small sequencing batch reactor for BOD and nitrogen removal based on on-line OUR calculation. *Wat. Sci. Tech.* 35. 6. pp.29-36.(1997)

Jones M and Stephenson T. The effects of temperature on enhanced biological phosphate removal. *Environment Technology.* 17. pp.965-976.(1996)

Kanaya T, Hirabayashi K, Fujita I, and Tsumura K. Detection of unusual data in online monitoring of wastewater processing. *Wat. Sci. Tech.* 33. 1. pp.71-79.(1996)

Kernighan B.W.; Ritchie, D.M. The C programming Language. 2a ed. Prentince Hall Inc. Englewood Cliffs, N.J.1988.

Klapwijk A, Brouwer H, Vrolijk E, and Kujawa K. Control of intermittently aerated nitrogen removal plants by detection endpoits of nitrification and denitrification using respirometry only. *Wat. Res.* 32. 5. pp.1700-1703.(1998)

Knight GC, Serviour EM, Serviour RJ et al. Development of the microbial community of a full scale biological nutrient removal activated sludge plant during start-up. *Wat. Res.* 29. 9. pp.2085-2093.(1995)

Kolkwitz R and Marsson M. Ökologie der tierische saprobien. Beiträge zur Lehre von der biologische gewasserbeurteilung. *Int Revue ges Hydrobiol.* 2. pp.126-162.(1935)

Kong Z, Vanrolleghem P, Willems P, and Verstraete W. Simultaneous determination of inhibition kinetics of carbon oxidation and nitrification with a respirometer. *Wat. Res.* 30. 4. pp.825-836.(1996)

Konstantinov KB, Aarts R, and Yoshida T. Blanch HW, Bungay HR, Cooney CL, et al, editors. Bioprocess Design and Control. Berlin Heildelberg: Springer-Verlag, 1993; 6, Expert Systems in Bioprocess Control, Requisite Features. p. 169-91.

Konstantinov KB and Yoshida T. Real-Time Qualitative Analysis of the Temporal Shapes of (Bio)process Variables. *AIChE Journal.* 38. 11. pp.1703-1715.(1992)

Konstantinov KB and Yoshida T. Knowledge-Based Control of Fermentation Processes. *Biotechnol. Bioeng.* 39. pp.479-486.(1992)

Krichten DJ, Wilson KD, and Tracy KD. Expert Systems Guide Biological Phosphorus Removal. *Water Environment & Technology.* October. pp.60-64.(1991)

Krovvidy, S.; Wee, W.G. Wastewater Treatment Systems from Case-Based Reasoning. Machine *Learning.* 10, pp 341-346.(1993)

Krovvidy, S.; Wee, W.G.; Summer, R.S.; Coleman, J.J. An A-I aproach for wastewater treatment systems. *Journal of Applied Intelligence.* 1, pp 247-261.(1991)

Kuhn, T.S. The Structure of Scientific Revolutions. University of Chicago Press. 1962

Lapointe J, Marcos B, Veillette M, Laflamme G, and Dumontier M. Bioexpert: an expert system for wastewater treatment process diagnosis. *Computers chem. Engng.* 13. 6. pp.619-630.(1989)

Leeuw EJ, Kramer JF, Bult BA, and Wijcherson MH. Optimization of nutrient removal with on-line monitoring and dynamic simulation. *Wat. Sci. Tech.* 33. 1. pp.203-209.(1996)

Leu H-G, Lee C-D, Ouyang CF, and Tseng H-T. Effects of organic matter on the conversion rates of nitrogenous compounds in a channel reactor under various flow conditions. *Wat. Res.* 32. 3. pp.891-899.(1998)

Liebmann H. Fisher G, editor.Handbuch der Frischwasser und abwasserbiologic. Jena: 1962;

Lindberg C. Control and estimation strategies applied to the activated sludge process Uppsala University, Departament of Materials Science, Systems and Control Grup; 1997; 1 p.

Lindberg C and Carlsson B. Estimation of the respiration rate and oxygen transfer function utilizing a slow DO sensor. *Wat. Sci. Tech.* 33. 1. pp.325-333.(1996)

Lindberg C and Carlsson B. Adaptive control of external carbon flow rate in an activated sludge process. *Wat. Sci. Tech.* 34. 3-4. pp.173-180.(1996)

Lo, C.K.; Yu, C.W.; Tam, N.F.Y.; Traynor, S. Enhaced nutrient removal by oxidation-reduction potential (ORP) controlled aeration in a laboratory scale extended aeration treatment system. *Wat. Res.* 28, 10, 2087-2094.(1994)

Londong J and Wachtl P. Six years of practical experience with the operation of on-line analysers. *Wat. Sci. Tech.* 33. 1. pp.159-164.(1996)

Luyben, W.L. Process Modeling Simulation and Control for Chemical Engineers. 2a ed. Mc.Graw *Hill* Publishing Co.. New York, USA. 1990

Lynggaard-Jensen A, Eisum NH, Rasmussen I, Svankjaer Jacobsen H, and Stenstrom T. Description and test of a new generation of nutrient sensors. *Wat. Sci. Tech.* 33. 1. pp.25-35.(1996)

Madoni P. A sludge biotic index (SBI) for the evaluation of the biological performance of activated sludge based on the microfauna analysis. *Wat. Res.* 28. pp.67-75.(1994)

Maeda, K. An Intelligent Decision Support System for Activated Sludge Wastewater Treatment Process. Instrumentation and control of water and wastewater treatment and transport systems.(1985).

Maeda, K. A knowledge-based system for the wastewater treatment plant. *Future* Generation *Computer Systems.* 5, pp 29-32. (1989)

Manesis SA, Sapidis DJ, and King RE. Intelligent control of wastewater treatment plants. *Artificial Intelligence in Engineering.* 12. pp.275-281.(1998)

Martinelle K and Häggström L. On the disociation constant of ammonium: effect of using an incorrect pKa in calculations of the ammonia concentration in animal cell cultures. *Biotechnol. Tech.* 11. 8. pp.549-551.(1997)

McCarty PL and et al. Sources of nitrogen and phosphorus supplies. *JAWWA.* 59. 344.(1967)

Menzl S, Stühler M, and Benz R. A self adaptive computer-based pH measurement and fuzzy-control system. *Wat. Res.* 30. 4. pp.981-991.(1996)

Metcalf and Eddy. McGraw-Hill, editor.Ingenieria de aguas residuales.Tratamiento, vertido y reutilización. 3rd ed. Madrid: 1995;

Mino T, Arun V, Tsuzuki Y, and Matsuo T. Ramadori R, editors.Biological phosphate removal from wastewaters. 1987;Effect of phosphorus accumulation on acetate metabolism in the biological phosphorus removal process. p. 27-38.

Momba MNB and Cloete TE. The relationship of biomass to phosphate uptake by Acinetobacter Junii in activated sludge mixed liquor. *Wat. Res.* 30. 2. pp.364-370.(1996)

Moreno, R.. Estimación de estados y control predictivo del proceso de fangos activados. Bellaterra.(1994)

Moreno R, de Prada C, Lafuente J, Poch M, and Montague GA. Non-linear predictive control of dissolved oxygen in the activated sludge process. Colorado. 1992; 289 p.

Morley RE and Schelberg C. An analysis of a plant-specific dynamic scheduler. Proceedings of NSF Workshop on Dynamic Scheduling. Cocoa Beach, FL. 1993;

Moulin B and Chaib-Draa B. Jennings NR, editors.O'Hare, G.M.P. New York: Wiley,J, 1996;An overview of distributed artificial intelligence. p. 3-55.

Müller A, Marsili-Libelli S, Aivasidis A, Lloyd T, Kroner S, and Wandrey C. Fuzzy control of disturbances in a wastewater treatment process. *Wat. Res.* 31. 12. pp.3157-3167.(1997)

Nielsen MK and Önnerth TB. Strategies for handling of on-line information for optimising nutrient removal. *Wat. Sci. Tech.* 33. 1. pp.211-222.(1996)

O'Neill M and Horan NJ. Achievig simultaneous nitrification and denitrification of wastewater at reduced cost. *Wat. Sci. Tech.* 32. 9-10. pp.303-312.(1995)

Ozgur NH and Stenstrom MK. KBES for process control of nitrification in activated sludge process. *Journal of Environmental Engineering.* 120. 1. pp.87-107.(1994)

Önnerth TB, Nielsen MK, and Stamer C. Advanced computer control based on real and software sensors. *Wat. Sci. Tech.* 33. 1. pp.237-245.(1996)

Parunak HVD. Jennings NR, editors.O'Hare,G.M.P. New York: Wiley,J. 1996;Applications of distributed artificial intelligence in industry. p. 139-64.

Patry GG and Barnett MW. Innovative computing techniques for development of an integrated computer control system. *Wat. Sci. Tech.* 26. 5-6. pp.1365-1374.(1992)

Paul E, Plisson-Saune S, Mauret M, and Cantet J. Process state evaluation of alternating oxic-anoxic activated sludge using ORP, pH and DO. *Wat. Sci. Tech.* 38. 3. pp.299-306.(1998)

Pedersen, K.M.; Kümmel, M.; Soeberg, H. Monitoring and control of biological removal of phosphorus and nitrogen by flow-injection analysers in a municipal pilot-scale wastewater treatment plant. *Analytica Chimica Acta.* 238, pp 191-199. (1990)

Plisson-Saune S, Capdeville B, Mauret M, Deguin A, and Baptiste P. Real-time control of nitrogen removal using three ORP bendig-points: signification, control strategy and results. *Wat. Sci. Tech.* 33. 1. pp.275-280.(1996)

Poch, M.; Lafuente, J.; Sánchez, M.; Cortés, U.; Tomás, R. Los sistemas expertos aplicados a las estaciones depuradoras de aguas residuales. *Tecnología del Agua.* Agosto, pp 25-32.(1995)

Poch M, Lafuente J, Moreno R, and Iza J. Mejora de la calidad del agua residual en una planta de tratamiento de aguas residuales con un

analizador automático de demanda química de oxígeno (DQO). PROMA; 1997; V Congreso de Ingeniería Ambiental.

Poch M, Lafuente J, Sànchez M, Cortés U, and Tomàs R. Los sistemas expertos: Una nueva herramienta para ayudar en la gestión de las estaciones depuradoras de aguas residuales. *Tecnología del agua.*(1998)

Potter TG, Koopman B, and Svonoros SA. Optimization of a periodic biological process for nitrogen removal from wastewater. *Wat. Res.* 30. 1. pp.142-152.(1996)

Pullammanappallil PC, Svonoros SA, Chynoweth DP, and Lyberatos G. Expert system for control of anaerobic digesters. *Biotechnol. Bioeng.* 58. 1. pp.13-22.(1998)

Puñal AM. Estratexias para a posta en marcha, operación e control de dixestores anaerobios Departamento de Enxeñeria Química de la Universitat de Compostela; 1999;

R-Roda I. Desenvolupament d'un protocol per l'aplicació de sistemes basats en el coneixement a la gestió d'estacions depuradores d'aigües residuals urbanes Departament d'Enginyeria química Universitat de Girona; 1998; 11 p.

R-Roda I, Comas J, Sànchez M, Cortés U, Lafuente J, and Poch M. Specific knowledge base development to build an expert system for wastewater treatment plant management. Vancouver. 1998;

R-Roda I, Sànchez-Marré M, Cortés U, Lafuente J, and Poch M. Case based reasoning systems:asuitable tool for the supervision of clomplex processes. *Chemical Engineering Progress.* 95. 6. pp.39-45.(1999)

R-Roda Layret, I. Definició d'un sistema basat en el coneixement per el control de plantes depuradores d'aigües residuals urbanes amb criteris de Nitrificació / Desnitrificació. *Tesi de Llicenciatura.* Universitat Autònoma de Barcelona. Bellaterra.1994.

Randall CW, Barnard JL, and Stensel HD. Eckenfelder WW, Malina JF, and Patterson JW, editors.Design and retrofit of wastewater treatment plants for biological nutrient removal. Lancaster,Pennsylvania U.S.A: Technomic publishing company; 1992;

Rodrigo MA, Seco A, Penya-Roja JM, and Ferrer J. Influence of sludge age on enhanced phosphorus removal in biological systems. *Wat. Sci. Tech.* 34. 1-2. pp.41-48.(1996)

Roffel, B.; Chin, P. Computer Control in the Process Industries. Lewis Publishers Inc.1989.

Sànchez M. DAI-DEPUR: an Integrated Supervisory Multi-level Architechture for Wastewater Treatment Plants [Tesi Doctoral]. Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya; 1996; 1 p.

Sànchez M, Cortés U, Béjar J, de Gràcia J, Lafuente J, and Poch M. Concept formation in WWTP by means of classification techniques: A compared study. *Applied Intelligence.* 7. pp.147-165.(1997)

Sànchez M, Cortés U, Lafuente J, R-Roda I, and Poch M. DAI-DEPUR: an integrated and distributed architecture for wastewater treatment plants supervision. *Artificial Intelligence in Engineering.* 1. pp.275-285.(1996)

Sànchez M, Cortés U, R-Roda I, Lafuente J, and Poch M. DAI-DEPUR: A distributed architecture for wastewater treatment plants supervision. *Artificial Intelligence in Engineering.* 10. pp.275-285.(1996)

Sànchez M, Cortés U, R-Roda I, Poch M, and Lafuente J. Learning and adaptation in wastewater treatment plants through case-based-reasoning. *Microcomputers in Civil Engineering.* 12. pp.251-266.(1997)

Sànchez M, R-Roda I, Poch M, Cortés U, and Lafuente J. DAI-DEPUR architecture: Distributed agents for real-time WWTP supervision and control. IFAC; 1994; Valencia. 1994; 147 p. Artificial Intelligence in real time control.

Sánchez, M.; R-Roda Layret, I.; Lafuente, J.; Cortés, U.; Solà, C. Real Time Supervision of Watewater Treatment Plants: a Distributed Approach. 2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS (CSI-AI/KBS'94). Lund (Sweden). (1994)

Sànchez-Marré M, Cortés U, R-Roda I, and Poch M. Case Learning through a similarity measure for continuous domains. Brighton (UK). 1998; p 39

Sànchez-Marré M, Cortés U, R-Roda I, and Poch M. Sustainable case learning for continuous

domains. *Environmental modelling & software.* 14. 5. pp.349-357.(1999)

Schlegel S and Baumann P. Requeriments with respect to on-line analyzers for N and P. *Wat. Sci. Tech.* 33. 1. pp.139-146.(1996)

Seborg DE, Edgar TF, and Mellichamp DA.Process dynamics and control. New York: 1989;

Serra P, Lafuente J, Moreno R, de Prada C, and Poch M. Development of a real-time expert system for wastewater treatment plants control. *Control Eng. Practice.* 1. 2. pp.329-335.(1993)

Serra P, Sànchez M, Lafuente J, Cortés U, and Poch M. DEPUR: a knowledge based tool for wastewater treatment plants. *Engng. Applic. Artif. Intell.* 7. 1. pp.23-30.(1994)

Serra P, Sànchez M, Lafuente J, Cortés U, and Poch M. ISCWAP: A knowledge-based system for supervisig activated sludge processes. *Computers chem. Engng.* 21. 2. pp.211-221.(1997)

Shoham Y. Agent-oriented programming. *Artificial Intelligence* . 60. 1. pp.51-92.(1993)

Smolders GJF, Klop JM, Jacobs R, Van Loosdrecht MCM, and Heijnen JJ. Metabolic model of the biological phosphorus removal process:II. Validation during start-up conditions. *Biotechnol. Bioeng.* 48. 3. pp.234-245.(1995)

Smolders GJF, Klop JM, Van Loosdrecht MCM, and Heijnen JJ. A metabolic model of the biological phosphorus removal process: I. Effect of the sludge retention time. *Biotechnol. Bioeng.* 48. 3. pp.222-233.(1995)

Sorensen J. Optimization of a nutrient removing wastewater treatment plant using on-line monitors. *Wat. Sci. Tech.* 33. 1. pp.265-273.(1996)

Stephanopoulos, G. Chemical Process Control. An Introduction to Theory and Practice. Prentince Hall Inc. Englewood Cliffs, N.J. 1984.

Stevens, W.R. Unix networking programming. Prentince Hall Software Series. 1990.

Surmacz-Gorska J, Gernaey K, Demuynck C, Vanrolleghem P, and Verstraete W. Nitrification monitoring in activated sludge by oxygen uptake rate (OUR) measurements. *Wat. Res.* 30. 5. pp.1228-1236.(1996)

Szafnicki K, Bourgois J, Graillot D, Di Benedetto D, Breuil P, and Poyet JP. Real-time supervision of industrial waste-water treatment plants applied to the surface treatment industries. *Wat. Res.* 32. 8. pp.2480-2490.(1998)

Szafnicki K and Graillot D. Knowledge-based real-time fault detection and supervision of urban drainage systems. *Automatica.* 32. 7. pp.1043-1047.(1996)

Temmink H, Petersen B, Isaacs S, and Henze M. Recovery of biological phosphorus removal after periods of low organic loading. *Wat. Sci. Tech.* 34. 1-2. pp.1-8.(1996)

Tenno R and Uronen P. Stock and Concentration Dynamics of Activated Sludge Processes. *IFAC Computer Applications in Biotechnology.* pp.287-291.(1998)

Thompson L and Mertz G. Real-time expert system implementation at Monsanto-Krummrich. *Automatica.* 29. 5. pp.1177-1183.(1993)

Thomsen HA and Kisbye K. N and P on-line meters: requeriments, maintenance and stability. *Wat. Sci. Tech.* 33. 1. pp.147-157.(1996)

Tseng CC, Potter TG, and Koopman B. Effect of influent chemical oxygen demand to nitrogen ratio on a partial nitrification / complete denitrification process. *Wat. Res.* 32. 1. pp.165-173.(1998)

Van Vooren L, Willems P, Ottoy JP, Vansteenkiste GC, and Verstraete W. Automatic buffer capacity based sensor for effluent quality monitoring. *Wat. Sci. Tech.* 33. 1. pp.81-87.(1996)

Vanrolleghem P and Coen F. Optimal design of in-sensor-experiments for on-line modelling of nitrogen removal processes. *Wat. Sci. Tech.* 31. 2. pp.149-160.(1995)

Vanrolleghem P, Van der Schueren D, Krikilion G, Grijspeerdt K, Willems P, and Werstraete W. On-line quantification of settling properties with in-sensor-experiments in an automated settlometer. *Wat. Sci. Tech.* 33. 1. pp.37-51.(1996)

Verheijen L, Uchiakawa I, Kobayashi T, and Endo I. IMCS (Intelligent Monitoring and Control system) for wastewater-treatment. Budapest. 1997; 299 p.

Wacheux H, Million J-L, Guillo C, and Alves E. NH4 automatic analyser for wastewater

treatment plant: evaluation test at laboratory and field level. *Wat. Sci. Tech.* 33. 1. pp.193-201.(1996)

Wareham DG, Hall KJ, and Mavinic DS. Real-time control of wastewater treatment systems using ORP. *Wat. Sci. Tech.* 28. 11-12. pp.273-282.(1993)

Water Environment Federation & American Society of Civil Engineers. Desing of municipal wastewater treatment plants. Manual of practice No.8. ASCE Manual and Report on Engineering Practice No. 76. WEF / ASCE. Alexandria. New York. 1992.

WPCF (The Water Pollution Control Federation). Nutrient control. Manual of practice FD-7. WPCF. Washington.1983.

Wentzel MC and Ekama GA. Principles in the design of single sludge activated sludge systems for biological removal of carbon, nitrogen and phosphorus. Valencia,Venezuela. 1999; III Simposio internacional sobre polución de aguas por procesos biológicos.

Wentzel MC, Ekama GA, Dold PL, and Marais GVR. Biological excess phophorus removal-steady state process design. *Water SA.* 16. pp.29-48.(1990)

Wentzel MC, Lötter LH, Loewenthal RE, and Marais GVR. Metabolic behaviour of Acinetobacter spp. in enhanced biological phosphorus removal-a biochemical model. *Water SA.* 12. 4.(1986)

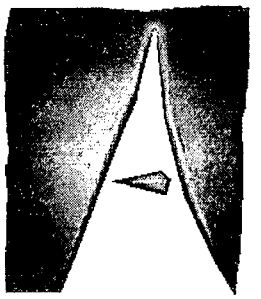
Wild D, von Schulthes R, and Gujer W. Structured modelling of denitrification intermediates. *Wat. Sci. Tech.* 31. 2. pp.45-54.(1995)

Yu RF, Liaw SL, Chang CN, Lu HJ, and Cheng WY. Monitoring and control using on-line ORP on the continuous-flow activated sludge batch reactor system. *Wat. Sci. Tech.* 35. 1. pp.57-66.(1997)

Zhao H, Isaacs S, Soeberg H, and Kümmel M. An analysis of nitrogen removal and control strategies in an alternating activated sludge process. *Wat. Res.* 29. 2. pp.535-544.(1995)

Zhao H, Mavinic DS, Oldham WK, and Koch FA. Factors affecting phosphorus removal in a two-stage intermittent aeration process treating domestic sewage. *Wat. Sci. Tech.* 38. 1. pp.115-122.(1998)

Zilio-Grandi F and Szpyrkowicz L. Seasonal phosphorus removal in a phostrip process-II. Phosphorus fractionation and sludge microbiology during start-up. *Wat. Res.* 29. 10. pp.2327-2338.(1995)



UNIVERSITAT AUTÒNOMA DE BARCELONA

DEPARTAMENT D'ENGINYERIA QUÍMICA
ESCOLA TÉCNICA SUPERIOR D'ENGINYERIES

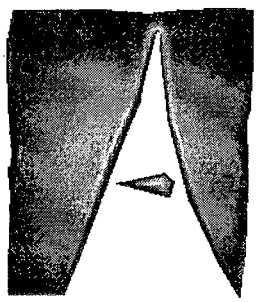
DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA SUPERVISOR PARA LA GESTIÓN Y CONTROL DE EDAR

APÉNDICES

Memoria que para optar al grado de doctor por la *Universitat Autònoma de Barcelona* presenta

Juan Antonio Baeza Labat

13 de Julio de 1999



UNIVERSITAT AUTÒNOMA DE BARCELONA

DEPARTAMENT D'ENGINYERIA QUÍMICA
ESCOLA TÉCNICA SUPERIOR D'ENGINYERIES

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA SUPERVISOR PARA LA GESTIÓN Y CONTROL DE EDAR

APÉNDICES

Memoria que para optar al grado de doctor por la *Universitat Autònoma de Barcelona* presenta

Juan Antonio Baeza Labat

13 de Julio de 1999

ÍNDICE

1	Programa de monitorización y control planta piloto	A-1
2	Programa de monitorización y control del sistema analítico	A-99
3	Programa del PLC	A-155
4	Servidor de datos	A-161
5	Comunicación GSI-Servidor de datos	A-191
6	Sistema experto	A-217

1 PROGRAMA MONITORIZACIÓN Y CONTROL PLANTA PILOTO

```
/*************************************************************************

   DEPURA.C - FUNCION PRINCIPAL DEL PROGRAMA DEPURA.EXE

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 11/12/98

*************************************************************************/

#include "depura.h"   /*Fichero con definiciones generales*/

/*************************************************************************/

void main(void)
 {

 lectura_configuracion();    /*Lee especificaciones de depura.dat*/
 lectura_alimentacion();     /*Lee el fichero de posible perfil de alimento*/

 inicializar_tiempos();      /*Inicializa los relojes*/
 modo_grafico();            /*Entra en modo gr fico*/
 graf_base();               /*Dibuja la pantalla gr fica*/
 graf_presentacion();       /*Copyright*/
 seguro_graficas(0);        /*Lectura ficheros de seguridad gr ficos*/

 inicializar_variables();
 inicio_filtro_digital();
 inicia_guardian();

 guarda_mensaje( "Inicio programa\t\t\x0");
 imprime("Inicio programa\v\v\n\x0");
 mensaje_a_SUN("Inicio programa\t\x0");

 graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
                     /*Actualiza los gr ficos de pantalla*/
 puesta_en_marcha();

/*
planta.variable_actual=3;
 graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
*/

do{
    contador_guardian=0;
    actualizar_tiempos();   /* Actualiza tiempos y detecta tareas */
    seleccionar_accion();   /* Selecciona la acci¢n a efectuar */

    selectmenu();          /* Tratamiento de peticiones de teclado */
    graf_reloj();      /*Grafica el radar de detecci¢n de sobrecargas*/
  }
while(fi==1);

 guarda_mensaje( "Fin programa\t\t\x0");
 imprime("Fin programa\v\v\n\x0");
 mensaje_a_SUN("Fin programa\t\x0");

 finaliza_guardian();

_setvideomode( _DEFAULTMODE );          /*Acaba con el modo gr fico, pasa a texto*/
}

/*************************************************************************/
/*Funci¢n para el tratamiento de peticiones de teclado*/

void selectmenu(void)
{
int tecla,i;
int debug=0;

if (planta.variable_actual==20) debug=1;

tecla=_bios_keybrd(_NKEYBRD_READY);  /*Comprueba si se ha presionado una tecla*/
if(tecla==0) return;

if (debug)     graf_mensaje("Tratamiento de peticiⁿ de teclado");

_bios_keybrd(_NKEYBRD_READ); /*Vac¡a el buffer, lee el £ltimo caracter*/
```



```
/*printf("%x\n",tecla);*/

switch(tecla){     /*Actuaci¢n en funci¢n de la tecla presionada*/
       case F1:    /*Cambio del menu de ayuda*/
            ayuda++;
            if (ayuda==1) graf_ayuda_general();
            else  {
                  graf_ayuda_graficas();
                  ayuda=0;
                  }
       break;
       case F2:    /*Variaci¢n manual de una salida de la placa PCL726*/
           graf_actuacion_externa_pcl726();
       break;
       case F3:    /*Enciende programa plc*/
           plc_on();
       break;
       case F4:    /*Apaga programa plc*/
           plc_off();
       break;
       case F5:    /*Enciende agitador/es*/
           graf_control_agitador(1);
       break;
       case F6:    /*Apaga agitador/es*/
           graf_control_agitador(0);
       break;
       case F7:    /*Parada de emergencia agitadores*/
           control_variador( 2 , 0 );
       break;
       case F8:    /*Variaci¢n velocidad agitadores*/
           graf_velocidad_agitadores();
       break;
       case F9:    /*Enciende/apaga salida PLC*/
           graf_actuacion_externa_salidas();
       break;
       case ALTQ:  /*Salida de programa parando todo*/
            pcl726( 0,4.0F);
            plc_off();
            seguro_graficas(1);
            fi=0;
       break;
       case F10:   /*Salida del programa dejando PLC encendido*/
          /*  pcl726( 0,4.0F);*/
           seguro_graficas(1);
           fi=0;
       break;
       case F11:   /*Reinicializar todas las variables*/
            inicializar_tiempos();
            reset_control_02(0);
            planta.conexion_SUN=1;
            planta.com_PLC=1;
            planta.control_nivel=1;
            planta.sondas=1;
            alarma(-1);
       break;
       case F12:
            inicializar_tiempos();
            alarma(5);
       break;
       case ALTC:  /*Desconexi¢n toma de medidas de las sondas*/
            if (planta.sondas==1)
              {
              planta.sondas=0;
              guarda_mensaje("Desconexi¢n sondas manual\t\x0");
              }
            else if (planta.sondas==0)
              {
              planta.sondas=1;
              guarda_mensaje("Conexi¢n sondas manual\t\x0");
              }
       break;
       case ALTD:  /*Salida al DOS*/
           dos=1;
           salir_al_dos();
           dos=0;
       break;
       case ALTP:
```

```
    if (debug) graf_mensaje("Toma de medidas pHrocon");
    for (i=20;i<23;i++)       estado_salida(i,1); /*para leer redox*/
    delay (2500);
    phrocon_lecturas();
    for (i=20;i<23;i++)       estado_salida(i,0); /*dejamos leyendo ox¡geno*/
    if (debug) graf_mensaje("Fin toma de medidas pHrocon");
    delay (1600);
break;
case ALTS:
   if (debug)  graf_mensaje("Supervisi¢n PLC");
       supervision_plc();
   if (debug)  graf_mensaje("Fin supervisi¢n PLC");
break;
case ESC:   /*Gr fica copyright*/
    graf_presentacion();
break;
case BACKSP:
    _clearscreen(_GCLEARSCREEN);
    graf_base();
break;
case o:     /*Gr fica ox¡geno*/
case O:
    planta.variable_actual= 3;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case h:     /*Gr fica pH*/
case H:
    planta.variable_actual= 0;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case r:     /*Gr fica redox*/
case R:
    planta.variable_actual= 2;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case t:     /*Gr fica temperatura*/
case T:
    planta.variable_actual= 1;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case a:     /*Gr fica mA*/
case A:
    planta.variable_actual= 4;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case e:     /*Cambio de escala*/
case E:
    planta.escala_actual++;
    if (planta.escala_actual>3) planta.escala_actual=1;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case NUM1:     /*Gr fica tanque 1*/
case NUM1E:
    planta.variable_actual= 5;          /*Tanque 1*/
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case NUM2:     /*Gr fica tanque 2*/
case NUM2E:
    planta.variable_actual= 6;          /*Tanque 2*/
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case NUM3:     /*Gr fica tanque 3*/
case NUM3E:
    planta.variable_actual= 7;          /*Tanque 3*/
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case d:
case D:
    planta.variable_actual= 20;
    graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
break;
case BARRA:    /*Actualiza el gr fico actual*/
    graf_valores_display();
    graf_actualizacion_grafico( planta.variable_actual, planta.escala_actual);
break;
case ALTO:
    if (planta.registro_O2==1)
```



```
            {
            if (planta.variable_actual==20)
                graf_mensaje("Registro de datos O2 inactivo");
            planta.registro_O2=0;
            break;
            }
        planta.registro_O2=1;
        if (planta.variable_actual==20)
            graf_mensaje("Registro de datos O2 activado");
    break;
    case ALTF1:
        if (planta.PID[0].activo==1)  our_estimacion(1,1);
        else  our_estimacion(1,0);
    break;
    case ALTF2:
        if (planta.PID[1].activo==1)  our_estimacion(2,1);
        else  our_estimacion(2,0);
    break;
    case ALTF3:
        if (planta.PID[2].activo==1)  our_estimacion(3,1);
        else  our_estimacion(3,0);
    break;
  }
return;
}

/*************************************************************************/
/*       Inicializacion de tiempos          */

void inicializar_tiempos(void)
{
#define ACTUALIZA(A) planta.A.tiempo=time(NULL)

t_actual=time(NULL);

ACTUALIZA(t_antic);
ACTUALIZA(t_inicio);
ACTUALIZA(t_phrocon);
ACTUALIZA(t_archivo);
ACTUALIZA(t_control_O2);
ACTUALIZA(t_seguro);
ACTUALIZA(t_grafica);
ACTUALIZA(t_actualizacion_matrices);
ACTUALIZA(t_reloj);
ACTUALIZA(t_SUN);
ACTUALIZA(t_pantalla);
ACTUALIZA(t_perfil);
ACTUALIZA(t_OUR);

return;
}


/*************************************************************************/
/* Calcula el d¡a actual respecto el d¡a de referencia, 132.
   El d¡a 1 es el viernes 17 de marzo de 1995                          */

float dia_absoluto(void)
{
float dias=0;
time_t t_referencia=3015817200L;
float  dia_referencia=132.0F;

dias=(float) (dia_referencia+difftime(t_actual,t_referencia)/3600.0F/24.0F);

return dias;
}

/*************************************************************************/

int puesta_en_marcha(void)
{
int debug=0, status;
float mA=0.0F;
char mensaje[100];

if (planta.variable_actual==20) debug=1;
```

```
if (debug)  graf_mensaje("Inicializando PLC");
  status=inicializar_plc();                 /*Inicializa PLC*/
if (debug)
    if (status==-1)  graf_mensaje("Error inicializando PLC");
    else  graf_mensaje("Inicializaci¢n PLC OK");

if (debug)  graf_mensaje("Inicializando Danfoss");
 inicializar_variador_frecuencia();  /*Inicializa los danfoss 2100*/
if (debug)  graf_mensaje("Inicializaci¢n Danfoss OK");

if (debug)  graf_mensaje("Variando velocidad agitadores");
 for (i=0;i<2;i++)
 {
 delay(200);
 variar_velocidad (planta.velocidad[0], 1);
 delay(200);
 variar_velocidad (planta.velocidad[1], 2);
 delay(200);
 variar_velocidad (planta.velocidad[2], 3);
 delay(300);
 control_variador( 1 , 0 );
 }       /*Env¡a la velocidad de los tres agitadores dos veces*/

if (debug)  graf_mensaje("Inicio activaci¢n PLC");
  plc_on();    /*Enciende las salidas del plc e inicia el programa del PLC*/
if (debug)  graf_mensaje("Programa PLC activado");

if (debug)  graf_mensaje("Inicio abertura v lvula agua");
status = mA_caudal( "c:\\depura\\v_agua.dat", planta.caudal_agua, &mA);
if (status==0)
    {
    pcl726(0,mA);     /*Abre la v lvula de entrada de agua*/
    sprintf(mensaje, " V lvula de agua abierta: Q = %4.1f ml/min, %4.2f mA",
            planta.caudal_agua,mA);
    if (debug) graf_mensaje(mensaje);
    }
else if (debug)  graf_mensaje("V lvula de agua no abierta");

if (debug)  graf_mensaje("Regulaci¢n caudal dosificadora 1 (RE)");
status = mA_caudal( "c:\\depura\\bomba1.dat", planta.caudal_RE, &mA);
if (status==0)
    {
    pcl726(1,mA);     /*Regula el caudal*/
    sprintf(mensaje, " Caudal recirculaci¢n externa: Q = %4.1f ml/min, %4.2f mA",
            planta.caudal_RE,mA);
    if (debug) graf_mensaje(mensaje);
    }
else if (debug)  graf_mensaje("Recirculaci¢n externa no regulada");

if (debug)  graf_mensaje("Regulaci¢n caudal dosificadora 2 (RI)");
status = mA_caudal( "c:\\depura\\bomba2.dat", planta.caudal_RI, &mA);
if (status==0)
    {
    pcl726(2,mA);     /*Regula el caudal*/
    sprintf(mensaje, " Caudal recirculaci¢n interna: Q = %4.1f ml/min, %4.2f mA",
            planta.caudal_RI,mA);
    if (debug) graf_mensaje(mensaje);
    }
else if (debug)  graf_mensaje("Recirculaci¢n interna no regulada");

if (debug)     graf_mensaje("Supervisi¢n PLC");
   supervision_plc();
if (debug)     graf_mensaje("Fin supervisi¢n PLC");

if (planta.perfil==1)
    {
    if (debug) graf_mensaje("Variando alimentaci¢n programada");
      cambiar_alimentacion();
    if (debug) graf_mensaje("Fin variaci¢n alimentaci¢n");
    }

for (i=0;i<3;i++)   /* Actuaci¢n fija de aireaci¢n */
   if (planta.PID[i].activo==2)
     {
     medidas[i].mA=planta.mA[i+3];   /* Guarda la actuaci¢n */
     pcl726 (i+3, planta.mA[i+3]);   /* Actuaci¢n */
```



```
     if (debug)
      {
      sprintf(mensaje,"PID %i desactivado, actuaci¢n constante de %4.2f mA",
        i+1, planta.mA[i+3]);
      graf_mensaje(mensaje);
      }
     }

if (debug)     graf_mensaje("Inicializando sondas");
  status=inicializar_sondas();                    /*Inicializa los pHrocon18*/
if (debug)
    if (status!=3)  graf_mensaje("Error inicializando sondas");
    else  graf_mensaje("Inicializaci¢n sondas OK");

delay(1000);

return 1;
}

/*************************************************************************/

void actualizar_tiempos(void)
{
#define RELOJ(A,B) if((int)difftime(t_actual,planta.A.tiempo)>=B) planta.A.fin=1;
struct tm *temps;

t_actual=time(NULL);      /*Actualizaci¢n del tiempo */

RELOJ(t_reloj,1);
RELOJ(t_control_O2,planta.intervalo_control_O2);
RELOJ(t_actualizacion_matrices,10);
RELOJ(t_antic,planta.intervalo_control_nivel);
RELOJ(t_phrocon,30);
RELOJ(t_archivo,planta.intervalo_toma_de_datos);
RELOJ(t_grafica,planta.actualizacion_grafica);
RELOJ(t_seguro,600);
RELOJ(t_SUN,planta.intervalo_SUN);
RELOJ(t_pantalla,600);
RELOJ(t_OUR,planta.intervalo_OUR);

planta.t_actual=time(NULL);
temps = localtime( &planta.t_actual );
/*Si el minuto es 0 y han pasado m s de 100 segundos desde el £ltimo cambio*/
if ( planta.perfil==1 &&
     temps->tm_min == 0 &&
     (int)difftime(t_actual,planta.t_perfil.tiempo)>100)
    planta.t_perfil.fin=1;

}

/*************************************************************************/

int seleccionar_accion(void)
{
#define ACTUALIZAR(A) planta.A.tiempo=time(NULL);  planta.A.fin=0;
int debug=0, status;

if (planta.variable_actual==20) debug=1;

if ( planta.t_control_O2.fin==1 &&
     planta.sondas==1)
      {
      ACTUALIZAR(t_control_O2);
       if (debug)  graf_mensaje("Actuaci¢n control ox¡geno");
      control_oxigeno();
       if (debug)  graf_mensaje("Fin actuaci¢n control ox¡geno");
      return 1;
      }    /*Activa el control de ox¡geno*/

if ( planta.t_phrocon.fin==1 &&
     planta.sondas==1)
      {
      ACTUALIZAR(t_phrocon);
       if (debug)  graf_mensaje("Toma de medidas pHrocon");
      for (i=20;i<23;i++)  estado_salida(i,1);       /*para leer redox*/
      while(difftime(time(NULL),planta.t_phrocon.tiempo)<=3) selectmenu();
      phrocon_mv();
```

```
      for (i=20;i<23;i++)  estado_salida(i,0);       /*dejamos leyendo ox¡geno*/
      phrocon_T();
      phrocon_ph();
      /*phrocon_lecturas();*/
       if (debug)  graf_mensaje("Fin toma de medidas pHrocon");
      while(difftime(time(NULL),planta.t_phrocon.tiempo)<=8) selectmenu();
      return 2;
      }   /*Activa la lectura de valores de los pHrocon18*/

if ( planta.t_antic.fin==1 &&
     planta.control_nivel==1)
      {
      ACTUALIZAR(t_antic);
       if (debug)  graf_mensaje("Supervisi¢n PLC");
      supervision_plc();
       if (debug)  graf_mensaje("Fin supervisi¢n PLC");
      return 3;
      }    /*Si la planta est  funcionado, activa la supervisi¢n del PLC*/

if ( planta.t_actualizacion_matrices.fin==1)
      {
      planta.t_actualizacion_matrices.tiempo+=10;
      planta.t_actualizacion_matrices.fin=0;
       if (debug)  graf_mensaje("Actualizaci¢n display y matrices");
      actualizar_matrices_graficas();
      graf_valores_display();
      return 4;
      }    /*Actualiza las matrices gr ficas y display de valores*/

if ( planta.t_SUN.fin==1 &&
     planta.sondas==1 &&
     planta.conexion_SUN==1 )
      {
      ACTUALIZAR(t_SUN);
       if (debug)  graf_mensaje("Inicio comunicaci¢n SUN");
      status=sockets_pasa_datos();
       if (debug)
       if (status!=0) graf_mensaje("Comunicaci¢n SUN incorrecta");
          else        graf_mensaje("Comunicaci¢n SUN correcta");
      return 5;
      }   /* Transmite datos a la SUN */

if ( planta.t_archivo.fin==1)
      {
      ACTUALIZAR(t_archivo);
       if (debug)  graf_mensaje("Archivando datos medida");
      archivar_datos_medida();
       if (debug)  graf_mensaje("Fin archivo datos medida");
      return 6;
      }   /*Almacena los valores en un fichero*/

if ( planta.t_seguro.fin==1)
      {
      ACTUALIZAR(t_seguro);
       if (debug)  graf_mensaje("Archivando datos de seguridad");
      seguro_graficas(1);
       if (debug)  graf_mensaje("Fin archivo datos de seguridad");
      return 7;
      }     /*Guarda fichero de seguridad con los valores de las gr ficas*/

if ( planta.t_grafica.fin==1)
      {
      ACTUALIZAR(t_grafica);
      graf_actualizacion_grafico( planta.variable_actual, planta.escala_actual);
      return 8;
      }     /*Actualiza la gr fica en pantalla*/

if ( planta.t_reloj.fin==1 )
      {
      ACTUALIZAR(t_reloj);
      graf_hora();
      return 9;
      }    /*Actualiza reloj gr fico*/

if ( planta.t_pantalla.fin==1 )
      {
      ACTUALIZAR(t_pantalla);
```



```
      _clearscreen(_GCLEARSCREEN);
      graf_base();
      return 10;
      }    /*Actualiza toda la pantalla*/

if ( planta.perfil==1 &&
     planta.t_perfil.fin==1 &&
     planta.control_nivel==1)
      {
      ACTUALIZAR(t_perfil);
      if (debug)  graf_mensaje("Variando alimentaci¢n programada");
      cambiar_alimentacion();
      if (debug)  graf_mensaje("Fin variaci¢n alimentaci¢n");
      return 11;
      }

if ( planta.conexion_SUN == 0 &&
     difftime(planta.t_actual,planta.t_SUN.tiempo) > 3600 )
     planta.conexion_SUN = 1;

if ( planta.t_OUR.fin==1 &&
     planta.sondas==1 &&
     planta.OUR == 1 )
      {
      ACTUALIZAR(t_OUR);
       if (debug)  graf_mensaje("Inicio estimaci¢n consumo ox¡geno");
          if (planta.PID[0].activo==1 &&
              planta.PID[0].set_point!=0.0F)  our_estimacion(1,1);
          if (planta.PID[1].activo==1 &&
              planta.PID[1].set_point!=0.0F)  our_estimacion(2,1);
          if (planta.PID[2].activo==1 &&
              planta.PID[2].set_point!=0.0F)  our_estimacion(3,1);
         delay(3000);
       if (debug)  graf_mensaje("Fin inicio estimaci¢n consumo ox¡geno");
      return 12;
      }    /*Activa la estimaci¢n del consumo de ox¡geno*/

return 0;
}

/*************************************************************************/

void inicializar_variables(void)
{
/*
int i;
for (i=0;i<3;i++) planta.PID[i].activo=1;*/

sprintf (medidas[0].nom, "%s","REACTOR 1");
sprintf (medidas[1].nom, "%s","REACTOR 2");
sprintf (medidas[2].nom, "%s","REACTOR 3");
medidas[0].controlador=1;
medidas[1].controlador=2;
medidas[2].controlador=3;
planta.variable_actual=20; /* Gr fico de debug */
planta.escala_actual=1;
planta.control_nivel=1;
planta.sondas=1;
planta.conexion_SUN=1;
planta.com_PLC=1;
planta.intervalo_SUN=30;
planta.registro_O2=0;
planta.alarma=-3;
old_intervalo_o2=planta.intervalo_control_O2;

}


/*************************************************************************/
```

```
/*************************************************************************

   DEPURA.H - DEFINICION DE VARIABLES GENERALES PARA EL PROGRAMA DEPURA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 30/4/97

*************************************************************************/

/******************** Librerias  a incluir ******************************/

#include <stdio.h>
#include <dos.h>
#include <bios.h>
#include <time.h>
#include <process.h>
#include <conio.h>
#include <ctype.h>
#include <stdlib.h>
#include <graph.h>
#include "tecias.h"
#include "nombres.h"

/******************** Subrutinas externas necesarias *****************/

/* danfoss */
extern void inicializar_variador_frecuencia(void);
extern int variar_velocidad(double velocidad, int variador);
extern int control_variador(int estado, int variador);
extern void graf_control_agitador(int on_off);
extern void graf_velocidad_agitadores(void);

/* pcl726 */
extern int pcl726 (int canal, float miliamperios);
extern void graf_actuacion_externa_pcl726(void);

/* plc */
extern void plc_on(void);
extern void plc_off(void);
extern void estado_salida(unsigned int numero, unsigned int off_on);
extern void graf_actuacion_externa_salidas(void);
extern void supervision_plc(void);
extern void alarma(int motivo);
extern int inicializar_plc(void);
extern int cambiar_alimentacion(void);

/* crison */
extern int inicializar_sondas(void);
extern void phrocon_lecturas(void);
extern void phrocon_ph(void);
extern void phrocon_T(void);
extern void phrocon_mv(void);

/* cont_oxi */
extern void control_oxigeno(void);
extern void reset_control_O2(int reactor);
extern void inicio_filtro_digital(void);
extern void our_estimacion(int reactor, int step);

/* delay */
extern void delay( clock_t wait );

/* graficos */
extern void modo_grafico(void);
extern void graf_colorines(int orden);
extern void graf_base(void);
extern void graf_ayuda_general(void);
extern void graf_ayuda_graficas(void);
extern void graf_reloj(void);
extern void salir_al_dos(void);
extern void graf_valores_display(void);
extern void actualizar_matrices_graficas(void);
extern void graf_hora(void);
extern void graf_actualizacion_grafico(int variable, int escala);
extern void graf_nuevo_grafico( int variable, int escala);
extern void graf_presentacion(void);
extern void graf_mensaje(char *mensaje);
```



```
/* ficheros */
extern void lectura_configuracion(void);
extern void lectura_alimentacion(void);
extern void archivar_datos_medida(void);
extern void fichero_semanal(void);
extern void guarda_mensaje( char *mensaje);
extern void seguro_graficas(int orden);

/* cliente */

extern int sockets_pasa_datos(void);
extern int mensaje_a_SUN(char *mensaje);

/* printer */

extern void imprime(char *mensaje);

/* calibra */
extern int mA_caudal( char *fichero, float caudal, float *mA);

/* guardian */
extern void inicia_guardian(void);
extern void finaliza_guardian(void);

/* depura */
void selectmenu(void);
void inicializar_tiempos(void);
float dia_absoluto(void);
int puesta_en_marcha(void);
void actualizar_tiempos(void);
int seleccionar_accion(void);
void inicializar_variables(void);

/**********************Estructuras y variables ************************/


/* depura.c */

time_t t_actual;

int fi=1;   /*variable que controla el final del programa*/
int i=0,j=0;
int ayuda=1;

int contador_guardian=0;
int dos=0;
int old_intervalo_o2;

estado_planta planta;
primera medidas[3];
```

```
/*************************************************************************

   NOMBRES.H - DEFINICION DE ESTRUCTURAS COMUNES PARA EL PROGRAMA DEPURA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 21/9/98

*************************************************************************/

typedef struct {        /*Par metros del controlador*/
       int activo;      /* Para activar o desactivar el control */
       float set_point;  /*Set point*/
       float Kc;         /*Ganancia*/
       float Ti;         /*Constante de tiempo integral*/
       float Td;        /*Constante de tiempo derivado*/
}controlador;

typedef struct{
       time_t        tiempo;
       unsigned char  fin;
}temporal;

typedef struct {        /*Par metros del perfil de alimentaci¢n*/
       int   carbono;   /* Intervalo dosificaci¢n carbono */
       int   nitrogeno;  /* Intervalo dosificaci¢n nitr¢geno */
       float agua;      /* Caudal de agua */
}alimentacion;

typedef struct{
       int          sondas;    /* Para activar o desactivar la lectura */
       int          variadores;
       int          control_nivel;
       int          conexion_SUN;
       int          com_PLC;
       int          alarma;
       time_t       t_actual;
       temporal     t_reloj;
       temporal     t_antic;
       temporal     t_archivo;
       temporal     t_inicio;
       temporal     t_control_O2;
       temporal     t_actualizacion_matrices;
       temporal     t_phrocon;
       temporal     t_intervalo_toma_de_datos;
       temporal     t_grafica;
       temporal     t_seguro;
       temporal     t_SUN;
       temporal     t_pantalla;
       temporal     t_perfil;
       temporal     t_OUR;
       controlador  PID[3];
       float        velocidad[3];
       float        caudal_agua;
       float        caudal_RE;
       float        caudal_RI;
       int          entradas_plc[16];/*Guarda estado de las entradas*/
       int          salidas_plc[24]; /*Guarda estado de las salidas*/
       float        mA[6];           /*Guarda miliamperios PCL726*/
       int          variable_actual;
       int          escala_actual;
       int          intervalo_control_O2;
       int          intervalo_dosificacion[2];
       int          intervalo_limp_dosif[2];
       int          pistonadas_limp_dosif[2];
       int          intervalo_control_nivel;
       int          intervalo_toma_de_datos;
       int          tiempo_purga;
       int          duracion_purga;
       int          actualizacion_grafica;
       int          intervalo_SUN;
       int          registro_O2;
       int          intervalo_flojet_on;
       int          intervalo_flojet_off;
       int          perfil;
       int          intervalo_OUR;
       int          OUR;
       alimentacion alimento[24];
```



```
}estado_planta;

typedef struct {                        /*Estructura  per  parametres  referents  als
controladors*/
       char nom[10];                   /*Nom del controlador*/
       int controlador;                /*Numero del controlador */
       float ph;                       /*Valor mesurat de la variable*/
       float temperatura;              /*Valor mesurat de la variable*/
       float potencial;                /*Valor mesurat de la variable*/
       float oxigeno;                  /*Valor mesurat de la variable*/
       float mA;                       /*Valor mesurat de la variable*/
       float velocidad;                /*Valor mesurat de la variable*/
}primera;
```

```
/*************************************************************************

   TECLAS.H - DEFINICION DE CODIGOS DE TECLADO

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 14/5/98

*************************************************************************/

#define ESC       0x011B
#define CTRLC     0x2E03
#define CTRLQ     0x1011
#define CTRLBS   0x0E7F
#define CTRLRET   0x1C0A
#define CTRLO    0x180F
#define ALTQ     0x1000
#define ALTP     0x1900
#define ALTS     0x1F00
#define ALTD     0x2000
#define ALTF     0x2100
#define ALTG     0x2200
#define ALTC     0x2E00
#define ALTN     0x3100
#define ALTM     0x3200
#define ALTO     0x1800

#define RETURN    0x1C0D
#define INTRO     0xE00D
#define BACKSP    0x0E08
#define TAB       0x0F09
#define BARRA     0x3920

#define AMUNT     0x48E0
#define ABAIX     0x50E0
#define DRETA     0x4DE0
#define ESQUERRA 0x4bE0

#define INSERT    0x52E0
#define SUPR      0x53E0
#define INICIO    0x47E0
#define FIN       0x4FE0
#define REPAG     0x51E0
#define AVPAG     0x49E0

#define F1        0x3B00
#define F2        0x3C00
#define F3        0x3D00
#define F4        0x3E00
#define F5        0x3F00
#define F6        0x4000
#define F7        0x4100
#define F8        0x4200
#define F9        0x4300
#define F10       0x4400
#define F11       0x8500
#define F12       0x8600
#define ALTF1    0x6800
#define ALTF2     0x6900
#define ALTF3     0x6A00
#define ALTF4     0x6B00
#define ALTF5     0x6C00
#define ALTF6     0x6D00
#define ALTF7     0x6E00
#define ALTF8     0x6F00
#define ALTF9     0x7000
#define ALTF10    0x7100

#define o         0x186F
#define O         0x184F
#define p         0x1970
#define P         0x1950
#define m         0x326D
#define M         0x324D
#define t         0x1474
#define T         0x1454
#define e         0x1265
#define E         0x1245
```



```
#define a         0x1E61
#define A         0x1E41
#define r         0x1372
#define R         0x1352
#define h        0x2368
#define H         0x2348
#define d        0x2064
#define D        0x2044

#define NUM1      0x0231
#define NUM2      0x0332
#define NUM3      0x0433
#define NUM1E     0x4F00
#define NUM2E     0x5000
#define NUM3E     0x5100
```

```
/*****************************************************************************

   CALCULO.C - FUNCIONES PARA REALIZAR CALCULOS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 27/10/98

*****************************************************************************/

#include <stdio.h>

/*#define TEST*/

int regresion_lineal( float *x, float *y, int n,
                      float *a, float *b, float *r);

/*****************************************************************************/
/* Calcula la regresi¢n lineal con los datos de x e y.
   Devuelve ordenada (a), pendiente (b) y coeficiente correlaci¢n (r)  */

int regresion_lineal( float *x, float *y, int n,
                      float *a, float *b, float *r)
{
float sumx=0.0F,
      sumy=0.0F,
      sumxy=0.0F,
      sumx2=0.0F,
      sumy2=0.0F;
float parcial;
int i;

if (n<2) return -1;

for (i=0;i<n;i++)
    {
    sumx=sumx+x[i];
    sumy=sumy+y[i];
    sumxy=sumxy+x[i]*y[i];
    sumx2=sumx2+x[i]*x[i];
    sumy2=sumy2+y[i]*y[i];
    }


parcial=(n*sumx2-sumx*sumx);
if (parcial !=0)  *b=(n*sumxy-sumx*sumy)/parcial;
else
    {
    *b=0.0F;
    *a=0.0F;
    *r=0.0F;
    return -1;
    }

*a=(sumy-(*b)*sumx)/n;

parcial=((n*sumx2-sumx*sumx)*(n*sumy2-sumy*sumy));
if (parcial !=0) *r=(n*sumxy-sumx*sumy)*(n*sumxy-sumx*sumy)/parcial;
else *r=0.0F;

return 1;
}

/*****************************************************************************/
#ifdef TEST
void main(void)
{
float misx[5]={1.0F,2.0F,3.0F,4.0F,5.0F};
float misy[5]={2.0F,4.0F,6.0F,8.0F,10.5F};
float a,b,r;

regresion_lineal( misx, misy, 5, &a, &b, &r);

printf("\ta: %f\tb: %f\tr: %f\n",a,b,r);

}
#endif
/*****************************************************************************/
```



```
/*****************************************************************************

   CALIBRA.C - FUNCIONES PARA CALIBRADOS DOSIFICADORAS Y AGUA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 30/5/97

*****************************************************************************/

/******************** Librerias  a incluir **************************/

#include <stdio.h>
#include <io.h>
#include <time.h>
#include <stdlib.h>
#include <memory.h>

#include "nombres.h"
/*#define TEST*/
/********************** Subrutinas del fichero **********************/

int  mA_caudal( char *fichero, float caudal, float *mA);

/********************** Subrutinas externas *************************/

/****************** Variables comunes del programa ******************/

extern estado_planta planta;

/*****************************************************************************/
/*****************************************************************************/

int  mA_caudal( char *fichero, float caudal, float *mA)
{
FILE *obert;
char linea1[150];
int numero=0,
    punto=0;
int i;
float cal_caudal[25],
      cal_mA[25];
float x1,x2,y1,y2,x;

if ( access(fichero,0) != 0) return -1;

obert=fopen(fichero,"rt");
memset(linea1,0,sizeof(linea1));

fgets(linea1,sizeof(linea1),obert);    /* Lee 1 l¡nea del fichero */
sscanf(linea1,"%d",&numero);

for (i=0;i<numero;i++)
   {
   fgets(linea1,sizeof(linea1),obert);
   sscanf(linea1,"%f%f", &cal_mA[i], &cal_caudal[i]);
   }
fclose(obert);

for (i=0;i<numero;i++)
    {
    if (cal_caudal[i]>caudal)
       {
       punto=i;
       break;
       }
    }

if (punto==0) return -2;

x1=cal_caudal[punto-1];
x2=cal_caudal[punto];
y1=cal_mA[punto-1];
y2=cal_mA[punto];
x=caudal;

*mA=y1+(y2-y1)/(x2-x1)*(x-x1);
```

```
return 0;
}

/*****************************************************************************/

#ifdef TEST
void main(void)
{
float a=400.0F,b=0.0F;
int retorno;
int i;

for(i=0;i<10;i++)
{
   a=100.0F*i;
retorno= mA_caudal( "c:\\depura\\bomba1.dat", a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);
}
for(i=0;i<10;i++)
{
   a=200.0F*i;
retorno= mA_caudal( "c:\\depura\\bomba2.dat", a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);
}
for(i=0;i<10;i++)
{
   a=50.0F*i;
retorno= mA_caudal( "c:\\depura\\v_agua.dat", a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);
}

}
#endif
/*****************************************************************************/
```



```
/*****************************************************************************

   CLIENTE.C - FUNCIONES PARA LA COMUNICACION CON LA SUN

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 24/10/98

*****************************************************************************/

/******************** Librerias  a incluir ***************************/

#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <process.h>
#include <stdlib.h>
#include <io.h>
#include <conio.h>
#include <time.h>
#include "nombres.h"

#include "c:\sockets\devkit\include\pctcp\types.h"
#include "c:\sockets\devkit\include\sys\types.h"
#include "c:\sockets\devkit\include\sys\ioctl.h"
#include "c:\sockets\devkit\include\pctcp\errno.h"
#include "c:\sockets\devkit\include\sys\socket.h"
#include "c:\sockets\devkit\include\netinet\in.h"
#include "c:\sockets\devkit\include\netdb.h"
/*#include "c:\sockets\devkit\include\arpa\inet.h"*/

#include "c:\sockets\devkit\include\pctcp\pctcp.h"
#include "c:\sockets\devkit\include\pctcp\ipconfig.h"
#include "c:\sockets\devkit\include\pctcp\sockets.h"
#include "c:\sockets\devkit\include\pctcp\options.h"
#include "c:\sockets\devkit\include\fcntl.h"

#include <sys\time.h>
/*********************** Definiciones ********************************/

/*#define DEPURA
#define TEST
#define extern*/

/*#define svrName "eq3.uab.es"  */      /* Direcci¢n SUN */
#define svrName "158.109.13.2"          /* Direcci¢n num,rica SUN */
#define Port (ushort) 32323             /* Numero de puerto del socket */
#define TAMANY 512                      /* Tama¤o del buffer */
#define TAMANY_PETIT 32
#define TAMANY_M   100              /* Tama¤o del mensaje cliente */

typedef unsigned short ushort;          /* Definicion de tipo de variables */
typedef unsigned long ulong;

/*********************** Subrutinas del fichero **********************/

int sockets_pasa_datos(void);
int sockets_lee_datos_SUN(void);
void medidas_al_buffer(void);
int cambio_consignas(void);
int nuevos_datos_SUN(void);
int mensaje_a_SUN(char *mensaje);
void medida_a_SUN_varios(char variable, float valor);
int sockets_pasa_datos_petit(void);
int connectToServer(char *serverName, ushort port);
int disconnectToServer(int toServerSocket);

void error_send(int errno);
void error_socket(int errno);
void error_connect(int errno);

#ifdef TEST
void main(void);
#define extern
#endif

/******************** Subrutinas externas necesarias *****************/
```

```
extern ulong inet_addr(char *servidor); /**JB**/
extern int close(int elsocket);         /**JB**/
extern int variar_velocidad(double velocidad, int variador);
extern void guarda_mensaje( char *mensaje);
extern void imprime(char *mensaje);
extern int mA_caudal( char *fichero, float caudal, float *mA);
extern int pcl726 (int canal, float miliamperios);
extern int plc_timer(int valor, int numero);
extern void alarma(int motivo);
extern void graf_mensaje(char *mensaje);

/****************** Variables comunes del programa ******************/

char buffer[TAMANY];              /* Variable a transmitir o recibir */
char buffer_p[TAMANY_PETIT];

extern estado_planta planta;
extern primera medidas[3];


/*********************************************************************/
/* Funci¢n para establecer la comunicaci¢n y controlar el env¡o y la
   recepci¢n de datos */


int sockets_pasa_datos(void)
{
int toServerSocket = -1;
int i;                            /* Variables de control */
char caracter[1]={'S'};

/* Si la conexi¢n anterior no ha sido correcta, vuelve */
if (planta.conexion_SUN == 0) return -2;

/* Conexi¢n con la SUN */
toServerSocket = connectToServer(svrName, Port);

/* Si la conexi¢n no es correcta, devuelve -1 */
if (toServerSocket==-1)
    {
#ifdef DEPURA
    printf ("Error en la conexi¢n\n");
#endif
    planta.conexion_SUN=0;
    guarda_mensaje( "La conexi¢n con la SUN no es correcta\t\t\x0");
    imprime( "La conexi¢n con la SUN no es correcta\v\v\n\x0");
    return -1;
    }

#ifdef DEPURA
printf("CONEXIO A %s REALITZADA, %i\n",svrName,toServerSocket);
#endif

/* Pedimos nuevas consignas */
caracter[0]='C';
send(toServerSocket, caracter, sizeof(caracter), 0);

recv(toServerSocket, caracter, sizeof(caracter), 0);
if (caracter[0]=='S')
    {
/*    printf("Tenemos consignas\n");*/
    recv(toServerSocket, buffer, sizeof(buffer), 0);

    if (cambio_consignas()==1)
       {
       caracter[0]='O';
       send(toServerSocket, caracter, sizeof(caracter), 0);
       /*printf("Consignas correctas\n");*/
       }
    else
       {
       caracter[0]='N';
       send(toServerSocket, caracter, sizeof(caracter), 0);
       /*printf("Las consignas no son correctas\n");*/
       }
    }
```



```
/* Introducimos los datos en el buffer */
medidas_al_buffer();

caracter[0]='S';

/* Mandamos una S, queremos establecer los valores, set */
send(toServerSocket, caracter, sizeof(caracter), 0);

/* Mandamos el buffer */
i = send(toServerSocket, buffer, sizeof(buffer), 0);
#ifdef DEPURA
if (i==-1) error_send(errno);
#endif

/* Recibimos respuesta */
recv(toServerSocket, caracter, sizeof(caracter), 0);
/* Lo env¡a de nuevo si no ha sido bien recibido */
if (caracter[0]!='O')
    {
#ifdef DEPURA
    printf("Recibimos mal\n");
#endif
    caracter[0]='S';
    send(toServerSocket, caracter, sizeof(caracter), 0);

    send(toServerSocket, buffer, sizeof(buffer), 0);
    recv(toServerSocket, caracter, sizeof(caracter), 0);
    if (caracter[0]!='O')       /* Si lo recibe mal de nuevo, desconecta */
       {
#ifdef DEPURA
printf("Recibimos mal de nuevo\n");
#endif
       disconnectToServer(toServerSocket);
       return -3;
       }
    }


/* Pedimos desconexi¢n */
caracter[0]='F';
send(toServerSocket, caracter, sizeof(caracter), 0);
disconnectToServer(toServerSocket);

return 0;
}

/*********************************************************************/
/* Funci¢n para pedir las lectura de la SUN */

int sockets_lee_datos_SUN(void)
{
int toServerSocket = -1;
int i;                          /* Variables de control */
char caracter[1]={'R'};


/* Conexi¢n con la SUN */
toServerSocket = connectToServer(svrName, Port);

/* Si la conexi¢n no es correcta, devuelve -1 */
if (toServerSocket==-1)
    {
#ifdef DEPURA
    printf ("Error en la conexi¢n\n");
#endif
    planta.conexion_SUN=0;
    return -1;
    }

#ifdef DEPURA
printf("CONEXIO A %s REALITZADA, %i\n",svrName,toServerSocket);
#endif

/* Mandamos una R, queremos leer los valores */
send(toServerSocket, caracter, sizeof(caracter), 0);
```

```
/* Recibimos el buffer */
recv(toServerSocket, buffer, sizeof(buffer), 0);

if (nuevos_datos_SUN()==1)
    {
#ifdef DEPURA
    printf("Datos enviados por la SUN:\n");
#endif
    for (i=0;i<3;i++)
        {
        printf( "%s\n", medidas[i].nom );
        printf( "pH: %7.4f\n", medidas[i].ph );
        printf( "T:  %7.4f\n", medidas[i].temperatura );
        printf( "mV: %7.4f\n", medidas[i].potencial );
        printf( "O2: %7.4f\n", medidas[i].oxigeno );
        printf( "mA: %7.4f\n", medidas[i].mA );
        printf( "V:  %7.4f\n", medidas[i].velocidad );
        }
    }
else
    printf("Los datos enviados por la SUN no son correctos\n");

/* Pedimos desconexi¢n */
caracter[0]='F';
send(toServerSocket, caracter, sizeof(caracter), 0);
disconnectToServer(toServerSocket);

return 0;
}

/*************************************************************************/
/* Introduce los datos de la estructura medidas en el buffer,
   a¤adiendo un cheksum   */

void medidas_al_buffer(void)
{
struct segunda{
        char    nom[3][10],
                controlador[3][10],
                ph[3][10],
                temperatura[3][10],
                potencial[3][10],
                oxigeno[3][10],
                mA[3][10],
                velocidad[3][10],
                caudales[3][10],
                set_point_oxigeno[3][10],
                set_point_agitacion[3][10],
                entradas_plc[16],
                salidas_plc[24],
                estados_varios[20],
                tiempos[10][10];
}datos;

int i;
unsigned long chequeo=0L;

memset(buffer, 0, sizeof(buffer));   /* Borra las estructuras */
memset(&datos, 0, sizeof(datos));

/* Introducimos las medidas en la estructura que pasamos a la SUN */

for (i=0;i<3;i++)
    {
    sprintf( datos.nom[i], "%s", medidas[i].nom );
    sprintf( datos.controlador[i], "%4i", medidas[i].controlador );
    sprintf( datos.ph[i], "%7.4f", medidas[i].ph );
    sprintf( datos.temperatura[i], "%7.4f", medidas[i].temperatura );
    sprintf( datos.potencial[i], "%7.4f", medidas[i].potencial );
    sprintf( datos.oxigeno[i], "%7.4f", medidas[i].oxigeno );
    sprintf( datos.mA[i], "%7.4f", medidas[i].mA );
    sprintf( datos.velocidad[i], "%7.4f", medidas[i].velocidad );
    sprintf( datos.set_point_agitacion[i], "%7.4f", planta.velocidad[i] );
    sprintf( datos.set_point_oxigeno[i], "%7.4f", planta.PID[i].set_point );
    }
```



```
sprintf( datos.caudales[0], "%7.4f", planta.caudal_agua);
sprintf( datos.caudales[1], "%7.4f", planta.caudal_RE);
sprintf( datos.caudales[2], "%7.4f", planta.caudal_RI);

for (i=0;i<16;i++)
    if (planta.entradas_plc[i]==1) datos.entradas_plc[i]=49;
    else datos.entradas_plc[i]=48;

for (i=0;i<24;i++)
    if (planta.salidas_plc[i]==1) datos.salidas_plc[i]=49;
    else datos.salidas_plc[i]=48;

for (i=0;i<3;i++) datos.estados_varios[i]=(char)(planta.PID[i].activo+48);
for (i=3;i<20;i++)  datos.estados_varios[i]=48;

sprintf(datos.tiempos[0],"%4i", planta.intervalo_SUN);
sprintf(datos.tiempos[1],"%5i", planta.duracion_purga);
sprintf(datos.tiempos[2],"%4i", planta.tiempo_purga);
sprintf(datos.tiempos[3],"%4i", planta.intervalo_dosificacion[0]);
sprintf(datos.tiempos[4],"%4i", planta.intervalo_dosificacion[1]);
sprintf(datos.tiempos[5],"%4i", planta.intervalo_flojet_on);
sprintf(datos.tiempos[6],"%4i", planta.intervalo_flojet_off);

for (i=7;i<9;i++) sprintf( datos.tiempos[i], "100");
sprintf( datos.tiempos[9], "%2i", planta.alarma);

/* Copiamos la estructura de datos al buffer */
memcpy(buffer, &datos, sizeof(datos));

/* Calculamos el cheksum del buffer */
for (i=0;i<(TAMANY-12);i++) chequeo+=buffer[i];

/* Introducimos el cheksum como cadena de caracteres en la posici¢n 500 */
sprintf(&buffer[500],"%lllu",chequeo);

}

/*************************************************************************/
/* A partir de las consignas mandadas por la SUN en el buffer, efectua las
   acciones necesarias para establecerlas en la planta piloto */

int cambio_consignas(void)
{
struct consignas{
       char cambio_oxigeno[3],
            cambio_agitacion[3],
            cambio_caudales[3],
            cambio_tiempos[10],
            valor_oxigeno[3][10],
            valor_agitacion[3][10],
            valor_caudales[3][10],
            tiempos[10][10],
            cambio_variable[20],
            valor_variable[20][10];
}datos;

unsigned long chequeo1=0L,
              chequeo2=0L;
int i;
float consigna;
int consignai;
int intervalo;
float mA=0.0F;
char mensaje[100];
int status=0;
int debug=0;

if (planta.variable_actual==20) debug=1;

for (i=0;i<(TAMANY-12);i++) chequeo1+=buffer[i];
chequeo2= atol(&buffer[500]);
/* Si falla el chequeo, devuelve un c¢digo de error -1 */
if (chequeo1!=chequeo2) return -1;

/* Copiamos el buffer a la estructura de datos */
memcpy(&datos,buffer,sizeof(datos));
```

```
for (i=0;i<3;i++)
       {
       if (datos.cambio_oxigeno[i]=='1')
           {
           if (debug) graf_mensaje("Cambio consigna ox¡geno G2");
           consigna = (float) atof(datos.valor_oxigeno[i]);
           if ( (consigna <= 10.0) && (consigna >= 0.0) )
                 planta.PID[i].set_point = consigna;
           sprintf(mensaje,
           "Nueva consigna ox¡geno reactor %i: %4.2f ppm", i+1, consigna);
           guarda_mensaje( mensaje );
           }
       if (datos.cambio_agitacion[i]=='1')
           {
           if (debug) graf_mensaje("Cambio consigna agitaci¢n G2");
           consigna = (float) atof(datos.valor_agitacion[i]);
           if ( (consigna <= 100.0) && (consigna >= 0.0) )
                 variar_velocidad( consigna, i+1);
           planta.velocidad[i]=consigna;
           sprintf(mensaje,
           "Nueva consigna agitaci¢n reactor %i: %4.2f ppm", i+1, consigna);
           guarda_mensaje( mensaje );
           }
       }

if (datos.cambio_caudales[0]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna caudal G2");
  consigna = (float) atof(datos.valor_caudales[0]);
  if ( (consigna <= 500.0) && (consigna >= 0.0) )
    {
    status = mA_caudal( "c:\\depura\\v_agua.dat", consigna, &mA);
    if (status==0)
          {
          planta.caudal_agua=consigna;
          pcl726(0,mA);       /*Abre la v lvula de entrada de agua*/
          sprintf(mensaje,
          "Nueva consigna: v lvula de agua abierta: Q = %4.1f ml/min, %4.2f mA  ",
                      planta.caudal_agua,mA);
          guarda_mensaje( mensaje );
          }
    else
     guarda_mensaje( "Consigna de caudal de agua recibida incorrecta\t\x0");

    }
  }

if (datos.cambio_caudales[1]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna caudal G2");
  consigna = (float) atof(datos.valor_caudales[1]);
  if ( (consigna <= 1000.0) && (consigna >= 0.0) )
    {
    status = mA_caudal( "c:\\depura\\bomba1.dat", consigna, &mA);
    if (status==0)
          {
          planta.caudal_RE=consigna;
          pcl726(1,mA);       /*Regula la bomba1, RE*/
          sprintf(mensaje,
          "Nueva consigna: bomba RE: Q = %4.1f ml/min, %4.2f mA  ",
                      planta.caudal_RE,mA);
          guarda_mensaje( mensaje );
          }
    else
     guarda_mensaje( "Consigna de caudal bomba-1 RE recibida incorrecta\t\x0");

    }
  }

if (datos.cambio_caudales[2]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna caudal G2");
  consigna = (float) atof(datos.valor_caudales[2]);
  if ( (consigna <= 2000.0) && (consigna >= 0.0) )
    {
    status = mA_caudal( "c:\\depura\\bomba2.dat", consigna, &mA);
    if (status==0)
```



```
          {
          planta.caudal_RI=consigna;
          pcl726(2,mA);       /*Regula la bomba2, RI*/
          sprintf(mensaje,
          "Nueva consigna: bomba RI: Q = %4.1f ml/min, %4.2f mA  ",
                      planta.caudal_RI,mA);
          guarda_mensaje( mensaje );
          }
    else
     guarda_mensaje( "Consigna de caudal bomba-2 RI recibida incorrecta\t\x0");

    }
  }

if (datos.cambio_tiempos[0]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna comunicaci¢n G2");
  consignai = atoi(datos.tiempos[0]);
  if ( (consignai <= 2000) && (consignai >= 10) )
    {
    planta.intervalo_SUN=consignai;
    sprintf(mensaje, " Nueva consigna: intervalo SUN, t = %4i seg",consignai);
    guarda_mensaje( mensaje );
    }
  else guarda_mensaje( "Consigna intervalo SUN recibida incorrecta\t\x0");
  }

if (datos.cambio_tiempos[1]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna purga G2");
  consignai =  atoi(datos.tiempos[1]);
  if ( (consignai <= 9999) && (consignai >= 0) )
    {
    planta.duracion_purga=consignai;
    intervalo=planta.duracion_purga;
    plc_timer( intervalo, 3);
    sprintf(mensaje,
         "Nueva consigna: duraci¢n purga, t = %5i /10 =seg",consignai);
    guarda_mensaje( mensaje );
    }
  else guarda_mensaje( "Consigna duraci¢n purga recibida incorrecta\t\x0");
  }
if (datos.cambio_tiempos[2]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna intervalo purga G2");
  consignai =  atoi(datos.tiempos[2]);
  if ( (consignai <= 1000) && (consignai >= 0) )
    {
    planta.tiempo_purga=consignai;
    intervalo=planta.tiempo_purga*10-planta.duracion_purga;
    plc_timer( intervalo, 4);
    sprintf(mensaje,
         "Nueva consigna: intervalo purga, t = %4i seg",consignai);
    guarda_mensaje( mensaje );
    }
  else guarda_mensaje( "Consigna intervalo purga recibida incorrecta\t\x0");
  }
if (datos.cambio_tiempos[3]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna intervalo N G2");
  consignai =  atoi(datos.tiempos[3]);
  if ( (consignai <= 1000) && (consignai >= 1) )
    {
    planta.intervalo_dosificacion[0]=consignai;
    intervalo=planta.intervalo_dosificacion[0]*10-1;
    if (plc_timer( intervalo, 11)!=0) plc_timer( intervalo, 11);
    sprintf(mensaje,
         "Nueva consigna: intervalo dosificaci¢n N, t = %4i seg",consignai);
    guarda_mensaje( mensaje );
    }
  else guarda_mensaje( "Consigna intervalo dosificaci¢n N recibida incorrecta\t\x0");
  }
if (datos.cambio_tiempos[4]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna intervalo C G2");
  consignai =  atoi(datos.tiempos[4]);
  if ( (consignai <= 1000) && (consignai >= 1) )
```

```
    {
    planta.intervalo_dosificacion[1]=consignai;
    intervalo=planta.intervalo_dosificacion[1]*10-1;
    if (plc_timer( intervalo, 13)!=0) plc_timer( intervalo, 13);
    sprintf(mensaje,
         "Nueva consigna: intervalo dosificación C, t = %4i seg",consignai);
    guarda_mensaje( mensaje );
    }
  else guarda_mensaje( "Consigna intervalo dosificación C recibida incorrecta\t\x0");
  }
if (datos.cambio_tiempos[5]=='1')
  {
  if (debug)   graf_mensaje("Cambio consigna flojet on G2");
  consignai =  atoi(datos.tiempos[5]);
  if ( (consignai <= 30) && (consignai >= 0) )
    {
    planta.intervalo_flojet_on=consignai;
    planta.intervalo_flojet_off=30-planta.intervalo_flojet_on;
     intervalo=planta.intervalo_flojet_on*10;
     if (plc_timer( intervalo, 30)!=0) plc_timer( intervalo, 30);
     if (plc_timer( intervalo, 32)!=0) plc_timer( intervalo, 32);

     intervalo=planta.intervalo_flojet_off*10;
     if (plc_timer( intervalo, 31)!=0) plc_timer( intervalo, 31);
     if (plc_timer( intervalo, 33)!=0) plc_timer( intervalo, 33);

    sprintf(mensaje,
         "Nueva consigna: intervalo flojet on, t = %4i seg",consignai);
    guarda_mensaje( mensaje );
    }
  else guarda_mensaje( "Consigna intervalo flojet on incorrecta\t\x0");
  }
if (datos.cambio_tiempos[9]=='1')
  {
  if (debug)   graf_mensaje("On-off G2");
  consignai =  atoi(datos.tiempos[9]);
  if  (consignai == 10)       alarma(6);
  else if  (consignai == 11) alarma(-2);
  else guarda_mensaje( "Alarma SUN incorrecta\t\x0");
  }


for (i=0;i<3;i++)   /* Cambio de estado PIDs */
  if (datos.cambio_variable[i]=='1')
    {
    consignai = atoi(datos.valor_variable[i]);
    if (planta.PID[i].activo!=0) /* Si no est  estimando OUR */
      {
      if (consignai == 1)
       {
       planta.PID[i].activo=1;
       sprintf(mensaje,"PID %i activado",i+1);
       guarda_mensaje( mensaje );
       }
      if (consignai == 2)
       {
       planta.PID[i].activo=2;
       sprintf(mensaje,"PID %i desactivado",i+1);
       guarda_mensaje( mensaje );
       }
      }
    }


for (i=4;i<6;i++)   /* Cambio de salida PCL726 */
  if (datos.cambio_variable[i]=='1')
    {
    consigna = (float) atof(datos.valor_variable[i]);
    if (consigna >= 4.0 && consigna <= 20.0)
      {
      pcl726 (i, consigna);   /* Actuación */
      sprintf(mensaje,"PCL726 %i, %7.4f mA", i, consigna);
      guarda_mensaje(mensaje);
      }
    else guarda_mensaje( "Consigna PCL726 incorrecta\t\x0");
    }
```



```
return 1;
}

/*************************************************************************/
/* Toma el buffer mandado por la SUN y da valores a la estructura
   de las medidas */

int nuevos_datos_SUN(void)
{
struct segunda{
       char nom[10],
            controlador[10],
            ph[10],
            temperatura[10],
            potencial[10],
            oxigeno[10],
            mA[10],
            velocidad[10];
}datos[3];

int i=0;
int  chequeo1=0L,
     chequeo2=0L;


for (i=0;i<(TAMANY-12);i++) chequeo1+=buffer[i];

chequeo2= atoi(&buffer[500]);
 if (chequeo1==chequeo2)
     printf("Cheksum OK\n");
else
     {
     printf("Falla el cheksum\n");
     return -1;
     }

memset(medidas, 0, sizeof(medidas));
memcpy(datos,buffer,sizeof(datos));

for (i=0;i<3;i++)
       {
       memcpy(medidas[i].nom, datos[i].nom, sizeof(medidas[i].nom));
       medidas[i].controlador =(short int)atoi(datos[i].controlador);
       medidas[i].ph = (float)atof(datos[i].ph);
       medidas[i].temperatura = (float)atof(datos[i].temperatura);
       medidas[i].potencial = (float)atof(datos[i].potencial);
       medidas[i].oxigeno = (float)atof(datos[i].oxigeno);
       medidas[i].mA = (float)atof(datos[i].mA);
       medidas[i].velocidad = (float)atof(datos[i].velocidad);
       }

return 1;
}

/*************************************************************************/

int mensaje_a_SUN(char *mensaje)
{
int toServerSocket = -1;
int i;                          /* Variables de control */
unsigned long chequeo=0L;
char caracter[1]={'M'};
time_t ltime;
char mensaje_cliente[TAMANY_M];

/*if (planta.conexion_SUN == 0) return -2;*/

memset(buffer, 0, sizeof(buffer));   /* Borra las estructuras */
memset (mensaje_cliente, 0, sizeof(mensaje_cliente));

time (&ltime);
sprintf(mensaje_cliente,"%s%s", mensaje, ctime( &ltime ));

/* Copiamos el mensaje al buffer */
memcpy(buffer, mensaje_cliente, sizeof(mensaje_cliente));

/* Calculamos el cheksum del buffer */
```

```
for (i=0;i<(TAMANY-12);i++) chequeo+=buffer[i];

/* Introducimos el cheksum como cadena de caracteres en la posici¢n 500 */
sprintf(&buffer[500],"%111u",chequeo);


/* Conexi¢n con la SUN */
toServerSocket = connectToServer(svrName, Port);

/* Si la conexi¢n no es correcta, devuelve -1 */
if (toServerSocket==-1)
    {
#ifdef DEPURA
    printf ("Error en la conexi¢n\n");
#endif
    planta.conexion_SUN=0;
    return -1;
    }
else planta.conexion_SUN=1;

#ifdef DEPURA
printf("CONEXIO A %s REALITZADA, %i\n",svrName,toServerSocket);
#endif

/* Mandamos una M, queremos mandar mensaje */
send(toServerSocket, caracter, sizeof(caracter), 0);

/* Mandamos el buffer */
send(toServerSocket, buffer, sizeof(buffer), 0);

/* Recibimos respuesta */
recv(toServerSocket, caracter, sizeof(caracter), 0);
/* Lo env¡a de nuevo si no ha sido bien recibido */
if (caracter[0]!='O')
    {
    caracter[0]='M';
    send(toServerSocket, caracter, sizeof(caracter), 0);

    send(toServerSocket, buffer, sizeof(buffer), 0);
    recv(toServerSocket, caracter, sizeof(caracter), 0);
    if (caracter[0]!='O')       /* Si lo recibe mal de nuevo, desconecta */
        {
        disconnectToServer(toServerSocket);
        return -3;
        }
    }

/* Pedimos desconexi¢n */
caracter[0]='F';
send(toServerSocket, caracter, sizeof(caracter), 0);
disconnectToServer(toServerSocket);

return 1;
}

/*************************************************************************/
/* Introduce los datos de la estructura medidas en el buffer,
   a¤adiendo un cheksum y llama a otra funci¢n que manda los datos */

void medida_a_SUN_varios(char variable, float valor)
{
struct variable_texto{
        char    variable[5];
        char    valor[10];
        }var_texto;

int i;
unsigned long chequeo=0L;

memset(buffer_p, 0, sizeof(buffer_p));      /* Borra las estructuras */
memset(&var_texto, 0, sizeof(var_texto));

/* Introducimos las medidas en la estructura que pasamos a la SUN */

sprintf(var_texto.variable,"%3i", variable);
sprintf(var_texto.valor,"%9.5f", valor);
```



```
/* Copiamos la estructura de datos al buffer */
memcpy(buffer_p, &var_texto, sizeof(var_texto));

/* Calculamos el cheksum del buffer */
for (i=0;i<(TAMANY_PETIT-12);i++) chequeo+=buffer_p[i];

/* Introducimos el cheksum como cadena de caracteres en la posici¢n 20 */
sprintf(&buffer_p[20],"%111u",chequeo);

/* Pasamos los datos a la SUN */
sockets_pasa_datos_petit();

}

/*************************************************************************/
/* Funci¢n para establecer la comunicaci¢n y controlar el env¡o y la
   recepci¢n de datos */

int sockets_pasa_datos_petit(void)
{
int toServerSocket = -1;
int i;                          /* Variables de control */
char caracter[1];

/* Si la conexi¢n anterior no ha sido correcta, vuelve */
if (planta.conexion_SUN == 0) return -2;

/* Conexi¢n con la SUN */
toServerSocket = connectToServer(svrName, Port);

/* Si la conexi¢n no es correcta, devuelve -1 */
if (toServerSocket==-1)
    {
#ifdef DEPURA
    printf ("Error en la conexi¢n\n");
#endif
    planta.conexion_SUN=0;
    guarda_mensaje( "La conexi¢n con la SUN no es correcta\t\t\x0");
    imprime( "La conexi¢n con la SUN no es correcta\v\v\n\x0");
    return -1;
    }

#ifdef DEPURA
printf("CONEXIO A %s REALITZADA, %i\n",svrName,toServerSocket);
#endif

caracter[0]='V';
/* Mandamos una V, queremos establecer un valor */
send(toServerSocket, caracter, sizeof(caracter), 0);

/* Mandamos el buffer */
i = send(toServerSocket, buffer_p, sizeof(buffer_p), 0);
#ifdef DEPURA
if (i==-1) error_send(errno);
#endif

/* Recibimos respuesta */
recv(toServerSocket, caracter, sizeof(caracter), 0);
/* Lo env¡a de nuevo si no ha sido bien recibido */
if (caracter[0]!='O')
    {
#ifdef DEPURA
    printf("miserver recibe mal\n");
#endif
    caracter[0]='V';
    send(toServerSocket, caracter, sizeof(caracter), 0);

    send(toServerSocket, buffer_p, sizeof(buffer_p), 0);
    recv(toServerSocket, caracter, sizeof(caracter), 0);
    if (caracter[0]!='O')       /* Si lo recibe mal de nuevo, desconecta */
        {
#ifdef DEPURA
printf("miserver recibe mal\n");
#endif
        disconnectToServer(toServerSocket);
        return -3;
        }
```

```
    }

/* Pedimos desconexi¢n */
caracter[0]='F';
send(toServerSocket, caracter, sizeof(caracter), 0);
disconnectToServer(toServerSocket);

return 0;
}

/*************************************************************************/
/*************************************************************************/
/*  connectToServer es usada por el cliente para abrir una conexi¢n     */
/*  con el servidor                                                      */
/*************************************************************************/

int connectToServer(char *serverName, ushort port)
{
struct sockaddr_in serverSockAddr;     /* Variable direcci¢n del host remoto */
struct hostent *serverHostEnt;         /* Puntero direcci¢n del host remoto */
int toServerSocket = (-1);             /* Descriptor del socket local */
ulong hostAddr;                        /* Direcci¢n del host */
int result = (-1);                     /* Supone fallo */
int error=0;
struct linger{
    int l_onoff;
    int l_linger;
}mi_linger;

/*************************************************************************/
/*      Borra y establece la estructura de la direcci¢n del servidor     */

memset(&serverSockAddr, 0, sizeof(serverSockAddr) );

hostAddr = inet_addr(serverName);       /* Devuelve la direcci¢n */
if((long)hostAddr != (long)(-1))        /* Hay direcci¢n */
       {
#ifdef DEPURA
printf("\nconnectToServer: se ha obtenido la direcci¢n de %s i es %lx \n", serverName,
hostAddr);
#endif
       memcpy( &(serverSockAddr.sin_addr), &hostAddr,sizeof(hostAddr));
       }
       else                                 /* No hay direcci¢n */
       {
#ifdef DEPURA
printf("\nconnectToServer: no se ha obtenido directamente la direcci¢n \n" );
#endif

/* Se consulta al servidor de nombres para determinar la direcci¢n del host */

       serverHostEnt =  gethostbyname(serverName); /* Asignaci¢n de la direcci¢n remota */
       if(NULL == serverHostEnt)
          {
#ifdef DEPURA
          fprintf(stderr, "Imposible localizar host \" %s \"\n", serverName);
#endif
          goto egress;
          }
/* Se copia la direcci¢n de entrada del host a la estructura del socket */

       memcpy(&(serverSockAddr.sin_addr), serverHostEnt->h_addr,serverHostEnt->h_length);
       };

serverSockAddr.sin_family = AF_INET;           /* Asignaci¢n de direcciones */
serverSockAddr.sin_port = (ushort)htons(port); /* Asignaci¢n del puerto */

/*************************************************************************/
/*      Se crea el socket                                                */

error = 0;

toServerSocket = socket( AF_INET, SOCK_STREAM, 0);
if (toServerSocket == -1)
       {
#ifdef DEPURA
       error_socket(errno);
```



```
#endif
       goto egress;
       }
#ifdef DEPURA
printf("El socket vale %i \n",toServerSocket);
#endif

if (setsockopt (toServerSocket, SOL_SOCKET, SO_REUSEADDR, "sisi", 4)==-1)
    printf ("Error en setsockopt, reusa\n");

mi_linger.l_onoff=1;
mi_linger.l_linger=10;

if (setsockopt (toServerSocket, SOL_SOCKET, SO_LINGER, (char *) &mi_linger,
              sizeof(mi_linger))==-1)
    printf ("Error en setsockopt, linger\n");

if (connect(toServerSocket, &serverSockAddr, sizeof(serverSockAddr)) == -1)
    {
    error = 1;
#ifdef DEPURA
    error_connect(errno);
#endif
    goto egress;
    }
result = toServerSocket;

egress:
return result;
}

/*************************************************************************/
/* Funci¢n para cerrar la conexi¢n con el servidor */

int disconnectToServer(int toServerSocket)
{
int shut,clo;

shut=shutdown(toServerSocket, 2);
clo=close(toServerSocket);

#ifdef DEPURA
if (shut== -1)   printf(" Error en shutdown toServerSocket.\n");
else             printf("Shutdown toServerSocket ok.\n");

if(clo == -1)    printf("Error en close toServerSocket.\n");
#endif

return clo;
}

/*************************************************************************/

void error_send(int errno)
{
switch(errno)
              {
              case EACCES:
              printf("ERROR: EACCES\n");
              break;
              case EFAULT:
              printf("ERROR: EFAULT\n");
              break;
              case ENOBUFS:
              printf("ERROR: ENOBUFS\n");
              break;
              case ENOTCONN:
              printf("ERROR: ENOTCONN\n");
              break;
              case ENOTSOCK:
              printf("ERROR: ENORSOCK\n");
              break;
              case ESHUTDOWN:
              printf("ERROR: ESHUTDOWN\n");
              break;
              case EWOULDBLOCK:
              printf("ERROR: EWOLULDBLOCK\n");
```

```
datf[i].xf = datf[i].lambda*datf[i].x+(1.0F-datf[i].lambda)*datf[i].xf_old;
datf[i].xf_old = datf[i].xf;

medidas[i].oxigeno=datf[i].xf;
}

if (planta.registro_O2==1)
    {
    stream = fopen( "c:\\datos\\filtroO2.dat", "a+");
    time (&ltime);
    t = localtime( &ltime );

    for (i=0;i<3;i++) fprintf( stream, " %6.4f ",datf[i].x);
    for (i=0;i<3;i++) fprintf( stream, " %6.4f ",datf[i].xf);
    for (i=0;i<3;i++) fprintf( stream, " %6.4f ",datf[i].lambda);
    for (i=0;i<3;i++) fprintf( stream, " %6.4f ",datf[i].deltafs);
    for (i=0;i<3;i++) fprintf( stream, " %6.4f ",medidas[i].mA);
    fprintf(stream, "   %2d %02d %02d  ", t->tm_hour, t->tm_min, t->tm_sec);
    fprintf( stream, " \n ");

    fclose(stream);
    }


return 1;
}

/*************************************************************************/

void inicio_filtro_digital(void)
{
int i;

for (i=0;i<3;i++)
    {
    datf[i].x_old=medidas[i].oxigeno;
    datf[i].xf_old=medidas[i].oxigeno;
    datf[i].E=0.5F;
    datf[i].deltafs=0.125F;
    }
}

/*************************************************************************/

int registro_O2(void)
{
int i;
FILE *stream;
struct tm *t;
time_t ltime;

stream = fopen( "c:\\datos\\O2.dat", "a+");
time (&ltime);
t = localtime( &ltime );
for (i=0;i<3;i++) fprintf( stream, " %6.4f ",medidas[i].oxigeno);
for (i=0;i<3;i++) fprintf( stream, " %6.4f ",medidas[i].mA);
fprintf(stream, "   %2d %02d %02d  ", t->tm_hour, t->tm_min, t->tm_sec);
fprintf(stream, "%s", ctime(&ltime));

fclose(stream);

return 1;
}

/*************************************************************************/

int contador_our[3];
float oxig[3][10];
float t_oxig[3][10];
time_t t_ini_our[3];
float our_a[3], our_b[3], our_r[3];
float old_ma[3];
float r_min1=0.75F, r_min2=0.85F, r_min3=0.75F, b_min=0.0F, b_max=0.05F;

void our_estimacion(int reactor, int step)
{
char mensaje[200];
```



```
switch(step)
   {
   case 0:   /* Fin estimaci¢n */
      planta.PID[reactor-1].activo=1;            /* Activa PID */
      if ( planta.PID[0].activo==1 && planta.PID[1].activo==1 &&
           planta.PID[2].activo==1)
        planta.intervalo_control_O2 = old_intervalo_o2;
                                        /* Recupera intervalo O2 */
    medidas[reactor-1].mA=old_ma[reactor-1];
                      /* Recupera £ltima acci¢n de control */
    pcl726 ((reactor-1)+3, old_ma[reactor-1]);      /* Abre la v lvula */
    variar_velocidad (planta.velocidad[reactor-1], reactor);
    delay(200);    /* Recupera la agitaci¢n */
    variar_velocidad (planta.velocidad[reactor-1], reactor);
   break;

   case 1:   /* Inicio estimaci¢n */
      planta.PID[reactor-1].activo=0;          /* Desactiva PID */
      old_ma[reactor-1]=medidas[reactor-1].mA;      /* Guarda actuaci¢n */
      medidas[reactor-1].mA=4.0F;
      pcl726 ((reactor-1)+3, 4.0F);  /* Cierra la v lvula */
      variar_velocidad (25.0F, reactor);
      delay(200);    /* Disminuye agitaci¢n */
      variar_velocidad (25.0F, reactor);  /* Disminuye agitaci¢n */

      planta.intervalo_control_O2=5; /* Aumenta frecuencia lectura */
      contador_our[reactor-1]=0;     /* Resetea contador */
      planta.t_phrocon.tiempo=planta.t_actual+45;  /* Evita lectura pHrocon */
      planta.t_phrocon.fin=0;
      memset(oxig,0,sizeof(oxig));          /* Borra datos anteriores */
      memset(t_oxig,0,sizeof(t_oxig));
      t_ini_our[reactor-1]=time(NULL);            /* Guarda t inicio */
   break;
   case 2:   /* Registro OUR */
    switch(reactor){
      case 1: /* Reactor 1 */
        oxig[0][contador_our[0]]=medidas[0].oxigeno;        /* Guarda medida */
        t_oxig[0][contador_our[0]]=
          (float)difftime(time(NULL),t_ini_our[0]); /* Guarda t medida */
        if (contador_our[0] > 4)                          /* Para a la 5| */
          {
          regresion_lineal( t_oxig[0], oxig[0], 6,
                   &(our_a[0]), &(our_b[0]), &(our_r[0]));
                              /* Calcula nuevos valores */
          our_estimacion(1,0);        /* Desactiva la estimaci¢n */
          sprintf(mensaje,
            "Nuevo Q O2 calculado reactor 1: %6.4e mg O2 / (1 s). r: %6.4e", -our_b[0],
our_r[0]);
          if (planta.variable_actual==20) graf_mensaje(mensaje);
         /*  guarda_mensaje( mensaje ); */
          if (our_r[0] > r_min1 && -our_b[0] > b_min && -our_b[0] < b_max )
                   medida_a_SUN_varios( 0, -our_b[0]);
          sprintf(mensaje, "%6.4e  %6.4e", -our_b[0], our_r[0]);
          guarda_medida( mensaje, "c:\\datos\\our1.dat");
          }
        /* sprintf( mensaje, " %6.4f ",medidas[0].oxigeno);
        guarda_medida( mensaje, "c:\\datos\\our-1.dat"); */
      break;
      case 2: /* Reactor 2 */
        oxig[1][contador_our[1]]=medidas[1].oxigeno;        /* Guarda medida */
        t_oxig[1][contador_our[1]]=
          (float)difftime(time(NULL),t_ini_our[1]); /* Guarda t medida */
        if (contador_our[1] > 4)                          /* Para a la 5| */
          {
          regresion_lineal( t_oxig[1], oxig[1], 6,
                   &(our_a[1]), &(our_b[1]), &(our_r[1]));
                              /* Calcula nuevos valores */
          our_estimacion(2,0);        /* Desactiva la estimaci¢n */
          sprintf(mensaje,
            "Nuevo Q O2 calculado reactor 2: %6.4e mg O2 / (1 s). r: %6.4e", -our_b[1],
our_r[1]);
          if (planta.variable_actual==20) graf_mensaje(mensaje);
         /*  guarda_mensaje( mensaje ); */
          if (our_r[1] > r_min2 && -our_b[1] > b_min && -our_b[1] < b_max )
                   medida_a_SUN_varios( 1,   -our_b[1]);
          sprintf(mensaje, "%6.4e  %6.4e", -our_b[1], our_r[1]);
```

```
            guarda_medida( mensaje, "c:\\datos\\our2.dat");
            }
          /* sprintf( mensaje, " %6.4f ",medidas[1].oxigeno);
          guarda_medida( mensaje, "c:\\datos\\our-2.dat"); */
        break;
        case 3: /* Reactor 3 */
          oxig[2][contador_our[2]]=medidas[2].oxigeno;        /* Guarda medida */
          t_oxig[2][contador_our[2]]=
            (float)difftime(time(NULL),t_ini_our[2]); /* Guarda t medida */
          if (contador_our[2] > 4)                        /* Para a la 5| */
            {
            regresion_lineal( t_oxig[2], oxig[2], 6,
                    &(our_a[2]), &(our_b[2]), &(our_r[2]));
                          /* Calcula nuevos valores */
            our_estimacion(3,0);        /* Desactiva la estimaci¢n */
            sprintf(mensaje,
              "Nuevo Q O2 calculado reactor 3: %6.4e mg O2 / (l s). r: %6.4e", -our_b[2],
our_r[2]);
            if (planta.variable_actual==20) graf_mensaje(mensaje);
           /*  guarda_mensaje( mensaje );*/
            if (our_r[2] > r_min3 && -our_b[2] > b_min && -our_b[2] < b_max )
                    medida_a_SUN_varios( 2,      -our_b[2]);
            sprintf(mensaje, "%6.4e  %6.4e", -our_b[2], our_r[2]);
            guarda_medida( mensaje, "c:\\datos\\our3.dat");
            }
          /* sprintf( mensaje, " %6.4f ",medidas[2].oxigeno);
          guarda_medida( mensaje, "c:\\datos\\our-3.dat"); */
        break;
      }
      contador_our[reactor-1]++;

    break;
    }
}
```



```
/*****************************************************************************

   CRISON.C -  FUNCIONES PARA LA COMUNICACION CON LOS CONTROLADORES DE
               SONDAS CRISON-PHROCON18

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 30/4/98

*****************************************************************************/

/******************** Librerias  a incluir *************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <math.h>
#include <time.h>
#include <memory.h>
#include "nombres.h"

/*********************** Definiciones ********************************/

#define COM4    3
#define SETTINGS4 (_COM_9600 | _COM_CHR7 | _COM_STOP1 | _COM_EVENPARITY)
#define DTR     0x01     /* Data Terminal Ready */
#define RTS     0x02     /* Ready To Send */
#define DCOM3   0x3E8
#define DCOM4   0x2E8
#define THRE    0x2000   /* Bit 13, Transmission-hold register empty */

/*define TEST*/

/*********************** Subrutinas del fichero **********************/

int inicializar_sondas(void);
int recibe(void);
int phrocon_orden(char *orden, char *respuesta);
void phrocon_lecturas(void);
void phrocon_ph(void);
void phrocon_T(void);
void phrocon_mv(void);
void phrocon_O2(void);
float lectura_ph(void);
float lectura_T(void);
float lectura_mv(void);
void lectura_valores(int sonda);

/******************** Subrutinas externas necesarias ******************/

extern void dec_to_bin(int numero_decimal);
extern float minutos_desde_inicio(void);

/****************** Variables comunes del programa ********************/

extern estado_planta planta;
extern primera medidas[3];

struct correlacion{
       float a;         /* ordenada origen correlaci¢n O2-mV */
       float b;         /* pendiente correlaci¢n O2-mV */
}corr[3];

char entrada[100];        /*Variable de lectura Phrocon18*/


/********************** Variables externas necesarias *****************/


/*****************************************************************************
***************************   PHROCON-18      ********************************
*****************************************************************************/

/*****************************************************************************/
/*  Obliga a los pHrocon 18 a leer los dos canales,
```

```
    cambia de hold a lectura, reinicializa                                  */

int inicializar_sondas(void)
{
int sonda,in;
int suma=0;
char obliga_lectura[3][26]={" S01I01C1M02WrC2M02WrK3C ",
                            " S01I02C1M02WrC2M02WrK3D ",
                            " S01I03C1M02WrC2M02WrK3E "};

in=_bios_serialcom(_COM_INIT, COM4, SETTINGS4);    /*Inicializa el port com.*/

/*fputs("STATUS COM4: \t",stdout);
dec_to_bin(in);*/

for (sonda=0;sonda<3;sonda++)
   {
   suma=suma+phrocon_orden(obliga_lectura[sonda] ,entrada);
   }
return suma;
}

/*************************************************************************/
/*  Recibe la respuesta a las ¢rdenes enviadas a los pHrocon 18          */
/*  La respuesta debe comenzar por un char 5 y acabar por un char 4.
    Para evitar el bloqueo si no existe comunicaci¢n, se establecen unos
    l¡mites de 15 lecturas para el comienzo y 50 lecturas para el total. */


int recibe(void)
{
int j=0,error=0;
int kk=0;

_outp ( DCOM4 + 4, 0);
memset(entrada,0,sizeof(entrada));

do {
   error++;
   kk=0;
   do kk++;
   while ((0x100 & _bios_serialcom(_COM_STATUS, COM4,0))==0 && kk<30000);
   entrada[0] =(char) _inp (DCOM4);
   }  while (entrada[0]!=5 && error<15);
   if (error==15) return 1;
error=0;
do {
   error++;
   j++;
   kk=0;
   do kk++;
   while ((0x100 & _bios_serialcom(_COM_STATUS, COM4,0))==0 && kk<30000);
   entrada[j] = (char) _inp (DCOM4);
   }  while (entrada[j]!=4 && error<50);
   if (error==50) return 2;

printf("%s\n",entrada);
return 0;
}

/*************************************************************************/
/*  Manda una orden a un pHrocon 18 y recibe la respuesta.
    La respuesta debe comenzar por un char 5 y acabar por un char 4.
    Para evitar el bloqueo si no existe comunicaci¢n, se establecen unos
    l¡mites de 15 lecturas para el comienzo y 50 lecturas para el total.
    Devuelve los siguientes valores:

        -1  En 15 lecturas no ha existido un 5 de inicio
        -2  En 50 lecturas no ha existido un 4 de finalizaci¢n
        -3  Orden no correcta
        -4  Existe respuesta pero no cumple el checksum
         1  Existe respuesta y cumple el checksum
*/

int phrocon_orden(char *orden, char *respuesta)
{
int     j=0,kk=0,
```



```
        largo,
        error=0,
        chequeo=0;
char cheksum[3];

/*printf("%s\n",orden);*/
if (strchr(orden,' ')==NULL) return -3;

memset(entrada,0,sizeof(entrada));

while(orden[j]!=4)  j++;   /* Para determinar la longitud de la orden */
largo=j;
_outp ( DCOM4 + 4, DTR | RTS );

for (j=0;j<=largo;j++)
    {
      _outp ( DCOM4 , orden[j]);
      kk=0;
      do kk++;
      while ((THRE & _bios_serialcom(_COM_STATUS, COM4,0))==0 && kk<30000);
    }

_outp ( DCOM4 + 4, 0 );

do {
   error++;
   kk=0;
   do kk++;
   while ((0x100 & _bios_serialcom(_COM_STATUS, COM4,0))==0 && kk<30000);
   respuesta[0] =(char) _inp (DCOM4);
   }  while (respuesta[0]!=5 && error<15);
   if (error==15) return -1;
error=0;
j=0;
do {
   error++;
   j++;
   kk=0;
   do kk++;
   while ((0x100 & _bios_serialcom(_COM_STATUS, COM4,0))==0 && kk<30000);
   respuesta[j] = (char) _inp (DCOM4);
   }  while (respuesta[j]!=4 && error<50);
   if (error==50) return -2;

respuesta[j+1]=0;

for (kk=0;kk<=j-4;kk++) chequeo=chequeo+respuesta[kk];
chequeo=0xFF & chequeo;

sprintf(cheksum,"%02X",chequeo);
/*
printf("%X\t%s\n",chequeo,cheksum);
printf("%s\n",respuesta);
*/
if (cheksum[0]==respuesta[j-2] && cheksum[1]==respuesta[j-1])  return 1;
else return -4;
}

/*************************************************************************/
/* Pide la lectura de todos los valores de los pHrocon 18.
   Si la lectura es correcta llama a una funci¢n que transforma la
   respuesta en los valores float de la estructura medidas.              */

void phrocon_lecturas(void)
{
int sonda,estado_lectura;
char salida[3][28]={" S01I01C1M03RM04RC2M03RK53 ",
                    " S01I02C1M03RM04RC2M03RK54 ",
                    " S01I03C1M03RM04RC2M03RK55 "};

for (sonda=0;sonda<3;sonda++)
{
   estado_lectura=phrocon_orden( salida[sonda], entrada);
        /* Si la lectura phrocon es correcta leemos los valores */
   if (estado_lectura==1) lectura_valores(sonda);
}
```

```
return;
}

/*************************************************************************/
/* Pide la lectura de los valores de pH de los pHrocon18
   Si la lectura es correcta llama a una funci¢n que transforma la
   respuesta en los valores float de la estructura medidas.              */

void phrocon_ph(void)
{
int sonda,estado_lectura;
char salida_ph[3][18]={" S01I01C1M03RKD9 ",
                       " S01I02C1M03RKDA ",
                       " S01I03C1M03RKDB "};

for (sonda=0;sonda<3;sonda++)
   {
   estado_lectura=phrocon_orden( salida_ph[sonda], entrada);
        /* Si la lectura phrocon es correcta leemos los valores */
   if (estado_lectura==1)  medidas[sonda].ph =lectura_ph();
   }
return;
}

/*************************************************************************/
/* Pide la lectura de los valores de T de los pHrocon18
   Si la lectura es correcta llama a una funci¢n que transforma la
   respuesta en los valores float de la estructura medidas.              */

void phrocon_T(void)
{
int sonda,estado_lectura;
char salida_T[3][18]={" S01I01C1M04RKDA ",
                      " S01I02C1M04RKDB ",
                      " S01I03C1M04RKDC "};

for (sonda=0;sonda<3;sonda++)
   {
   estado_lectura=phrocon_orden( salida_T[sonda], entrada);
        /* Si la lectura phrocon es correcta leemos los valores */
   if (estado_lectura==1) medidas[sonda].temperatura =lectura_T();
   }
return;
}

/*************************************************************************/
/* Pide la lectura de los valores de mV de los pHrocon18
   Si la lectura es correcta llama a una funci¢n que transforma la
   respuesta en los valores float de la estructura medidas.              */

void phrocon_mv(void)
{
int sonda,estado_lectura;
char salida_mv[3][18]={" S01I01C2M03RKDA ",
                       " S01I02C2M03RKDB ",
                       " S01I03C2M03RKDC "};

for (sonda=0;sonda<3;sonda++)
   {
   estado_lectura=phrocon_orden( salida_mv[sonda], entrada);
        /* Si la lectura phrocon es correcta leemos los valores */
   if (estado_lectura==1) medidas[sonda].potencial =lectura_mv();
   }
return;
}

/*************************************************************************/
/* Pide la lectura de los valores de O2 de los pHrocon18.
   Primero debe asegurarse que los rel,s correspondientes a las salidas
   20 a 22 est n apagados. Despu,s se pide la lectura de los milivoltios.
   Si la lectura es correcta llama a una funci¢n que transforma la
   respuesta en valores float. Utilizando la correlaci¢n mv-O2 se transforma
   esa lectura en los valores de O2 de la estructura medidas. Por £ltimo
   debe asegurarse que los rel,s quedan encendidos para leer mV.         */

void phrocon_O2(void)
{
```



```
int sonda,estado_lectura;
float mv=0;
char salida_mv[3][18]={" S01I01C2M03RKDA ",
                       " S01I02C2M03RKDB ",
                       " S01I03C2M03RKDC "};


/*for (j=20;j<23;j++)  estado_salida(j,0);
delay (1500);*/

for (sonda=0;sonda<3;sonda++)
   {
   estado_lectura=phrocon_orden( salida_mv[sonda], entrada);
        /* Si la lectura phrocon es correcta leemos los valores */
   if (estado_lectura==1) mv =lectura_mv();
   else continue;
   medidas[sonda].oxigeno=corr[sonda].a+corr[sonda].b*mv;
   if (medidas[sonda].oxigeno < 0.07) medidas[sonda].oxigeno=0;
   }

/*for (j=20;j<23;j++)  estado_salida(j,1);
delay(1000);*/

return;
}

/*************************************************************************/
/* Funci¢n que transforma la respuesta a la petici¢n de pH al pHrocon 18
   en un valor float que devuelve.                                       */

float lectura_ph(void)
{
  int i,inicio,final;
  char ph[6];
  char *ptr, c1 = 'M', c2='p';
  float valor_medido;
  ptr = strchr(entrada, c1);
  inicio = ptr-entrada+3;
  ptr = strchr(entrada, c2);
  final = ptr-entrada;

  for (i=0;i<6;i++)  ph[i]=' ';
  for (i=inicio;i<final;i++) ph[i-inicio]=entrada[i];
  valor_medido= (float)atof(ph);

return valor_medido;
}

/*************************************************************************/
/* Funci¢n que transforma la respuesta a la petici¢n de T al pHrocon 18
   en un valor float que devuelve.                                       */

float lectura_T(void)
{
  int i,inicio,final;
  char T[5];
  char *ptr, c1 = 'M', c2='c';
  float valor_medido;
  ptr = strchr(entrada, c1);
  inicio = ptr-entrada+3;
  ptr = strchr(entrada, c2);
  final = ptr-entrada;

  for (i=0;i<5;i++)  T[i]=' ';
  for (i=inicio;i<final;i++) T[i-inicio]=entrada[i];
  valor_medido= (float) atof(T);

return valor_medido;
}

/*************************************************************************/
/* Funci¢n que transforma la respuesta a la petici¢n de mV al pHrocon 18
   en un valor float que devuelve.                                       */

float lectura_mv(void)
{
  int i,inicio,final;
```

```
  char mv[5];
  char *ptr, c1 = 'M', c2='m';
  float valor_medido;
  ptr = strchr(entrada, c1);
  inicio = ptr-entrada+3;
  ptr = strchr(entrada, c2);
  final = ptr-entrada;

  for (i=0;i<5;i++)  mv[i]=' ';
  for (i=inicio;i<final;i++) mv[i-inicio]=entrada[i];
  valor_medido= (float)atof(mv);

return valor_medido;
}

/*************************************************************************/
/* Funci¢n que transforma la respuesta a la petici¢n de todos los valores
   al pHrocon18 en los valores float de las variables en la estructura
   medidas.                                                              */


void lectura_valores(int sonda)
{
  int i;
  char ph[6], T[6], mv[6];

  memset(ph, 0, sizeof(ph));
  memset(T, 0, sizeof(T));
  memset(mv, 0, sizeof(mv));

  if (entrada[12]==49)
    {
    for (i=0;i<5;i++) ph[i]=entrada[12+i];
    for (i=1;i<48;i++) entrada[i]=entrada[i+1];
    }
  else
    for (i=0;i<4;i++) ph[i]=entrada[12+i];
  medidas[sonda].ph= (float) atof(ph);

  for (i=0;i<4;i++) T[i]=entrada[21+i];
  medidas[sonda].temperatura= (float) atof(T);

  if (entrada[31]!=45)
    {
    mv[0]=48;
    for (i=0;i<3;i++) mv[i+1]=entrada[31+i];
    }
  else
   {
   for (i=0;i<4;i++) mv[i]=entrada[31+i];
   }
   medidas[sonda].potencial= (float) atof(mv);

return;
}

/*************************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
    Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST
int main(void)
{
inicializar_sondas();
phrocon_lecturas();
phrocon_ph();
phrocon_T();
phrocon_mv();
phrocon_O2();

return 0;
}


void dec_to_bin(int numero_decimal)
{
   int bin[16];
```



```
   int i;

   for (i=0;i<16;i++)
     {
      bin[i] = numero_decimal & 1;
      numero_decimal >>= 1;
     }
return;
}


time_t t_actual, t_phrocon, t_inicio;

float dia_absoluto(void)
{
float dias=0;
time_t t_referencia=3015817200L;
float  dia_referencia=132.0F;

dias=(float) (dia_referencia+difftime(t_actual,t_referencia)/3600.0F/24.0F);

return dias;
}

float minutos_desde_inicio(void)
{
   float segundos=0,
         minutos=0;

   segundos= (float) t_phrocon-t_inicio;
   minutos=segundos/60.0F;

return minutos;
}

#endif
```

```
/*****************************************************************************

   DANFOSS.C - FUNCIONES PARA LA COMUNICACION CON LOS VARIADORES DE
               FRECUENCIA DANFOSS 2100-LT

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 17/9/95

*****************************************************************************/

/********************* Librerias  a incluir ***************************/

#include <stdio.h>
#include <string.h>
#include <bios.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <graph.h>
#include <time.h>
#include "nombres.h"

/************************ Definiciones *********************************/

#define COM2    1
#define SETTINGS2 (_COM_300 | _COM_CHR8 | _COM_STOP1 | _COM_NOPARITY)
#define DCOM2   0x2F8
#define THRE    0x2000  /* Bit 13, Transmission-hold register empty */
/*#define TEST*/

/************************ Subrutinas del fichero ***********************/

void inicializar_variador_frecuencia(void);
int danfoss_orden(char *orden, char *respuesta);
int variar_velocidad(double velocidad, int variador);
int control_variador(int estado, int variador);
void graf_control_agitador(int on_off);
void graf_velocidad_agitadores(void);

/********************* Subrutinas externas necesarias ******************/

extern void dec_to_bin(int numero_decimal);
extern void graf_borra_ayuda(void);
extern void graf_ayuda_general(void);

#ifdef TEST
void dec_to_bin(int numero_decimal);
void graf_borra_ayuda(void);
void graf_ayuda_general(void);
void selectmenu(void);
void main(void);
#endif

/******************* Variables comunes del programa ********************/

extern primera medidas[3];
extern estado_planta planta;

/*****************************************************************************/

/*****************************************************************************/
/*  Inicializa el puerto de comunicaciones COM2 con las variables
    definidas en SETTINGS2.                                                  */

void inicializar_variador_frecuencia(void)

{
int in;

/*char out[23]={"0<00COOE@0516+800003??>"}; */

/*                        bytes         chr$()      ASCII        */
/*                      ------------------------------------    */
/* start byte           1              60          <  (inicio) */
/* address                2,3             48,48       0,0 (enviar a todos) */
/* control char.          4               67           C  (control) */
/* control/status word    5,6,7,8                                 */
```



```
/* menu/parameter #      9,10,11,12                                  */
/* sign                  13              43          +    (signo) */
/* data                  14,15,16,17,18                              */
/* comma                 19                                          */
/* check sum             20,21           63,63       ?,? (pasar de checksum) */
/* stop byte             22              62              >  (fin)    */


in=_bios_serialcom(_COM_INIT, COM2, SETTINGS2);    /*Inicializa el puerto com.*/
/*fputs("STATUS COM2: \t",stdout);
dec_to_bin(in);*/

return;
}

/*****************************************************************************/
/*  Esta funci¢n s¢lo funciona si hay un solo variador encendido.*/

/*  Manda una orden a un danfoss y recibe la respuesta.
    La respuesta debe comenzar por un char 60 y acabar por un char 62.
    Para evitar el bloqueo si no existe comunicaci¢n, se establecen unos
    l¡mites de 15 lecturas para el comienzo y 50 lecturas para el total.
    Devuelve los siguientes valores:

        -1  En 15 lecturas no ha existido un 60 de inicio
        -2  En 50 lecturas no ha existido un 62 de finalizaci¢n
        -3  Orden no correcta
        -4  Existe respuesta pero no cumple el checksum
         1  Existe respuesta y cumple el checksum
*/

int danfoss_orden(char *orden, char *respuesta)
{
int     j=0,kk=0,
        largo=21,
        error=0,
        chequeo=0;
char cheksum[5];

if (strchr(orden,62)==NULL) return -3;

for (kk=1;kk<=18;kk++) chequeo=chequeo+orden[kk];
sprintf(cheksum,"%i",chequeo);

orden[19]=cheksum[2];
orden[20]=cheksum[3];

printf("%s\n",orden);

for (j=0;j<=largo;j++)
    {
      _outp ( DCOM2 , orden[j]);
      kk=0;
      do kk++;
      while ((THRE & _bios_serialcom(_COM_STATUS, COM2,0))==0 && kk<30000);
    }

do {
   error++;
   kk=0;
   do kk++;
   while ((0x100 & _bios_serialcom(_COM_STATUS, COM2,0))==0 && kk<30000);
   respuesta[0] =(char) _inp (DCOM2);
   }  while (respuesta[0]!=60 && error<15);
   if (error==15) return -1;
error=0;
j=0;
do {
   error++;
   j++;
   kk=0;
   do kk++;
   while ((0x100 & _bios_serialcom(_COM_STATUS, COM2,0))==0 && kk<30000);
   respuesta[j] = (char) _inp (DCOM2);
   }  while (respuesta[j]!=62 && error<50);
   if (error==50) return -2;
```

```
respuesta[j+1]=0;

chequeo=0;
for (kk=1;kk<=j-3;kk++) chequeo=chequeo+respuesta[kk];
sprintf(cheksum,"%i",chequeo);

printf("%i\n",chequeo);
printf("%s\n",respuesta);

if (cheksum[0]==respuesta[j-2] && cheksum[1]==respuesta[j-1])  return 1;
else return -4;
}

/*************************************************************************/
/* Funci¢n para variar la velocidad de cualquier variador independientemente,
   o los tres a la vez.                                                  */

int variar_velocidad(double velocidad, int variador)
{
   char out[23]={"0<00UOOE@0516+500003\?\?>"};
   char *string;
   int j, dec, sign;
   int ndig=5;

if (velocidad<-100 || velocidad>100) return 1;
if (variador<0 || variador>3)     return 2;
        /*Si los par metros no son posibles devuelve un c¢digo de error*/
out[3]= (char) (48+variador);

string = ecvt(velocidad, ndig, &dec, &sign);

if (sign==0) out[13]='+';               /*Signo de la velocidad*/
       else out[13]='-';

for (j=0;j<5;j++) out[14+j]=string[j];   /*Velocidad*/
out[19]= (char) (5-dec+48);           /*N€mero de decimales*/

for (j=1;j<23;j++) _bios_serialcom(_COM_SEND, COM2, out[j]);
                        /* envia el caracter por el puerto de com.*/

/*if (variador!=0)
    for (j=1;j<23;j++) out[j]=_bios_serialcom(_COM_RECEIVE, COM2, 0);*/
                        /* recibe el mensaje devuelto por el variador */
                        /* solo si utilizamos checksum */

if (variador==0)  for (j=0;j<3;j++) medidas[j].velocidad=(float)velocidad;
else medidas[variador-1].velocidad=(float)velocidad;

return 0;
}

/*************************************************************************/
/* Funci¢n para encender, apagar o apagada de emergencia de cualquiera
   de los variadores o de los tres a la vez.                             */


int control_variador(int estado, int variador)
{
   int j;
   char out[23]={"0<00COOE@0000+000000\?\?>"};

if (estado!=0 && estado!=1 && estado!=2) return 1;
if (variador<0 || variador>3)     return 2;
        /*Si los par metros no son posibles devuelve un c¢digo de error*/

out[3]= (char) (48+variador);

if (estado==0) out[5]=78;
if (estado==1) out[5]=79;
if (estado==2) out[5]=75;

for (j=1;j<23;j++) _bios_serialcom(_COM_SEND, COM2, out[j]);
                        /* envia el caracter por el puerto de com.*/
/*for (j=1;j<23;j++) printf("%c",out[j]);
    printf("\n");*/

return 0;
```



```
}

/*************************************************************************/
/* Funci¢n para llamar a control_variador desde la pantalla gr fica.     */


void graf_control_agitador(int on_off)
{
int var;
char fuente[20]="t'Tms Rmn'h12w50b";

graf_borra_ayuda();      /*Borra la zona de la ayuda*/
_setgtextvector( 1, 0 );   /*Texto horizontal*/
_setfont( fuente );      /*Fuente utilizada*/

_setcolor(7);
_moveto(5,2);
if (on_off==1) _outgtext("Encendido  agitador(es):   1,2,3  ¢  0 (todos)");
if (on_off==0) _outgtext("Apagado  agitador(es):   1,2,3  ¢  0 (todos)");
_moveto(5,19);
_setcolor(29);
_outgtext("N€mero de variador: ");
_settextposition(2,17);
fscanf (stdin, "%d", &var);
control_variador( on_off , var );

graf_ayuda_general();     /*Deja las cosas como estaban*/
}

/*************************************************************************/
/* Funci¢n para llamar a variar_velocidad desde la pantalla gr fica.     */


void graf_velocidad_agitadores(void)
{
int var;
double vel;
char fuente[20]="t'Tms Rmn'h12w50b";

graf_borra_ayuda();      /*Borra la zona de la ayuda*/
_setgtextvector( 1, 0 );   /*Texto horizontal*/
_setfont( fuente );      /*Fuente utilizada*/

_setcolor(7);
_moveto(5,2);
_outgtext("Variaci¢n  velocidad  agitador(es):   1,2,3  ¢  0 (todos)");
_moveto(5,19);
_setcolor(29);
_outgtext("N€mero de variador: ");
_settextposition(2,17);
fscanf (stdin, "%d", &var);
_moveto(250,18);
_outgtext("Velocidad:  ");
_settextposition(2,46);
fscanf (stdin, "%lf", &vel);

variar_velocidad (vel, var);

graf_ayuda_general();     /*Deja las cosas como estaban*/
}

/*************************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
    Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST
int fi=1;
void main(void)
{
char out[23]={"<00UOOE@0516+000003\?\?>"};
char entrada[100];

inicializar_variador_frecuencia();

danfoss_orden(out, entrada);

/*
```

```
control_variador(1,1);
do{
selectmenu();

   }
while(fi==1);
*/
}

void selectmenu(void)
{
int tecla;
int var;
double vel;

tecla=_bios_keybrd(_NKEYBRD_READY);
if(tecla==0) return;
_bios_keybrd(_NKEYBRD_READ); /*Vac¡a el buffer, lee el fltimo caracter*/

switch(tecla){
       case 0x3f00:
           printf ("N£mero de variador: ");
           fscanf (stdin, "%d", &var);
           control_variador( 1 , var );
       break;
       case 0x4000:
       printf ("N£mero de variador: ");
           fscanf (stdin, "%d", &var);
           control_variador( 0 , var );
       break;
       case 0x4100:
           control_variador( 2 , 0 );
       break;
       case 0x4200:
           printf ("N£mero de variador: ");
           fscanf (stdin, "%d", &var);
           printf ("Velocidad:  ");
           fscanf (stdin, "%lf", &vel);

           variar_velocidad (vel, var);
       break;
       case 0x4400:
            fi=0;
       break;
     }
}


void dec_to_bin(int numero_decimal)
{
   int bin[16];
   int i;

   for (i=0;i<16;i++)
     {
      bin[i] = numero_decimal & 1;
      numero_decimal >>= 1;
     }

   for (i=0;i<16;i++)  printf("%i",bin[15-i]);
   printf("\n");
return;
}
void graf_borra_ayuda(void)
{}
void graf_ayuda_general(void)
{}

#endif
```



```
/*****************************************************************************

   DELAY.C      FUNCION PARA ESPERAR LOS MILISEGUNDOS DETERMINADOS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 23/6/95

*****************************************************************************/

/********************* Libreriras  a incluir ************************/

#include <time.h>
#include <conio.h>
#include <stdio.h>

/********************** Subrutinas del fichero **********************/

void delay( clock_t wait );

/*****************************************************************************/

/* Pauses for a specified number of microseconds. */
void delay( clock_t wait )
{
    clock_t goal;

    goal = wait + clock();
    while( goal > clock() )
        ;
}

/*****************************************************************************/

#ifdef  TEST
void main (void)
{
    int duracion=1000;
    clock_t  cstart, cend;

    cstart = clock();

    delay ( duracion );

    cend = clock();
    printf( "\nTiempo transcurrido:\t\t%6.4f seconds \n",
           ((float)cend - cstart) / CLOCKS_PER_SEC );
}
#endif
```

```
/*****************************************************************************

   DIGITAL.C - FUNCION PARA CONVERSION DE ENTEROS A BINARIOS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 26/6/95

*****************************************************************************/

/******************** Librerias  a incluir ************************/

#include <math.h>
#include <stdio.h>

/********************** Subrutinas del fichero **********************/

#ifdef TEST
void main(void);
#endif
void dec_to_bin(int numero_decimal);


/************************************************************
* Subrutina de conversion de un numero decimal en un binario *
************************************************************/

void dec_to_bin(int numero_decimal)
{
   int bin[16];
   int i;

   for (i=0;i<16;i++)
     {
      bin[i] = numero_decimal & 1;
      numero_decimal >>= 1;
     }
/*
   for (i=0;i<16;i++)  printf("%i",bin[15-i]);
   printf("\n");
*/
return;
}

#ifdef TEST
void main(void)
{
   int i;
   for (i=0;i<10;i++)  dec_to_bin(i);
}
#endif
```



```
/*****************************************************************************

   FICHEROS.C -       FUNCIONES PARA LA LECTURA Y ESCRITURA DE FICHEROS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 21/9/98

*****************************************************************************/

/********************** Definiciones ********************************/
#ifdef TEST
#define extern
#endif
/********************** Librerias  a incluir ************************/

#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <process.h>
#include <conio.h>
#include <time.h>
#include <dos.h>
#include <direct.h>
#include <errno.h>
#include "nombres.h"

/********************** Subrutinas del fichero **********************/

void lectura_configuracion(void);
void lectura_alimentacion(void);
void archivar_datos_medida(void);
void fichero_semanal(void);
void guarda_mensaje( char *mensaje);
void guarda_medida( char *mensaje, char *fichero);
void seguro_graficas(int orden);
void archivar_datos_escalon(void);

/******************** Subrutinas externas necesarias ****************/

extern float dia_absoluto(void);

/****************** Variables comunes del programa ******************/

extern int hora, dia;

extern primera medidas[3];
extern estado_planta planta;

extern struct correlacion{
       float a;       /* ordenada origen correlaci¢n O2-mV */
       float b;       /* pendiente correlaci¢n O2-mV */
}corr[3];

extern float mat_graf_e1[3][6][360]; /*Matrices para las gr ficas*/
extern float mat_graf_e2[3][6][480];
extern float mat_graf_e3[3][6][480];
extern time_t tiempo_e1[360];        /*Matrices tiempo gr ficas*/
extern time_t tiempo_e2[480];
extern time_t tiempo_e3[480];

/*****************************************************************************/
/* Fichero para leer la configuraci¢n base inicial del fichero de
   datos depura.dat                                                          */


void lectura_configuracion(void)
{
FILE *obert;
char linea[150];
int i;
int si;

if ( access("c:\\depura\\depura.dat",0) == 0)
  {
  obert=fopen("c:\\depura\\depura.dat","rt");

  /*En primer lugar se lee la l¡nea y despu,s se traduce a n£meros*/
```

```
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.intervalo_control_O2);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.intervalo_OUR);
     if (planta.intervalo_OUR==-1) planta.OUR=0;
     else planta.OUR=1;
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d%d",                                  &planta.intervalo_dosificacion[0],
&planta.intervalo_dosificacion[1]);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d%d", &planta.intervalo_limp_dosif[0], &planta.intervalo_limp_dosif[1]);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d%d", &planta.pistonadas_limp_dosif[0], &planta.pistonadas_limp_dosif[1]);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.intervalo_control_nivel);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.intervalo_toma_de_datos);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.tiempo_purga);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.duracion_purga);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.actualizacion_grafica);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%f", &planta.caudal_agua);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%f", &planta.caudal_RI);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%f", &planta.caudal_RE);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.intervalo_flojet_on);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.intervalo_flojet_off);

  for (i=0;i<3;i++) fscanf(obert,"%f",&planta.velocidad[i]);
  fscanf(obert,"%*[^0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%f",&planta.PID[i].set_point);
  fscanf(obert,"%*[^0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%f",&planta.PID[i].Kc);
  fscanf(obert,"%*[^0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%f",&planta.PID[i].Ti);
  fscanf(obert,"%*[^0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%f",&planta.PID[i].Td);
  fscanf(obert,"%*[^-0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%d",&planta.PID[i].activo);
  fscanf(obert,"%*[^-0123456789]");
  for (i=3;i<6;i++) fscanf(obert,"%f",&planta.mA[i]);
  fscanf(obert,"%*[^-0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%f",&corr[i].a);
  fscanf(obert,"%*[^-0123456789]");
  for (i=0;i<3;i++) fscanf(obert,"%f",&corr[i].b);
  si=fscanf(obert,"%*[^0123456789]");
  if (si==EOF)         /*Si hay menos datos de los necesarios*/
    {
    printf("\nError:  Faltan datos en el fichero de configuraci¢n depura.dat \n");
    fclose(obert);
    exit(1);
    }

fclose(obert);
  }
else
  {
  printf("\nError:  El fichero de configuraci¢n depura.dat no existe\n");
  exit (1);
  }

return;
}

/*************************************************************************/
/* Fichero para leer el perfil de alimentaci¢n en alimento.dat           */

void lectura_alimentacion(void)
{
FILE *obert;
```



```
char linea[150];
int i;
int si;

if ( access("c:\\depura\\alimento.dat",0) == 0)
  {
  obert=fopen("c:\\depura\\alimento.dat","rt");

  /*En primer lugar se lee la l¡nea y despu,s se traduce a n£meros*/

  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &planta.perfil);
  for (i=0;i<3;i++)
    fgets(linea,sizeof(linea),obert);

  for (i=0;i<24;i++) fscanf(obert,"%d",&planta.alimento[i].nitrogeno);
  fscanf(obert,"%*[^-0123456789]");
  for (i=0;i<24;i++) fscanf(obert,"%d",&planta.alimento[i].carbono);
  fscanf(obert,"%*[^-0123456789]");
  for (i=0;i<24;i++) si=fscanf(obert,"%f",&planta.alimento[i].agua);

  if (si==EOF)         /*Si hay menos datos de los necesarios*/
    {
    printf("\nError:  Faltan datos en el fichero de configuraci¢n alimento.dat \n");
    fclose(obert);
    exit(1);
    }

fclose(obert);
  }
else
  {
  printf("\nError:  El fichero de configuraci¢n alimento.dat no existe\n");
  exit (1);
  }

return;
}

/*************************************************************************/
/* Guarda los valores de los datos de la estructura "medidas" en
   el fichero depura.dat. Si es lunes a las ocho de la ma¤ana, llama a
   fichero semanal, que renombra depura.dat a la fecha actual.      */


void archivar_datos_medida(void)
{
   FILE *stream;
   int i;
   float dias=0;
   struct _dosdate_t d;
   struct  tm *t;
   time_t ltime;

   time (&ltime);
   t = localtime( &ltime );
   dias=dia_absoluto();

   stream = fopen("c:\\datos\\depura.dat", "a+");
   fprintf(stream,"%5.4f    ", dias);

   for (i=0;i<3;i++)    /*Escribimos el promedio de los £ltimos 3 minutos*/
       {
       fprintf(stream, " %4.2f",mat_graf_e2[i][0][479]);      /*pH*/
       fprintf(stream, " %4.1f",mat_graf_e2[i][1][479]);      /*T*/
       fprintf(stream, " %4.0f",mat_graf_e2[i][2][479]);      /*mV*/
       fprintf(stream, " %4.1f",mat_graf_e2[i][3][479]);      /*O2*/
       fprintf(stream, "    ");
       }
         fprintf(stream, " %4.2f",mat_graf_e2[0][4][479]);
         fprintf(stream, " %4.2f",mat_graf_e2[1][4][479]);
         fprintf(stream, " %4.2f",mat_graf_e2[2][4][479]);
   fprintf(stream, "    ");


   fprintf(stream, "%2d %02d %02d  ", t->tm_hour, t->tm_min, t->tm_sec);
   fprintf(stream,"%s", ctime( &ltime ));
```

```
   fclose(stream);

time (&ltime);
t = localtime( &ltime );

_dos_getdate(&d);
if ((t->tm_hour==8) && (t->tm_min<=9) && (d.dayofweek==1)) fichero_semanal();

return;
}

/*************************************************************************/
/* Renombra depura.dat a la fecha actual, comprobando primero que el
   fichero no exista ya. Tambi¡n copia el fichero de seguridad de las
   gr ficas a otro fichero con la fecha actual.                          */


void fichero_semanal(void)
{
char fitxer[25];
struct tm *time_ara;
time_t segns_ara;

tzset();
time(&segns_ara);
time_ara = localtime(&segns_ara);

strftime( fitxer, 21, "c:\\datos\\%d%b%y.dat", time_ara);

if ( access( fitxer, 0) == 0) return;

rename("c:\\datos\\depura.dat",fitxer);

strftime( fitxer, 21, "c:\\datos\\%d%b%y.gra", time_ara);
if ( access( fitxer, 0) == 0) return;
system("copy c:\\depura\\tmp\\sec-graf.dat c:\\datos\\sec-graf.dat > copy.tmp /y");
remove ("copy.tmp");
rename("c:\\datos\\sec-graf.dat",fitxer);

return;
}


/*************************************************************************/
/* Guarda el mensaje pasado como argumento en el fichero depura.dat,
   a¤adiendo la hora y la fecha.                                         */


void guarda_mensaje( char *mensaje)
{
   FILE *stream;
   time_t ltime;

   time (&ltime);

   stream = fopen("c:\\datos\\depura.dat", "a+");

   fprintf(stream,"%s", mensaje);
   fprintf(stream,"%s", ctime( &ltime ));

   fclose(stream);

return;
}

/*************************************************************************/

void guarda_medida( char *mensaje, char *fichero)
{
FILE *stream;
struct tm *t;
time_t ltime;

   time (&ltime);
   t = localtime( &ltime );
```



```
   stream = fopen( fichero, "a+");
   fprintf(stream, "%s", mensaje);
   fprintf(stream, "   %2d %02d %02d  \t", t->tm_hour, t->tm_min, t->tm_sec);
   fprintf(stream, "%s", ctime(&ltime));

   fclose(stream);

return;
}

/*************************************************************************/
/*    Subrutina de seguridad para los datos de las gr ficas y de medidas.
      Con orden=0, lee los datos; con orde=1 los sobreescribe.           */


void seguro_graficas(int orden)
{
FILE *obert;

/* Si comienza el programa y existe el fichero de seguridad, lo lee */
if (orden==0 && access("c:\\depura\\tmp\\sec-graf.dat",0) == 0)
  {
  obert=fopen("c:\\depura\\tmp\\sec-graf.dat","rb");

  fread(&hora, sizeof(hora), 1, obert);
  fread(&dia, sizeof(dia), 1, obert);

  fread(&mat_graf_e1,sizeof(mat_graf_e1),1,obert);
  fread(&mat_graf_e2,sizeof(mat_graf_e2),1,obert);
  fread(&mat_graf_e3,sizeof(mat_graf_e3),1,obert);

  fread(&tiempo_e1,sizeof(tiempo_e1),1,obert);
  fread(&tiempo_e2,sizeof(tiempo_e2),1,obert);
  fread(&tiempo_e3,sizeof(tiempo_e3),1,obert);

  fread(&medidas,sizeof(medidas),1,obert);

  fclose(obert);
  }

/* Sobreescribe el fichero con datos actuales */
if (orden==1)
  {
  obert=fopen("c:\\depura\\tmp\\sec-graf.dat","wb");

  fwrite(&hora, sizeof(hora), 1, obert);
  fwrite(&dia, sizeof(dia), 1, obert);

  fwrite(&mat_graf_e1,sizeof(mat_graf_e1),1,obert);
  fwrite(&mat_graf_e2,sizeof(mat_graf_e2),1,obert);
  fwrite(&mat_graf_e3,sizeof(mat_graf_e3),1,obert);

  fwrite(&tiempo_e1,sizeof(tiempo_e1),1,obert);
  fwrite(&tiempo_e2,sizeof(tiempo_e2),1,obert);
  fwrite(&tiempo_e3,sizeof(tiempo_e3),1,obert);

  fwrite(&medidas,sizeof(medidas),1,obert);

  fclose(obert);

  }
}

/*************************************************************************/
/* Funci¢n que guarda los valores de ox¡geno y mA en un fichero, con la
   hora actual.                                                          */

void archivar_datos_escalon(void)
{
FILE *fichero;
   int i=0;
   struct  tm *t;
   time_t ltime;

fichero=fopen("c:\\datos\\escalon.dat","at+");

time (&ltime);
```

```
t = localtime( &ltime );

for (i=0;i<3;i++) fprintf(fichero, " %4.2f ",medidas[i].mA);
for (i=0;i<3;i++) fprintf(fichero, " %4.2f ",medidas[i].oxigeno);

fprintf(fichero, "   %2d %02d %02d  ", t->tm_hour, t->tm_min, t->tm_sec);
fprintf(fichero,"\n");

fclose (fichero);
}

/*****************************************************************************/
/* Código necesario si desean hacerse pruebas.
    Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST
float dia_absoluto(void)
{
    return 0.0F;
}

void main(void)
{
/*lectura_configuracion();*/
/*archivar_datos_escalon();
archivar_datos_medida();*/
lectura_alimentacion();
}
#endif
/*****************************************************************************/
```



```
/*****************************************************************************

   GRAFICOS.C -        FUNCIONES PARA LA PRESENTACION DE DATOS Y GRAFICAS
                POR PANTALLA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 10/10/96

*****************************************************************************/

/********************** Librerias  a incluir ************************/

#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <graph.h>
#include <math.h>
#include <time.h>
#include "nombres.h"

/************************ Definiciones ********************************/
/*#define TEST*/
#ifdef TEST
#define extern
#endif

#define PATHFONTS "c:\\depura\\fonts\\*.fon"

/********************** Subrutinas del fichero **********************/

void modo_grafico(void);
void graf_colorines(int orden);
void graf_base(void);
void graf_plc(void);
void graf_led( int led, int on_off );
void graf_nivel(int nivel, int on_off);
void graf_reloj(void);
void graf_hora(void);
void graf_sombra_grafica(void);
void graf_ayuda_general(void);
void graf_ayuda_graficas(void);
void graf_borra_ayuda(void);
void graf_display(void);
void graf_valores_display(void);
void salir_al_dos(void);
void graf_mA(float *mA);
void actualizar_matrices_graficas(void);
void graf_ejes_tiempo(int escala);
void graf_eje_pH(void);
void graf_eje_T(void);
void graf_eje_mV(void);
void graf_eje_O2(void);
void graf_eje_mA(void);
void graf_eje_multiple(void);
void graf_eje_debug(void);
void graf_tanques(void);
void graf_tanque(int tanque);
void grafica_variable(int variable, int tanque, int escala, short color);
void graf_actualizacion_grafico(int variable, int escala);
void graf_nuevo_grafico( int variable, int escala);
void graf_presentacion(void);
void graf_mensaje(char *mensaje);

/****************** Variables comunes del programa ******************/


float mat_graf_e1[3][6][360]; /*Matrices para las gr ficas*/
float mat_graf_e2[3][6][480];
float mat_graf_e3[3][6][480];
time_t tiempo_e1[360];        /*Matrices tiempo gr ficas*/
time_t tiempo_e2[480];
time_t tiempo_e3[480];


int hora,dia;
```

```
extern primera medidas[3];
extern estado_planta planta;

/*************************************************************************/
/* Funci¢n para dar un poco de color a la vida en una esquina.           */


void graf_colorines(int orden)
{
int i,k;

switch(orden)
{
    case 0:
       for (i=116;i>20;i--)
       {
           for (k=40;k>0;k--)
           {
           _setcolor( (short)(i+k) );
           _moveto( (short)(600+k), 430 );
           _lineto( (short)(600+k), 480 );
           }
       }
    break;

    case 1:
        for (i=20;i<116;i++)
        {
            for (k=0;k<40;k++)
            {
           _setcolor( (short)(i+k) );
           _moveto( (short)(600+k), 430 );
           _lineto( (short)(600+k), 480 );
           }
        }
    break;
}

}

/*************************************************************************/
/*  Pasa del modo texto al gr fico.                                      */


void modo_grafico(void)
{
if( !_setvideomode( _MAXRESMODE ) )     exit( 1 );

_registerfonts( PATHFONTS ); /*Fuentes gr ficas*/

}

/*************************************************************************/
/* Funci¢n que llama a las funciones necesarias para dibujar la pantalla. */


void graf_base(void)
{

graf_plc();
graf_nivel( 1, 1);
graf_reloj();
graf_sombra_grafica();
/*graf_ayuda_graficas();*/
graf_ayuda_general();
graf_display();
graf_hora();
graf_nuevo_grafico( planta.variable_actual, planta.escala_actual);
}



/*************************************************************************/
/* Funci¢n para dibujar el recuadro y los c¡rculos inferiores de los leds
   indicadores del estado del PLC.                                       */

void graf_plc(void)
```



```
{
short i, x;
char fuente[20]="t'Tms Rmn'h10w50b";
char numero[3];

_setcolor(242);                                 /*Establece el color activo*/
_rectangle( _GFILLINTERIOR, 600, 0 ,639, 365);      /*Dibuja rect ngulo relleno*/
_setcolor(29);
_rectangle( _GBORDER, 600, 0 ,639, 365);     /*Dibuja rect ngulo s¢lo borde*/

_setcolor(7);
_setgtextvector( 1, 0 );                    /*Orientaci¢n del texto*/
_setfont( fuente );                         /*Selecci¢n de la fuente*/
for (i=0;i<24;i++)
     {
     if ( i>=10 ) x = 605;
     else x = 610;
     _moveto ( x, (short)(5+i*15) );      /*Se posciona en x-y indicado*/
     sprintf(numero, "%li",i);                  /*Escribe en el buffer el n$*/
     _outgtext(numero);                     /*Escribe el buffer en pantalla*/
     }
_setcolor(183);
for (i=0;i<24;i++)
    _ellipse( _GFILLINTERIOR, 620, (short) (i*15+5), 630, (short)(15+i*15) );
_setcolor(130);                                 /*Dibuja c¡rculo*/
for (i=0;i<24;i++)
    _ellipse( _GBORDER, 620, (short) (i*15+5), 630, (short)(15+i*15) );
                                                /*Dibuja circunferencia*/
}

/*************************************************************************/
/* Dibuja un led del color indicado, para indicar su estado.             */

void graf_led( int led, int on_off )
{
short x1= 621, x2=629;
short y1, y2;

if ( led<0 || led>23 ) return;
if ( on_off!=0 && on_off!=1 && on_off!=2 ) return;

y1 = (short) (led*15+6);
y2 = (short) (led*15+14);

switch(on_off)
{
    case 0:           /*Apagado*/
       _setcolor(183);
    break;
    case 1:           /*Encendido*/
       _setcolor(4);
    break;
    case 2:           /*Error comunicaci¢n*/
       _setcolor(0);
    break;
}

_ellipse( _GFILLINTERIOR, x1, y1, x2, y2 );

}

/*************************************************************************/
/* Dibuja el indicador del estado del tanque 3, es decir si se est  vaciando
   o llenando demasiadas veces seguidas. Tambi¡n indica si existe alguna
   alarma de funcionamiento.                                             */


void graf_nivel(int nivel, int on_off)
{
char fuente[20]="t'Tms Rmn'h10w50b";

_setcolor(242);
_rectangle( _GFILLINTERIOR, 600, 370 ,639, 420);
_setcolor(29);
_rectangle( _GBORDER, 600, 370 ,639, 420);
_setcolor(24);
_rectangle( _GFILLINTERIOR, 615, 375 ,634, 415);
```

```
_setcolor(7);
_setfont( fuente );
_moveto ( 603, 407 );
_setgtextvector( 0, 1 );
_outgtext("Nivel");

_moveto ( 615, 400 );
_lineto ( 634, 400 );

_setcolor(187);
switch(on_off)
{
    case 0:    /*Bomba apagada*/
       _rectangle( _GFILLINTERIOR, 615, 401 ,634, (short)(400+nivel/2) );
    break;
    case 1:    /*Bomba encendida*/
       _rectangle( _GFILLINTERIOR, 615, 399 ,634, (short)(400-nivel/2) );
    break;
    case -1:   /*Alarma de nivel, Tanques 1 ¢ 2*/
       _setcolor(4);
       _rectangle( _GFILLINTERIOR, 615, 375 ,634, 415);
       _setcolor(7);
       _moveto ( 620, 412 );
       _outgtext("A.Tanc");
    break;
    case -2:   /*Alarma de nivel, bomba*/
       _setcolor(6);
       _rectangle( _GFILLINTERIOR, 615, 375 ,634, 415);
       _setcolor(7);
       _moveto ( 620, 415 );
       _outgtext("A.Bomba");
    break;
    case -3:   /*Alarma escape*/
       _setcolor(12);
       _rectangle( _GFILLINTERIOR, 615, 375 ,634, 415);
       _setcolor(7);
       _moveto ( 620, 415 );
       _outgtext("A.Escape");
    break;
    case -4:   /*Alarma manual*/
       _setcolor(110);
       _rectangle( _GFILLINTERIOR, 615, 375 ,634, 415);
       _setcolor(7);
       _moveto ( 620, 415 );
       _outgtext("A.Manual");
    break;
}
}


/*************************************************************************/
/* Gr fica del indicador de detecci¢n de sobrecargas, el reloj que da
   vueltas en la esquina inferior derecha. Es llamada una vez cada ciclo
   del do while del programa principal.                                  */


void graf_reloj(void)
{
short x, y;
short radio=19;
double alfa;
static int i=0;

i++;
if (i>=180) i=0;

_setcolor(242); /*antes 171*/
_rectangle( _GFILLINTERIOR, 601, 426, 638, 473);
_setcolor(29);
_rectangle( _GBORDER, 600, 425 ,639, 474);

alfa= (double) (2*3.14159265*i*2./360.);
x= (short) (619 - radio * sin(alfa));
y= (short) (449 + radio * cos(alfa));

/*_setcolor((short) i);*/
```



```
_setcolor(31);
_moveto ( 619, 449);
_lineto ( x, y);
_lineto ( 618, 450);

}

/*************************************************************************/
/* Reloj digital, con fecha y hora.                                      */

void graf_hora(void)
{
time_t ltime;
short x=445,
      y=460;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[30];

_setgtextvector( 1, 0 );
_setfont( fuente );

time (&ltime);
sprintf(buffer,"%s", ctime( &ltime ));

_setcolor(242);
_rectangle( _GFILLINTERIOR, 441, 459 ,587, 471);
_setcolor(28);
_rectangle( _GBORDER, 441, 459 ,587, 471);

_setcolor(29);
_moveto(x,y);
_outgtext(buffer);

}

/*************************************************************************/
/* Escribe la pantalla de ayuda general del programa.                    */

void graf_ayuda_general(void)
{
int i;
short x,y;
char fuente[20]="t'Tms Rmn'h8w5b";
char buffer[15][28]={/************************/
                     "F1   Ayuda gr ficas",
                     "F2   Variar mA PCL726",
                     "F3   Encender PLC 1-4",
                     "F4   Apagar PLC 1-4",
                     "F5   Encender agitador",
                     "F6   Apagar agitador",
                     "F7   Parada emerg. agit.",
                     "F8   Variar vel. agit.",
                     "F9   Encender salida PLC",
                     "F10  Salir del programa",
                     "F11  Reinicializar ",
                     "F12  Parada manual ",
                     "ALT-Q  Salida + parada",
                     "ALT-D  Salida al DOS",
                     "ESC  Cr,ditos programa"};

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(242);/*antes 19*/
_rectangle( _GFILLINTERIOR, 0, 0, 595, 38);
_setcolor(29);
_rectangle( _GBORDER, 0, 0, 595, 38);

for (i=0;i<15;i++)
    {
    x=(short)(5+i*120-(int)(i/5)*600);
    y=(short)(2+12*(int)(i/5));
    _moveto(x,y);
    _outgtext(buffer[i]);
    }
}
```

```
/*************************************************************************/
/* Escribe la pantalla de ayuda referida a las gr ficas.                 */

void graf_ayuda_graficas(void)
{
int i;
short x,y;
char fuente[20]="t'Tms Rmn'h8w5b";
char buffer[15][28]={/************************/
                     "F1  Ayuda general",
                     "O   Gr fica  Ox;geno",
                     "H   Gr fica  pH",
                     "R   Gr fica  Redox",
                     "T   Gr fica  Temperatura",
                     "A   Gr fica  mA",
                     "1   Variables  tanque 1",
                     "2   Variables  tanque 2",
                     "3   Variables  tanque 3",
                     "E   Cambio  de  escala",
                     "ESPACIO     Actualizar",
                     "D   Debug programa"};

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(171);
_rectangle( _GFILLINTERIOR, 0, 0, 595, 38);
_setcolor(30);
_rectangle( _GBORDER, 0, 0, 595, 38);

for (i=0;i<15;i++)
    {
    x=(short)(5+i*120-(int)(i/5)*600);
    y=(short)(2+12*(int)(i/5));
    _moveto(x,y);
    _outgtext(buffer[i]);
    }
}

/*************************************************************************/
/* Dibuja un rect ngulo negro para borrar el cuadro de ayuda, y as; poder
   escribir un mensaje en esa zona.                                      */

void graf_borra_ayuda(void)
{
_setcolor(0);
_rectangle( _GFILLINTERIOR, 0, 0, 595, 38);
_setcolor(29);
_rectangle( _GBORDER, 0, 0, 595, 38);
}

/*************************************************************************/
/* Dibuja el rect ngulo, las rayas interiores y los t;tulos del display
   donde se muestran los valores.                                        */

void graf_display(void)
{
int i, j;
short x, y;

char fuente[20]="t'Tms Rmn'h8w5b";
char variables[6][12]={/**********/
                       "     pH   ",
                       "   T (C) ",
                       "Redox (mV)",
                       " O2 (ppm) ",
                       " mA pc1726",
                       "V (%) agit"};
char tanques[3][12]=  {/**********/
                       " Tanque 1 ",
                       " Tanque 2 ",
                       " Tanque 3 "};

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(171);
```



```
_rectangle( _GFILLINTERIOR, 0, 41, 595, 97);
_setcolor(29);
_rectangle( _GBORDER, 0, 41, 595, 97);

for (i=1;i<7;i++)
    {
    _setcolor(30);
    x=(short)(3+i*70);
    y=(short)(45);
    _rectangle( _GBORDER, x, y, (short)(x+70), (short)(y+12));
    _setcolor(10);
    _moveto((short)(x+10),y);
    _outgtext(variables[i-1]);
    }
for (j=1;j<4;j++)
    {
    _setcolor(30);
    x=(short)(3);
    y=(short)(45+j*12);
    _rectangle( _GBORDER, x, y, (short)(x+70), (short)(y+12));
    _setcolor(10);
    _moveto((short)(x+10),y);
    _outgtext(tanques[j-1]);
    }

graf_valores_display();
}

/*************************************************************************/
/* Escribe los valores de las variables medidas del sistema en el display.*/

void graf_valores_display(void)
{
int i, j;
short x, y;
char fuente[20]="t'Tms Rmn'h8w5b";
char buffer[6][10];

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(0);
_rectangle( _GFILLINTERIOR, 73, 57, 493, 93);

_setcolor(242);
for (j=1;j<4;j++)
{
  for (i=1;i<7;i++)
    {
    x=(short)(3+i*70);
    y=(short)(45+j*12);
    _rectangle( _GBORDER, x, y, (short)(x+70), (short)(y+12));
    }
}

_setcolor(30);
_rectangle( _GBORDER, 73, 57, 493, 93);


_setcolor(30);
for (i=0;i<3;i++)
   {
   sprintf (buffer[0], "%2.2f", mat_graf_e1[i][0][359]);
   sprintf (buffer[1], "%2.1f", mat_graf_e1[i][1][359]);
   sprintf (buffer[2], "%4.0f", mat_graf_e1[i][2][359]);
   sprintf (buffer[3], "%2.1f", mat_graf_e1[i][3][359]);
   sprintf (buffer[4], "%2.2f", mat_graf_e1[i][4][359]);
   sprintf (buffer[5], "%2.1f", mat_graf_e1[i][5][359]);
   for (j=0;j<6;j++)
     {
     _moveto((short)(100+j*70), (short)(58+i*12));
     _outgtext(buffer[j]);
     }
   }
}
/*************************************************************************/
/* Dibuja la sombra de la gr fica, la pantalla base.                     */
```

```
void graf_sombra_grafica(void)
{
_setcolor(23);              /*sombra*/
_rectangle( _GFILLINTERIOR, 5, 105, 595, 479);
_setcolor(171);
_rectangle( _GFILLINTERIOR, 0, 100, 590, 474);
_setcolor(29);
_rectangle( _GBORDER, 0, 100 , 590, 474);
_setcolor(0);
_rectangle( _GFILLINTERIOR, 80, 115 , 560, 434);
_setcolor(29);
_rectangle( _GBORDER, 80, 115 , 560, 434);
}


/*************************************************************************/
/* Funci¢n para salir al dos. Cambio de modo gr fico a texto, ejecuta un
   command y despu,s redibuja la pantalla, leds incluidos.          */

void salir_al_dos(void)
{
int i;

_setvideomode( _DEFAULTMODE );
printf("\nEscribe 'exit' para volver al programa\n");
system("COMMAND.COM");
if( !_setvideomode( _MAXRESMODE ) )     exit( 1 );

graf_base();
for (i=0;i<24;i++) graf_led( i, planta.salidas_plc[i]);

}

/*************************************************************************/
/* Funci¢n temporal utilizada para mostrar los valores de aire introducidos
   en los tanques.                                                  */

void graf_mA(float *mA)
{
char fuente[20]="t'Tms Rmn'h14w50b";
char buffer[80];

sprintf(buffer,"mA:   %5.2f   %5.2f   %5.2f",mA[0],mA[1],mA[2]);

_setcolor(0);
_rectangle( _GFILLINTERIOR, 0, 10, 400, 30);
_setcolor(7);

_setgtextvector( 1, 0 );
_setfont( fuente );
_moveto(0,10); /*arreglo temporal*/
_outgtext(buffer);

}
/*************************************************************************/
/* Actualiza las matrices utilizadas para dibujar las gr ficas.
   mat_graf_e1[3][6][360]; datos de una hora, cada 10 segundos
   mat_graf_e2[3][6][480]; datos de un d¡a, cada 3 minutos
   mat_graf_e3[3][6][480]; datos de una semana, cada 21 minutos
   Los datos de cada matriz est n promediados de las anteriores.
   18 de _e1 -> 1 de _e2.  7 de _e2 -> 1 de _e3                  */


void actualizar_matrices_graficas(void)
{
int sonda,i,j;
float sum;
double temp;

for (sonda=0;sonda<3;sonda++)
  for (i=0;i<6;i++)
    for (j=0;j<359;j++) mat_graf_e1[sonda][i][j]=mat_graf_e1[sonda][i][j+1];
for (j=0;j<359;j++) tiempo_e1[j]=tiempo_e1[j+1];
                            /*Hacemos sitio para el £ltimo valor de _e1*/

for (sonda=0;sonda<3;sonda++)
```



```
  {
  mat_graf_e1[sonda][0][359]=medidas[sonda].ph;
  mat_graf_e1[sonda][1][359]=medidas[sonda].temperatura;
  mat_graf_e1[sonda][2][359]=medidas[sonda].potencial;
  mat_graf_e1[sonda][3][359]=medidas[sonda].oxigeno;
  mat_graf_e1[sonda][4][359]=medidas[sonda].mA;
  mat_graf_e1[sonda][5][359]=medidas[sonda].velocidad;
  }                          /*Asignamos el £ltimo valor*/
tiempo_e1[359]=time(NULL);

hora++;
if (hora>=18)
  {
  hora=0;

  for (sonda=0;sonda<3;sonda++)
    for (i=0;i<6;i++)
      for (j=0;j<479;j++) mat_graf_e2[sonda][i][j]=mat_graf_e2[sonda][i][j+1];
                                 /*Hacemos sitio para el £ltimo valor de _e2*/
  for (j=0;j<479;j++) tiempo_e2[j]=tiempo_e2[j+1];

  for (sonda=0;sonda<3;sonda++)
    for (i=0;i<6;i++)
      {
      sum=0;
      for (j=342;j<360;j++) sum=sum+mat_graf_e1[sonda][i][j];
      mat_graf_e2[sonda][i][479]=sum/18.0F;
      }                          /*Promediamos y asignamos el £ltimo valor*/
  temp=0.0F;
  for (j=342;j<360;j++) temp=temp+tiempo_e1[j];
  tiempo_e2[479]=(long int)(temp/18.0F);
  dia++;

  if (dia>=7)
    {
    dia=0;

    for (sonda=0;sonda<3;sonda++)
      for (i=0;i<6;i++)
        for (j=0;j<479;j++) mat_graf_e3[sonda][i][j]=mat_graf_e3[sonda][i][j+1];
    for (j=0;j<479;j++) tiempo_e3[j]=tiempo_e3[j+1];
                                 /*Hacemos sitio para el £ltimo valor de _e3*/

    for (sonda=0;sonda<3;sonda++)
      for (i=0;i<6;i++)
        {
        sum=0;
        for (j=473;j<480;j++) sum=sum+mat_graf_e2[sonda][i][j];
        mat_graf_e3[sonda][i][479]=sum/7.0F;
        }                        /*Promediamos y asignamos el £ltimo valor*/
    temp=0.0F;
    for (j=473;j<480;j++) temp=temp+tiempo_e2[j];
    tiempo_e3[479]=(long int)(temp/7.0F);
    }/*if (dia>=7)*/

  }/*if (hora>=18)*/

}

/*************************************************************************/
/* Dibuja el eje horizontal del tiempo, en tres escalas diferentes.     */

void graf_ejes_tiempo(int escala)
{
int i;
short x;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[20];

_setgtextvector( 1, 0 );
_setfont( fuente );
_setcolor (30);

switch(escala)
  {
  case 1:                          /*Escala de una hora*/
     for (i=0;i<13;i++)
```

```
      {
      x=(short)(80+i*40);
      _moveto(x,434);
      _lineto(x,439);
      _moveto((short)(x-5),441);
      sprintf(buffer, "%2i",i*5);
      _outgtext(buffer);
      }
    _moveto(150,460);
    _outgtext("Tiempo  (minutos)");
  break;
  case 2:                          /*Escala de un d¡a*/
    for (i=0;i<25;i++)
      {
      x=(short)(80+i*20);
      _moveto(x,434);
      _lineto(x,439);
      _moveto((short)(x-5),441);
      sprintf(buffer, "%2i",i);
      _outgtext(buffer);
      }
    _moveto(150,460);
    _outgtext("Tiempo  (horas)");
  break;
  case 3:                          /*Escala de una semana*/
    for (i=0;i<8;i++)
      {
      x=(short)(80+i*68.58F);
      _moveto(x,434);
      _lineto(x,439);
      _moveto((short)(x-5),441);
      sprintf(buffer, "%2i",i);
      _outgtext(buffer);
      }
    _moveto(150,460);
    _outgtext("Tiempo  (d¡as)");
  break;
  }

/*
_rectangle( _GBORDER, 80, 115 , 560, 434);
*/
}
/*************************************************************************/
/* Dibuja el eje vertical del pH.                                        */

void graf_eje_pH(void)
{
int i;
short y;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[20];

_setcolor (30);
_setgtextvector( 1, 0 );
_setfont( fuente );

for (i=0;i<5;i++)    /*Escala 5 a 9*/
  {
  y=(short)(434-i*79.75F);
  _moveto(75,y);
  _lineto(80,y);
  _moveto(65,(short)(y-5));
  sprintf(buffer, "%2i",i+5);
  _outgtext(buffer);
  }
_setgtextvector( 0, 1);
_moveto(30,270);
_outgtext("pH");
}

/*************************************************************************/
/* Dibuja el eje vertical de la Temperatura.                             */

void graf_eje_T(void)
{
int i;
```



```
short y;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[20];

_setcolor (30);
_setgtextvector( 1, 0 );
_setfont( fuente );

for (i=0;i<6;i++)    /*Escala 10 a 35*/
  {
  y=(short)(434-i*63.8F);
  _moveto(75,y);
  _lineto(80,y);
  _moveto(65,(short)(y-5));
  sprintf(buffer, "%2i",(i*5+10));
  _outgtext(buffer);
  }
_setgtextvector( 0, 1);
_moveto(30,300);
_outgtext("Temperatura (C)");
}


/*************************************************************************/
/* Dibuja el eje vertical del redox.                                     */

void graf_eje_mV(void)
{
int i;
short y;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[20];

_setcolor (30);
_setgtextvector( 1, 0 );
_setfont( fuente );

for (i=0;i<11;i++)    /*Escala -600 a 400*/
  {
  y=(short)(434-i*31.9F);
  _moveto(75,y);
  _lineto(80,y);
  _moveto(55,(short)(y-5));
  sprintf(buffer, "%4i",(i*100-600));
  _outgtext(buffer);
  }
_setgtextvector( 0, 1);
_moveto(30,290);
_outgtext("Redox (mV)");
}

/*************************************************************************/
/* Dibuja el eje vertical del ox¡geno.                                   */

void graf_eje_O2(void)
{
int i;
short y;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[20];

_setcolor (30);
_setgtextvector( 1, 0 );
_setfont( fuente );

for (i=0;i<9;i++)  /*Escala 0 a 8*/
  {
  y=(short)(434-i*39.87F);
  _moveto(75,y);
  _lineto(80,y);
  _moveto(65,(short)(y-5));
  sprintf(buffer, "%2i",i);
  _outgtext(buffer);
  }
_setgtextvector( 0, 1);
_moveto(30,300);
_outgtext("Ox¡geno (ppm)");
```

```
}


/*************************************************************************/
/* Dibuja el eje vertical de los miliamperios de la v lvula de aire.     */

void graf_eje_mA(void)
{
int i;
short y;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[20];

_setcolor (30);
_setgtextvector( 1, 0 );
_setfont( fuente );

for (i=0;i<17;i++)  /*Escala 4 a 20*/
  {
  y=(short)(434-i*19.93F);
  _moveto(75,y);
  _lineto(80,y);
  _moveto(65,(short)(y-5));
  sprintf(buffer, "%2i",i+4);
  _outgtext(buffer);
  }
_setgtextvector( 0, 1);
_moveto(30,340);
_outgtext("Abertura v lvulas de aire (mA)");

}

/*************************************************************************/
/* Dibuja el eje vertical m£ltiple de cada tanque: pH, T, mV, O2, mA.    */

void graf_eje_multiple(void)
{
int i;
short y;
char fuente[20]="t'Tms Rmn'h8w50b";
char buffer[20];

_setcolor (40);
_setgtextvector( 0, 1 );
_setfont( fuente );

_moveto(16,115);
_lineto(16,434);
for (i=0;i<5;i++)    /*pH escala 5 a 9*/
  {
  y=(short)(434-i*79.75F);
  _moveto(14,y);
  _lineto(16,y);
  _moveto(4,(short)(y+5));
  sprintf(buffer, "%2i",i+5);
  _outgtext(buffer);
  }
_moveto(6,471);
_outgtext("pH");

_setcolor (44);
_moveto(32,115);
_lineto(32,434);
for (i=0;i<6;i++)    /*T escala 10 a 35*/
  {
  y=(short)(434-i*63.8F);
  _moveto(30,y);
  _lineto(32,y);
  _moveto(20,(short)(y+5));
  sprintf(buffer, "%2i",(i*5+10));
  _outgtext(buffer);
  }
_moveto(22,471);
_outgtext("T(C)");

_setcolor (48);
```



```
_moveto(48,115);
_lineto(48,434);
for (i=0;i<11;i++)    /*mV escala -600 a 400*/
  {
  y=(short)(434-i*31.9F);
  _moveto(46,y);
  _lineto(48,y);
  _moveto(36,(short)(y+10));
  sprintf(buffer, "%4i",(i*100-600));
  _outgtext(buffer);
  }
_moveto(38,471);
_outgtext("mV");

_setcolor (52);
_moveto(64,115);
_lineto(64,434);
for (i=0;i<9;i++)  /*O2 escala 0 a 8*/
  {
  y=(short)(434-i*39.87F);
  _moveto(62,y);
  _lineto(64,y);
  _moveto(52,(short)(y+5));
  sprintf(buffer, "%2i",i);
  _outgtext(buffer);
  }
_moveto(54,471);
_outgtext("ppm");

_setcolor (56);
_moveto(80,115);
_lineto(80,434);
for (i=0;i<17;i++)  /*mA  escala 4 a 20*/
  {
  y=(short)(434-i*19.93F);
  _moveto(78,y);
  _lineto(80,y);
  _moveto(68,(short)(y+5));
  sprintf(buffer, "%2i",i+4);
  _outgtext(buffer);
  }
_moveto(70,471);
_outgtext("mA");
}

/*************************************************************************/
/* Dibuja el eje de debug.                                               */

void graf_eje_debug(void)
{
char fuente[20]="t'Tms Rmn'h12w50b";

_setcolor (30);
_setfont( fuente );

_setgtextvector( 0, 1);
_moveto(30,350);
_outgtext("Seguimiento del programa");
}

/*************************************************************************/
/* Dibuja el recuadro donde se indica el color correspondiente
   a cada tanque.                                                        */

void graf_tanques(void)
{
int j;
short x=3,
      y;
char fuente[20]="t'Tms Rmn'h10w50b";
char buffer[20];

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(0);
_rectangle(_GFILLINTERIOR,3,434,53,468);
```

```
for (j=1;j<4;j++)
    {
    _setcolor(30);
    y=(short)(422+j*12);
    _rectangle( _GBORDER, x, y, (short)(x+50), (short)(y+12));
    _setcolor((short)(j*8+44));
    _moveto((short)(x+6),y);
    sprintf(buffer,"Tanque %1i",j);
     _outgtext(buffer);
    }
}


/*************************************************************************/
/* Dibuja el nombre del tanque al que corresponde la gr fica m£ltiple.     */

void graf_tanque(int tanque)
{
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[30];

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(0);
_rectangle( _GFILLINTERIOR, 300, 459 ,370, 471);
_setcolor(30);
_rectangle( _GBORDER, 300, 459 ,370, 471);

_moveto(310 ,460);
sprintf(buffer,"Tanque %1i",tanque);
_setcolor(51);
_outgtext(buffer);
}

/*************************************************************************/
/* Grafica las l¡neas correspondientes para una variable de un tanque
   determinado, en la escala de tiempo y con el color proporcionado.       */

void grafica_variable(int variable, int tanque, int escala, short color)
{
int i;
short x,y,xt;
float x0=80.0F,
      y0=433.0F,
      alto=317.0F;
float sets[6][3]={/*m¡nimo, m ximo, rango*/
                  {5.0F,9.0F,4.0F},          /*pH*/
                  {10.0F,35.0F,25.0F},       /*Temperatura*/
                  {-600.0F,400.0F,1000.0F},  /*mV*/
                  {0.0F,8.0F,8.0F},          /*O2*/
                  {4.0F,20.0F,16.0F},        /*mA*/
                  {0.0F,100.0F,100.0F}};     /*Velocidad*/
double increm_t, increm_tt;

_setcolor(color);
switch(escala)
 {
 case 1:                                /*Escala para una hora*/
   increm_tt=60.0F*60.0F; /*1 hora, 60 minutos*60 segundos*/
   x=(short)(x0+1);
   y=(short)(y0-alto/sets[variable][2]*
       (mat_graf_e1[tanque-1][variable][1]-sets[variable][0]) );
   if (y<116) y=116;
   if (y>433) y=433;
   _moveto(x,y);
   for (i=2;i<360;i++)
      {
      increm_t=(double) difftime(tiempo_e1[359],tiempo_e1[i]);
      if (increm_t>increm_tt) continue;
      xt=(short)(559-increm_t/increm_tt*478.0F);
      if ((xt-x)>10) _setlinestyle(0xAAAA);
      else  _setlinestyle(0xFFFF);
      x=xt;
      y=(short)(y0-alto/sets[variable][2]*
          (mat_graf_e1[tanque-1][variable][i]-sets[variable][0]) );
```



```
      if (y<116) y=116;
      if (y>433) y=433;
      _lineto(x,y);
      }
 break;
 case 2:                                /*Escala para un d¡a*/
   increm_tt=24.0F*60.0F*60.0F; /*24 horas* 60 minutos*60 segundos*/
   x=(short)(x0+1);
   y=(short)(y0-alto/sets[variable][2]*
       (mat_graf_e2[tanque-1][variable][1]-sets[variable][0]) );
   if (y<116) y=116;
   if (y>433) y=433;
   _moveto(x,y);
   for (i=2;i<480;i++)
      {
      increm_t=(double) difftime(tiempo_e2[479],tiempo_e2[i]);
      if (increm_t>increm_tt) continue;
      xt=(short)(559-increm_t/increm_tt*478.0F);
      if ((xt-x)>10) _setlinestyle(0xAAAA);
      else  _setlinestyle(0xFFFF);
      x=xt;
      y=(short)(y0-alto/sets[variable][2]*
          (mat_graf_e2[tanque-1][variable][i]-sets[variable][0]) );
      if (y<116) y=116;
      if (y>433) y=433;
      _lineto(x,y);
      }
 break;
 case 3:                                /*Escala para una semana*/
   increm_tt=7.0F*24.0F*60.0F*60.0F;/*7 d¡as*24 horas*60 minutos*60 segundos*/
   x=(short)(x0+1);
   y=(short)(y0-alto/sets[variable][2]*
       (mat_graf_e3[tanque-1][variable][1]-sets[variable][0]) );
   if (y<116) y=116;
   if (y>433) y=433;
   _moveto(x,y);
   for (i=2;i<480;i++)
      {
      increm_t=(double) difftime(tiempo_e3[479],tiempo_e3[i]);
      if (increm_t>increm_tt) continue;
      xt=(short)(559-increm_t/increm_tt*478.0F);
      if ((xt-x)>10) _setlinestyle(0xAAAA);
      else  _setlinestyle(0xFFFF);
      x=xt;
      y=(short)(y0-alto/sets[variable][2]*
          (mat_graf_e3[tanque-1][variable][i]-sets[variable][0]) );
      if (y<116) y=116;
      if (y>433) y=433;
      _lineto(x,y);
      }
 break;
 }
 _setlinestyle(0xFFFF);
}

/*************************************************************************/
/* Llama a la funci¢n grafica_variable las veces necesarias para dibujar
   todas las variables que aparecer n en el gr fico. El tratamiento es
   diferente si se trata de la gr fica de una variable para los tres
   tanques o de las variables de un tanque.                                */

void graf_actualizacion_grafico(int variable, int escala)
{
int i;

if (variable!=20)
    {
    _setcolor(0);
    _rectangle( _GFILLINTERIOR, 81, 116 , 559, 433);
    }

if (variable>=0 && variable<5)             /*Una variable de los tres tanques*/
  for (i=1;i<4;i++)   grafica_variable( variable, i, escala, (short)(i*8+44));

if (variable>=5 && variable<8)             /*Variables de un tanque*/
  for (i=0;i<5;i++)   grafica_variable( i, variable-4, escala, (short)(i*4+40));
```

```
}

/*************************************************************************/
/* Funci¢n con el copyright, presentaci¢n del programa.            */

void graf_presentacion(void)
{
char fuente1[20]="t'Tms Rmn'h60w50b";
char fuente2[20]="t'Tms Rmn'h16w50b";
char fuente3[20]="t'Tms Rmn'h12w50b";

_setgtextvector( 1, 0 );

_setcolor(104);
_rectangle( _GFILLINTERIOR, 81, 116 , 559, 433);

_moveto(250 ,150);
_setcolor(79);
_setfont( fuente1 );
_outgtext("DEPURA");
_moveto(160 ,200);
_setfont( fuente2 );
_outgtext("Programa per la Monitoritzaci¢ i Control");
_moveto(130 ,225);
_outgtext("de plantes depuradores d'aigues residuals urbanes");
_setcolor(77);
_moveto(100 ,362);
_setfont( fuente3 );
_outgtext("Copyright: Juan Baeza 1996");
_moveto(100 ,380);
_outgtext("Unitat d'Enginyer¡a Qu¡mica");
_moveto(100 ,398);
_outgtext("Universitat Aut*noma de Barcelona");

/* 80, 115 , 560, 434*/
}


/*************************************************************************/
/* Funci¢n que hace las llamadas necesarias a otras funciones, para dibujar
   la gr fica de la variable en la escala determinada.                */

void graf_nuevo_grafico( int variable, int escala)
{

graf_sombra_grafica();                    /*Sombra inferior*/
graf_hora();                              /*Reloj digital, hora actual*/
if (variable>=0 && variable<8) graf_ejes_tiempo(escala);
                                          /*Escala horizontal tiempo*/
if (variable>=0 && variable<5) graf_tanques();      /*Leyenda tanques*/
switch(variable)                          /*Eje/s vertical/es*/
   {
   case 0:
     graf_eje_pH();
   break;
   case 1:
     graf_eje_T();
   break;
   case 2:
     graf_eje_mV();
   break;
   case 3:
     graf_eje_O2();
   break;
   case 4:
     graf_eje_mA();
   break;
   case 5:
     graf_tanque(1);
     graf_eje_multiple();
   break;
   case 6:
     graf_tanque(2);
     graf_eje_multiple();
   break;
   case 7:
     graf_tanque(3);
```



```
     graf_eje_multiple();
   break;
   case 20:
     graf_eje_debug();
   break;
   }
graf_actualizacion_grafico(variable, escala);
}

/*************************************************************************/
/* Salida de un mensaje por pantalla gr fica */

void graf_mensaje(char *mensaje)
{
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[100];
static short yd=125;
short xd =100;

if (strlen(mensaje)>82) mensaje[81]=0;

_setgtextvector( 1, 0 );
_setfont( fuente );
if (yd>410)
    {
    _setcolor(0);
    _rectangle( _GFILLINTERIOR, 81, 116 , 559, 433);
    yd=125;
    }
sprintf(buffer, mensaje);
_moveto(xd,yd);
yd=(short)(yd+15);
_setcolor (30);
_outgtext(buffer);
}

/*************************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
   Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST

void main()
{
/* int i;*/
modo_grafico();
graf_display();
graf_base();
/* for (i=0;i<1000;i++) graf_reloj();*/
graf_nuevo_grafico(5,3);
getch();
graf_nuevo_grafico(1,3);
getch();
graf_nuevo_grafico(2,3);
getch();
graf_nuevo_grafico(3,3);
getch();
graf_nuevo_grafico(4,3);
getch();
graf_nuevo_grafico(5,3);
getch();
    _setvideomode( _DEFAULTMODE );
}
#endif
```

```
/*****************************************************************************

   GUARDIAN.C -        FUNCIONES PARA WATCHDOG

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 6/6/97

*****************************************************************************/

/******************** Librerias  a incluir *************************/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <process.h>
#include <sys\types.h>

#define INTR 0X1C    /* Interrupci¢n del reloj */
/*#define TEST*/

/********************** Subrutinas del fichero **********************/
void (interrupt *oldhandler)();
void reboot(void);
void inicia_guardian(void);
void finaliza_guardian(void);

/******************* Variables comunes del programa *******************/

extern int contador_guardian;
extern int dos;

/*****************************************************************************/

void interrupt guardian()
{
int rebotar=0;
_disable(); /* desactiva interrupciones durante el tratamiento */
if (dos==0) contador_guardian++;
if (contador_guardian>2000) rebotar=1;
_enable();  /* reactiva la interrupci¢n al finalizar el tratamiento */
oldhandler(); /* llama a la rutina inicial */
if (rebotar==1)         reboot();
}

/*****************************************************************************/

#define MAGIC          0              /* for cold restart */
/* #define MAGIC           0x1234          /* for warm restart */

#define BOOT_SEG       0xffffL
#define BOOT_OFF       0x0000L
#define BOOT_ADR       ((BOOT_SEG << 16) | BOOT_OFF)

#define DOS_SEG               0x0040L
#define RESET_FLAG     0x0072L
#define RESET_ADR      ((DOS_SEG << 16) | RESET_FLAG)

void reboot(void)
{
       void ((far *fp)()) = (void (far *)()) BOOT_ADR;

       *(int far *)RESET_ADR = MAGIC;
       (*fp)();
}

/*****************************************************************************/

void inicia_guardian(void)
{
oldhandler = _dos_getvect(INTR);     /* Guarda el vector de interrupciones inicial */
_dos_setvect(INTR, guardian); /* Instala el nuevo vector de interrupciones */
}

/*****************************************************************************/

void finaliza_guardian(void)
{
_dos_setvect(INTR, oldhandler);      /* Recupera el vector de interrupciones inicial */
```



```
}

/*****************************************************************************/

#ifdef TEST

int fi=0;

int main(void)
{

inicia_guardian();

printf("Antes: %i\n ",contador_guardian);
dos=1;
system("command.com");
dos=0;
printf("Despues: %i\n ",contador_guardian);

/*
while (fi==0)
{
printf("%i\n ",contador_guardian);
}
*/

finaliza_guardian();

return 0;
}
#endif
```

```
/*************************************************************************

   PCL726.C -  FUNCIONES PARA UTILIZAR LA PLACA DE SALIDAS 4-20 mA PCL726

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 21/9/98

*************************************************************************/

/********************* Librerias  a incluir ***************************/

#include <stdio.h>
#include <conio.h>
#include <graph.h>
#include <time.h>
#include "nombres.h"

/************************ Definiciones ********************************/

#define BASE 0x2c0                    /*direcci¢ base de la placa pcl726*/

/************************ Subrutinas del fichero **********************/

int pcl726 (int canal, float miliamperios);
void actuacion_externa_pcl726(void);

/************************ Subrutinas externas *************************/

extern void graf_borra_ayuda(void);
extern void graf_ayuda_general(void);
extern void delay( clock_t wait );

/******************* Variables comunes del programa *******************/

extern estado_planta planta;
extern primera medidas[3];

/*************************************************************************/
/* Funci¢n para pedir por pantalla en modo texto los par metros para
   ejecutar la funci¢n pcl726, que var¡a los mA de salida de un canal. */

void actuacion_externa_pcl726(void)
{
int canal;
float miliamperios;

printf ("N£mero de canal: ");
fscanf (stdin, "%d", &canal);
printf ("Miliamperios:  ");
fscanf (stdin, "%f", &miliamperios);

pcl726 (canal, miliamperios);
}

/*************************************************************************/
/* Funci¢n para pedir por pantalla en modo gr fico los par metros para
   ejecutar la funci¢n pcl726, que var¡a los mA de salida de un canal. */

void graf_actuacion_externa_pcl726(void)
{
int canal;
float miliamperios;
char fuente[20]="t'Tms Rmn'h12w50b";

graf_borra_ayuda();
_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(7);
_moveto(5,2);
_outgtext("Variaci¢n de los miliamperios de salida de la placa PCL726");
_moveto(5,19);
_setcolor(29);
_outgtext("N£mero de canal: ");
_settextposition(2,17);
fscanf (stdin, "%d", &canal);
_moveto(250,18);
```



```
_outgtext("Miliamperios:  ");
_settextposition(2,46);
fscanf (stdin, "%f", &miliamperios);

graf_ayuda_general();

pcl726 (canal, miliamperios);
}

/*************************************************************************/
/* Control de la placa pcl726.
   Funci¢n para variar los mA de salida de un canal de la placa pcl726.    */


int pcl726 (int canal, float miliamperios)
{
unsigned char high_byte,low_byte;
int rango=20-4;
int consigna;

    if (miliamperios>20 || miliamperios<4) return 1;
    if (canal>5 || canal<0) return 2;
     /*Si los par metros no son correctos, devuelve un c¢digo de error*/

    if (canal==5 && miliamperios==4.0F &&  planta.mA[5]!=4.0F)
       /* Para abrir un poco la v lvula antes de cerrarla*/
       {
       _outp(BASE+canal*2, (char) 11);
       _outp(BASE+1+canal*2, (char) 0);
       delay(200);
       }

    consigna=(int) (4095*(miliamperios-4.)/rango);

    high_byte=(char) (consigna>>8);
    low_byte=(char) (consigna & 255);

    _outp(BASE+canal*2,high_byte);
    _outp(BASE+1+canal*2,low_byte);

    planta.mA[canal]=miliamperios;  /* Guarda mA */
    if (canal>2 && canal<6) medidas[canal-3].mA=miliamperios;
                  /* Guarda mA actuaci¢n ox¡geno */

return 0;
}

/*************************************************************************/
```

```
/*****************************************************************************

   PLC.C - FUNCIONES PARA LA COMUNICACION CON EL PLC IZUMI/FA2J

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 20/1/99

*****************************************************************************/

/********************* Librerias  a incluir ***************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <graph.h>
#include "nombres.h"

/************************ Definiciones *******************************/

/*#define TEST*/
#ifdef TEST
#define extern
#endif

#define COM1     0
#define SETTINGS1 (_COM_9600 | _COM_CHR8 | _COM_STOP1 | _COM_EVENPARITY)


/************************ Subrutinas del fichero *********************/

void plc_on(void);
void plc_off(void);
void dosifica(int bomba);
void alarma(int motivo);
void estado_salida(unsigned int numero, unsigned int off_on);
int envia_orden(unsigned char orden[8]);
void actuacion_externa_salidas(void);
void purga(int duracion);
void graf_actuacion_externa_salidas(void);

void supervision_plc(void);
int inicializar_plc(void);
int plc_inputs(void);
int plc_outputs(void);
int plc_monitor(int tipo, int numero);
int plc_set(int accion, int numero);
int plc_timer(int valor, int numero);
int plc_counter(int valor, int numero);
int plc_programa(int on_off);
int cambiar_alimentacion(void);

/********************* Subrutinas externas necesarias ******************/

extern void dec_to_bin(int numero_decimal);
extern int pcl726 (unsigned char canal, float miliamperios);
extern void delay( clock_t wait );
extern void graf_led( int led, int on_off );
extern void graf_nivel(int nivel, int on_off);
extern void guarda_mensaje( char *mensaje);
extern void graf_borra_ayuda(void);
extern void graf_ayuda_general(void);
extern void inicializar_tiempos(void);
extern void imprime(char *mensaje);
extern int mensaje_a_SUN(char *mensaje);
extern int mA_caudal( char *fichero, float caudal, float *mA);

/******************* Variables comunes del programa ********************/

extern estado_planta planta;

int monitor[16];

/*****************************************************************************/
```



```
/* Enciende las salidas del plc indicadas en salidas_iniciales.          */

void plc_on(void)
{
unsigned int in;

in=_bios_serialcom(_COM_INIT, COM1, SETTINGS1);    /*Inicializa el port com.*/
/*fputs("STATUS COM1: \t",stdout);
dec_to_bin(in);*/

plc_programa(1);  /* Prepara programa PLC */

return;
}

/*****************************************************************************/
/* Apaga las salidas del plc indicadas en salidas_iniciales.             */

void plc_off(void)
{

plc_programa(0);  /* Apaga programa PLC */

return;
}

/*****************************************************************************/
/* Supervisa el funcionamiento del PLC                                   */

void supervision_plc(void)
{
int i;
int status=0;

if (planta.com_PLC==0) return;

if (inicializar_plc()==0)   /*Si la comunicación es correcta*/
  {
   /* Gráfica del nivel en el tercer tanque */
  plc_monitor(5,0); /*Contador bomba encendida*/
  for (i=0;i<10;i++)  status=status | ((monitor[i] & 1)<<i);
  if (status>0 && status< 31) graf_nivel ( status, 1);
  else {
       plc_monitor(5,1); /*Contador bomba apagada*/
       status=0;
       for (i=0;i<10;i++)   status=status | ((monitor[i] & 1)<<i);
       if (status>0 && status< 21)   graf_nivel ( status, 0);
       }

   /* Detección de alarmas PLC*/
  plc_monitor(2,40);          /*Rels internos M50-M57*/
  if (monitor[0]==1 || monitor[1]==1 || monitor[2]==1 || monitor[3]==1
      || monitor[4]==1)  plc_monitor(2,40);  /* Comprobamos el estado */
  if (monitor[0]==1 || monitor[1]==1 || monitor[2]==1 || monitor[3]==1
      || monitor[4]==1)  plc_monitor(2,40);  /* Comprobamos el estado */

  if (monitor[0]==1) alarma(0);  /*M50 encendido, 30 veces bomba encendida*/
  if (monitor[1]==1) alarma(1);       /*M51 encendido, 20 veces bomba apagada*/
  if (monitor[2]==1) alarma(2);       /*M52 encendido, nivel tanque 1 alto*/
  if (monitor[3]==1) alarma(3);       /*M53 encendido, nivel tanque 2 alto*/
  if (monitor[4]==1) alarma(4);       /*M54 encendido, escape de agua*/

   /* Estado de las salidas del PLC */
  plc_outputs();          /*Mira el estado de la salidas*/
  for (i=0;i<24;i++) graf_led( i, planta.salidas_plc[i]);
  }
else
  {
  guarda_mensaje( "No es posible la comunicación con el PLCt\t\x0");
  imprime("No es posible la comunicación con el PLC\v\v\n\x0");
/*  planta.com_PLC=0;*/
  }

}

/*****************************************************************************/
/* Tratamiento de las alarmas detectadas por control_de_nivel.           */
```

```
void alarma(int motivo)
{
float mA=0.0F;
int i;

switch(motivo)
{
    case 0:             /*Posible escape*/
      /* pcl726(0,4);          /*cierra entrada*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -2);
       guarda_mensaje( "Alarma, de funcionamiento bombas, 30 veces encendida\t\t\x0");
       imprime("Alarma de funcionamiento bombas, 30 veces encendida\v\v\n\x0");
       mensaje_a_SUN("Alarma, de funcionamiento bombas, 30 veces encendida\t\x0");
       planta.alarma=0;
    break;
    case 1:             /*Posible escape*/
      /* pcl726(0,4);          /*cierra entrada*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -2);
       guarda_mensaje( "Alarma, de funcionamiento bombas, 20 veces apagada\t\t\x0");
       imprime("Alarma de funcionamiento bombas, 20 veces apagada\v\v\n\x0");
       mensaje_a_SUN("Alarma, de funcionamiento bombas, 20 veces apagada\t\x0");
       planta.alarma=1;
    break;
    case 2:             /*Nivel alto T-1*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -1);
       guarda_mensaje( "Alarma de nivel alto tanque 1\t\t\x0");
       imprime("Alarma de nivel alto tanque 1\v\v\n\x0");
       mensaje_a_SUN("Alarma de nivel alto tanque 1\t\x0");
       planta.alarma=2;
    break;
    case 3:             /*Nivel alto T-2*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -1);
       guarda_mensaje( "Alarma de nivel alto tanque 2\t\t\x0");
       imprime("Alarma de nivel alto tanque 2\v\v\n\x0");
       mensaje_a_SUN("Alarma de nivel alto tanque 2\t\x0");
       planta.alarma=3;
    break;
    case 4:             /*Alarma escape detectado*/
      /* pcl726(0,4);          /*cierra entrada*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -3);
       guarda_mensaje( "Alarma, escape de agua\t\t\x0");
       imprime("Alarma de funcionamiento, escape de agua\v\v\n\x0");
       mensaje_a_SUN("Alarma, escape de agua\t\x0");
       planta.alarma=4;
    break;
    case 5:             /*Parada manual*/
       pcl726(0,4);          /*cierra entrada*/
       plc_off();            /*apaga bombas*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -4);
       guarda_mensaje( "Parada manual\t\t\x0");
       imprime("Alarma de funcionamiento, parada manual\v\v\n\x0");
       mensaje_a_SUN("Alarma, parada manual\t\x0");
       planta.alarma=5;
    break;
    case 6:             /*Parada SUN*/
       pcl726(0,4);          /*cierra entrada*/
       plc_off();            /*apaga bombas*/
       planta.control_nivel=0;          /*desactiva la planta*/
       graf_nivel ( 0, -4);
       guarda_mensaje( "Parada SUN\t\t\x0");
       imprime("Alarma de funcionamiento, parada SUN\v\v\n\x0");
       planta.alarma=6;
    break;
    case -1:   /* Reinicia la planta */
       plc_on();
       mA_caudal( "c:\\depura\\v_agua.dat", planta.caudal_agua, &mA);
       pcl726(0,mA);  /*Abre la v lvula de entrada de agua*/
       planta.control_nivel=1;          /*Reactiva la planta*/
       planta.sondas=1;
       planta.conexion_SUN=1;
```



```
       planta.com_PLC=1;
       supervision_plc();

       guarda_mensaje( "Reinicializaci¢n\t\t\x0");
       imprime("Reinicializaci¢n\v\v\n\x0");
       mensaje_a_SUN("Reinicializaci¢n\t\x0");
       planta.alarma=-1;
    break;
    case -2:   /* Reinicia la planta desde la SUN*/
       plc_on();
       mA_caudal( "c:\\depura\\v_agua.dat", planta.caudal_agua, &mA);
       pcl726(0,mA);  /*Abre la v lvula de entrada de agua*/
       planta.control_nivel=1;          /*Reactiva la planta*/
       planta.sondas=1;
       planta.conexion_SUN=1;
       planta.com_PLC=1;

       guarda_mensaje( "Reinicializaci¢n SUN\t\t\x0");
       imprime("Reinicializaci¢n SUN\v\v\n\x0");
       planta.alarma=-2;
    break;
}
plc_outputs();     /*Mira el estado de la salidas*/
for (i=0;i<24;i++) graf_led( i, planta.salidas_plc[i]);

return;
}

/*********************************************************************/
/* Funci¢n para encender o apagar una salida del PLC.                */

void estado_salida(unsigned int numero, unsigned int off_on)
{
int j=0;
int data1,data2,data3;
unsigned char estado[5] = {0x4B,0x31,0x38,0x30,0x0D};

if (off_on==1)  estado[1]=0x39;  /*Salida encendida*/

if (numero<0x10)           /*Adapta el n£mero a su nomenclatura*/
  estado[3]=(unsigned char) (0x30+numero);
else
  {
  estado[2]=0x39;
  estado[3]=(unsigned char) (0x20+numero);
  }

 for (j=0;j<5;j++)           /*Escribimos y leemos en el puerto*/
     {
     _bios_serialcom(_COM_SEND, COM1, estado[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
     data1=_bios_serialcom(_COM_RECEIVE, COM1, 0);
     data2=_bios_serialcom(_COM_RECEIVE, COM1, 0);

     if (data1==0x63)
       {
       graf_led(numero, off_on);                    /*Encendido o apagado*/
       planta.salidas_plc[numero] = off_on;             /*Guardamos estado*/
       }
     if (data1==0x64)
       {
       data3=_bios_serialcom(_COM_RECEIVE, COM1, 0);
       graf_led(numero, 2);                         /*Error mensaje*/
       planta.salidas_plc[numero] = 2;                  /*Guardamos estado*/
       }
    if (data1!=0x63 && data1!=0x64)
       {
       graf_led(numero, 2);                         /*Error comunicaci¢n*/
       planta.salidas_plc[numero] = 2;                  /*Guardamos estado*/
       }
return;
}


/*********************************************************************/
/* Funci¢n para pedir por pantalla en modo texto los par metros para
```

```
   ejecutar estado_salida, que enciende o apaga una salida del PLC.   */

void actuacion_externa_salidas(void)
{
int salida,status;

printf("N£mero de salida: ");
fscanf(stdin, "%d", &salida);

printf("Estado (Abierto 1, Cerrado 0): ");
fscanf(stdin, "%d", &status);

estado_salida(salida,status);

return;
}


/************************************************************************/
/* Funci¢n para pedir por pantalla en modo gr fico los par metros para
   ejecutar estado_salida, que enciende o apaga una salida del PLC.   */

void graf_actuacion_externa_salidas(void)
{
int salida,status;
char fuente[20]="t'Tms Rmn'h12w50b";

graf_borra_ayuda();
_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(7);
_moveto(5,2);
_outgtext("Actuaci¢n sobre las salidas del PLC FA2J");
_moveto(5,19);
_setcolor(29);
_outgtext("N£mero de salida: ");
_settextposition(2,17);
fscanf (stdin, "%d", &salida);
_moveto(250,18);
_outgtext("Estado (Abierto 1, cerrado 0):");
_settextposition(2,55);
fscanf (stdin, "%d", &status);

estado_salida(salida,status);

graf_ayuda_general();
}

/************************************************************************/
/* Implementaci¢n purga. Enciende durante el intervalo de tiempo especificado
   la salida correspondiente a la purga.                                */

void purga(int duracion)
    {
       estado_salida(17,1);
       delay(duracion);
       estado_salida(17,0);
return;
    }


/************************************************************************/

int plc_inputs(void)
{
int j=0;
unsigned char orden1[5] = {0x42,0x31,0x30,0x30,0x0D};
unsigned char orden2[5] = {0x42,0x31,0x30,0x38,0x0D};
unsigned char respuesta[4];
int status=0;

/* orden 1
   0x42 monitor de se¤ales
   0x31 I/O, entradas y salidas
   0x30,0x30 con esto, segun la pagina 21, le estoy pidiendo el estado de 8 de
             las entradas, a partir de la 00, segun la pagina 20 debe a¤adirse
```



```
             0x3_ y 0x3_ donde _ son los numeros 00
   0x0D fin de mensaje*/

/* orden 2
   0x42 monitor de se¤ales
   0x31 I/O, entradas y salidas
   0x30,0x38 con esto, segun la pagina 21, le estoy pidiendo el estado de 8 de
             las entradas, a partir de la 08, segun la pagina 20 debe a¤adirse
             0x3_ y 0x3_ donde _ son los numeros 08
   0x0D fin de mensaje*/

for (j=0;j<5;j++)     /*mando y recibo mensaje*/
     {
     _bios_serialcom(_COM_SEND, COM1, orden1[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
for (j=0;j<4;j++)     /*recibo respuesta*/
    respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);

if (respuesta[0]!=0x66) return -1;
  /*si no ha leido bien sale devolviendo error*/

status = respuesta[2]-0x30;
  /*en el byte 2 de la respuesta est  lo que quiero, segun pagina 26
    el estado de las 4 primeras entradas*/
/*el formato de respuesta es el 66, esta en la pagina 16*/
for (j=0;j<8;j++)
     {
     planta.entradas_plc[j] = status & 1;
     status >>= 1;        /*transformo de decimal a binario, bit a bit*/
     if (j==3) status = respuesta[1]-0x30;
  /*en el byte 1 de la respuesta est  lo que quiero, segun pagina 26
    el estado de las 4 siguientes entradas*/
     }


/*aqui repetimos el procedimiento, pero a partir de la entrada numero 8*/
for (j=0;j<5;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden2[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
for (j=0;j<4;j++)
    respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);

status = respuesta[2]-0x30;
for (j=8;j<16;j++)
     {
     planta.entradas_plc[j] = status & 1;
     status >>= 1;
     if (j==11) status = respuesta[1]-0x30;
     }

return 0;
}

/************************************************************************/

int plc_outputs(void)
{
int j=0;
unsigned char orden1[5] = {0x42,0x31,0x38,0x30,0x0D};
unsigned char orden2[5] = {0x42,0x31,0x38,0x38,0x0D};
unsigned char orden3[5] = {0x42,0x31,0x39,0x30,0x0D};
unsigned char respuesta[4];
int status=0;
/* orden 1
   0x42 monitor de se¤ales
   0x31 I/O, entradas y salidas
   0x38,0x30 con esto, segun la pagina 21, le estoy pidiendo el estado de 8 de
             las salidas, a partir de la 80, segun la pagina 20 debe a¤adirse
             0x3_ y 0x3_ donde _ son los numeros 80
   0x0D fin de mensaje*/

/* orden 2
   0x42 monitor de se¤ales
   0x31 I/O, entradas y salidas
```

```
   0x38,0x38 con esto, segun la pagina 21, le estoy pidiendo el estado de 8 de
             las salidas, a partir de la 88, segun la pagina 20 debe a¤adirse
             0x3_ y 0x3_ donde _ son los numeros 88
   0x0D fin de mensaje*/

/* orden 3
   0x42 monitor de se¤ales
   0x31 I/O, entradas y salidas
   0x39,0x30 con esto, segun la pagina 21, le estoy pidiendo el estado de 8 de
             las salidas, a partir de la 08, segun la pagina 20 debe a¤adirse
             0x3_ y 0x3_ donde _ son los numeros 90
   0x0D fin de mensaje*/


for (j=0;j<5;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden1[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
for (j=0;j<4;j++)
    respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);

if (respuesta[0]!=0x66) return -1;

status = respuesta[2]-0x30;
for (j=0;j<8;j++)
     {
     planta.salidas_plc[j] = status & 1;
     status >>= 1;
     if (j==3) status = respuesta[1]-0x30;
     }

for (j=0;j<5;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden2[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
for (j=0;j<4;j++)
    respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);

status = respuesta[2]-0x30;
for (j=8;j<16;j++)
     {
     planta.salidas_plc[j] = status & 1;
     status >>= 1;
     if (j==11) status = respuesta[1]-0x30;
     }

for (j=0;j<5;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden3[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
for (j=0;j<4;j++)
    respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);

status = respuesta[2]-0x30;
for (j=16;j<24;j++)
     {
     planta.salidas_plc[j] = status & 1;
     status >>= 1;
     if (j==19) status = respuesta[1]-0x30;
     }

return 0;
}


/*************************************************************************/

int inicializar_plc(void)
{
unsigned int in;
int j=0;
unsigned char orden[2] = {0x45,0x0D};
unsigned char respuesta[3];
int status=0;
```



```
in=_bios_serialcom(_COM_INIT, COM1, SETTINGS1);    /*Inicializa el port com.*/
for (j=0;j<2;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }
for (j=0;j<3;j++)
     respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
status =  respuesta[1] & 0x0F ;

if (respuesta[0]!=0x61) return -1;

return 0;
}

/*************************************************************************/
/* Funci¢n general para monitorizar el estado de entradas, salidas,
   rel,s internos... del PLC.                                            */

int plc_monitor(int tipo, int numero)
{
/***********************************
   Tipo: 1 ---> Input/output   I/O
   Tipo: 2 ---> Internal relay        IR
   Tipo: 3 ---> Shift register        SFR
   Tipo: 4 ---> Timer          TIM
   Tipo: 5 ---> Counter        CNT
   Tipo: 7 ---> D register     DR
***********************************/

int j=0;
unsigned char orden[5] = {0x42,0x30,0x30,0x30,0x0D};
unsigned char respuesta[6];
int status=0;

if ( tipo<1 || tipo >7 || tipo==6 ) return -2;
orden[1]=(unsigned char) (0x30 + tipo);

if (numero<0 || numero >255) return -2;
orden[2]=(unsigned char) (0x30 | ((numero>>4) & 0x0F));
orden[3]=(unsigned char) (0x30 | (numero & 0x0F));

for (j=0;j<5;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }

switch(tipo)
    {
    case 1:
    case 2:
    case 3:
       for (j=0;j<4;j++)
          respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
       status =  ((respuesta[1] & 0x0F) << 4) + (respuesta[2] & 0x0F) ;
    break;
    case 4:
    case 5:
    case 7:
       for (j=0;j<6;j++)
          respuesta[j]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
       status =  ((respuesta[1] & 0x0F) << 12) + ((respuesta[2] & 0x0F) << 8)
             + ((respuesta[3] & 0x0F) << 4) +  (respuesta[4] & 0x0F) ;
    break;
    }

for (j=0;j<16;j++)
     {
     monitor[j] = status & 1;
     status >>= 1;
     }

return 0;
}
```

```
/*********************************************************************/

int plc_set(int accion, int numero)
{
/*************************************************
   Accion: 1    ---> Input/output      I/O   OFF
   Accion: 2    ---> Internal relay    IR    OFF
   Accion: 3    ---> Shift register    SFR   OFF
   Accion: 9    ---> Input/output      I/O   ON
   Accion: A(10) ---> Internal relay   IR    ON
   Accion: B(11) ---> Shift register   SFR   ON
*************************************************/

int j=0;
unsigned char orden[5] = {0x4b,0x30,0x30,0x30,0x0D};
unsigned char respuesta[3];

if ( accion<1 || accion >11 ) return -2;
orden[1]=(unsigned char) (0x30 + accion);

if (numero<0 || numero >255) return -2;
orden[2]=(unsigned char) (0x30 | ((numero>>4) & 0x0F));
orden[3]=(unsigned char) (0x30 | (numero & 0x0F));

for (j=0;j<5;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }

respuesta[0]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
respuesta[1]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
if (respuesta[0]==0x64)
    {
    respuesta[2]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
    return -1;
    }

return 0;
}
/*********************************************************************/

int plc_timer( int valor, int numero)
{
int j=0;
unsigned char orden[8] = {0x43,0x30,0x30,0x30,0x30,0x30,0x30,0x0D};
unsigned char respuesta[3];

if ( valor<0 || valor > 0xFFFF ) return -2;

if (numero<0 || numero >255) return -2;
orden[1]=(unsigned char) (0x30 | ((numero>>4) & 0x0F));
orden[2]=(unsigned char) (0x30 | (numero & 0x0F));

orden[3]=(unsigned char) (0x30 | ((valor>>12) & 0x0F));
orden[4]=(unsigned char) (0x30 | ((valor>>8) & 0x0F));
orden[5]=(unsigned char) (0x30 | ((valor>>4) & 0x0F));
orden[6]=(unsigned char) (0x30 | (valor & 0x0F));

for (j=0;j<8;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }

respuesta[0]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
respuesta[1]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
if (respuesta[0]==0x64)
    {
    respuesta[2]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
    return -1;
    }

return 0;
}
```



```
/*********************************************************************/

int plc_counter( int valor, int numero)
{
int j=0;
unsigned char orden[8] = {0x44,0x30,0x30,0x30,0x30,0x30,0x30,0x0D};
unsigned char respuesta[3];

if ( valor<0 || valor > 0xFFFF ) return -2;

if (numero<0 || numero >255) return -2;
orden[1]=(unsigned char) (0x30 | ((numero>>4) & 0x0F));
orden[2]=(unsigned char) (0x30 | (numero & 0x0F));

orden[3]=(unsigned char) (0x30 | ((valor>>12) & 0x0F));
orden[4]=(unsigned char) (0x30 | ((valor>>8) & 0x0F));
orden[5]=(unsigned char) (0x30 | ((valor>>4) & 0x0F));
orden[6]=(unsigned char) (0x30 | (valor & 0x0F));

for (j=0;j<8;j++)
     {
     _bios_serialcom(_COM_SEND, COM1, orden[j]);
     _bios_serialcom(_COM_RECEIVE, COM1, 0);
     }

respuesta[0]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
respuesta[1]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
if (respuesta[0]==0x64)
    {
    respuesta[2]=(unsigned char)_bios_serialcom(_COM_RECEIVE, COM1, 0);
    return -1;
    }

return 0;
}

/*********************************************************************/

int plc_programa(int on_off)
{
int intervalo;

switch(on_off)
  {
  case 1:
     intervalo=planta.duracion_purga;
     if (plc_timer( intervalo, 3)!=0) plc_timer( intervalo, 3);
     intervalo=planta.tiempo_purga*10-planta.duracion_purga;
     if (plc_timer( intervalo, 4)!=0) plc_timer( intervalo, 4);

     intervalo=planta.intervalo_dosificacion[0]*10-1;
     if (plc_timer( intervalo, 11)!=0) plc_timer( intervalo, 11);
     intervalo=planta.intervalo_dosificacion[1]*10-1;
     if (plc_timer( intervalo, 13)!=0) plc_timer( intervalo, 13);
     intervalo=planta.intervalo_limp_dosif[0]*10-1;
     if (plc_timer( intervalo, 14)!=0) plc_timer( intervalo, 14);
     plc_counter( planta.pistonadas_limp_dosif[0], 10);

     intervalo=planta.intervalo_flojet_on*10;
     if (plc_timer( intervalo, 30)!=0) plc_timer( intervalo, 30);
     if (plc_timer( intervalo, 32)!=0) plc_timer( intervalo, 32);

     intervalo=planta.intervalo_flojet_off*10;
     if (plc_timer( intervalo, 31)!=0) plc_timer( intervalo, 31);
     if (plc_timer( intervalo, 33)!=0) plc_timer( intervalo, 33);

     if (plc_set(10,48)!=0) plc_set(10,48); /*Enciende programa PLC*/
  break;
  case 0:
     if (plc_set(10,49)!=0) plc_set(10,49); /*Apaga programa PLC*/
  break;
  }
return 0;
}

/*********************************************************************/
```

```
int cambiar_alimentacion(void)
{
char mensaje[100];
int intervalo,hora;
float mA=0.0F;
struct  tm *t;

  planta.t_actual=time(NULL);
  t = localtime( &planta.t_actual );
  hora = t->tm_hour;

  planta.intervalo_dosificacion[0]=planta.alimento[hora].nitrogeno;
  planta.intervalo_dosificacion[1]=planta.alimento[hora].carbono;
  planta.caudal_agua=planta.alimento[hora].agua;

  intervalo=planta.intervalo_dosificacion[0]*10-1;
  if (plc_timer( intervalo, 11)!=0) plc_timer( intervalo, 11);
  intervalo=planta.intervalo_dosificacion[1]*10-1;
  if (plc_timer( intervalo, 13)!=0) plc_timer( intervalo, 13);

  mA_caudal( "c:\\depura\\v_agua.dat", planta.caudal_agua, &mA);
  pcl726(0,mA);       /*Abre la v lvula de entrada de agua*/

sprintf(mensaje,"Cambio alimentaci¢n: N %4i seg, C %4i seg, Q %5.2f ml/min",
   planta.intervalo_dosificacion[0], planta.intervalo_dosificacion[1],
   planta.caudal_agua);
guarda_mensaje( mensaje );

return 0;
}

/*************************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
    Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST
void dec_to_bin(int numero_decimal)
{numero_decimal=0;}
int pcl726 (unsigned char canal, float miliamperios)
{canal=0; miliamperios=0.0F; return 0;}
void delay( clock_t wait )
{wait=0;}
void graf_led( int led, int on_off )
{led=0; on_off=0;}
void graf_nivel(int nivel, int on_off)
{nivel=0; on_off=0;}
void guarda_mensaje( char *mensaje)
{mensaje=0;}
void graf_borra_ayuda(void){}
void graf_ayuda_general(void){}
void inicializar_tiempos(void){}
int mensaje_a_SUN(char *mensaje)
{mensaje=0; return 0;}
void imprime(char *mensaje)
{mensaje=0;}

int main( int argc, char *argv[])
  {
int accion,numero;

if (argc==1)
    {
    printf("\nUtiliza:  plc   valor numero \n");
    return -1;
    }

accion=atoi(argv[1]);
numero=atoi(argv[2]);

if (inicializar_plc()!=0)
    {
    printf("ERROR: PLC no conectado, o comunicaci¢n incorrecta");
    return -2;
    }

if (plc_timer( accion, numero)==0) printf("\nSet OK\n");
else printf("\nSet NO OK\n");
```



```
return 0;
p}

#endif

/*************************************************************************/
```

```
/*****************************************************************************

   PRINTER.C - FUNCIONES PARA MANEJAR LA IMPRESORA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 4/10/95

*****************************************************************************/

#include <dos.h>
#include <bios.h>
#include <conio.h>
#include <stdio.h>
#include <time.h>

#define LPT1 0

void imprime(char *mensaje);

/*****************************************************************************/
/* Mira el estado de la impresora, y si es correcto, imprime el mensaje.   */


void imprime(char *mensaje)
{
unsigned  pstatus;
int i=0,largo;
time_t ltime;
char hora[30];

time (&ltime);
sprintf(hora,"%s", ctime( &ltime ));

while(mensaje[i]!=10 && i<80 )       i++;
largo=i;
if (largo==80) return;

/* Fail if any error bit is on, or if either operation bit is off. */
pstatus = _bios_printer( _PRINTER_STATUS, LPT1, 0 );
if( (pstatus & 0x29) || !(pstatus & 0x80) || !(pstatus & 0x10) )
        pstatus = 0;
    else
        pstatus = 1;
if (pstatus==1)
    {
    for (i=0;i<26;i++) _bios_printer( _PRINTER_WRITE, LPT1, hora[i] );
    for (i=0;i<largo;i++) _bios_printer( _PRINTER_WRITE, LPT1, mensaje[i] );
    }
}

/*****************************************************************************/

#ifdef TEST
void main(void)
{
char aviso[]="Hola hola pepsicola\v\v\n\x0";
imprime(aviso);
}
#endif

/*****************************************************************************/
```



```
/*****************************************************************************

   TROVIS.C - FUNCIONES PARA LA COMUNICACION CON EL PID TROVIS 6497

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 9/4/97

*****************************************************************************/

/******************** Librerias  a incluir *************************/

#include <bios.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <time.h>

/********************** Definiciones ********************************/

#define TEST

#define COM3     2
#define SETTINGS3 (_COM_9600 | _COM_CHR8 | _COM_STOP1 | _COM_NOPARITY)

/********************** Subrutinas del fichero **********************/

int inicio_comunica(void);
int comunica(unsigned char *orden,unsigned char *respuesta);
unsigned short CRC16(unsigned char *puchMsg,unsigned short usDataLen);
int orden_MODBUS(int direccion, int funcion, int param_a, int param_b);

/******************** Subrutinas externas necesarias ****************/

/****************** Variables comunes del programa ******************/

int registros[60];
int coils[16];

/*****************************************************************/
/* Manda una orden de comprobacion de la comunicacion entre el PC */
/* y el equipo a controlar.                                      */
/*                                                               */
/* Como la lectura del primer registro siempre da lo mismo       */
/* se aprovecha esta lectura para comprobar el estado de la      */
/* comunicacion                                                  */
/*                                                               */
/* Si no la lectura no es la prevista se devuelven los siguientes */
/* valores:                                                      */
/*                                                               */
/*      0 Respuesta correcta.                                    */
/*      1 Comunicacion no establecida correctamente.             */
/*                                                               */
/*****************************************************************/


int inicio_comunica(void)
{
  unsigned char mensaje_comprobacion[9]={0x01,0x03,0x00,0x00,0x00,0x01,0x00,0x00,0x00};
  unsigned char respuesta_comprobacion[8];

  memset(respuesta_comprobacion,0,sizeof(respuesta_comprobacion));

  _bios_serialcom(_COM_INIT, COM3, SETTINGS3); /*Inicializacion del puerto serie*/

  if (comunica(mensaje_comprobacion,respuesta_comprobacion))
        return 1;

  return 0;
}

/*****************************************************************/
/* Manda una orden al equipo a controlar y recibe la respuesta.  */
/* Como hay que enviar un checksum (CRC) high y low se llama a la */
/* subrutina CRC16 para calcularlo.                              */
/*                                                               */
/* Para evitar problemas en el programa si no existe comunicacion */
```

```
/* entre el PC y el equipo se devuelven los siguientes valores:   */
/*                                                                */
/*      0 Respuesta correcta.                                     */
/*      1 Orden incorrecta.                                       */
/*      2 Respuesta nula.                                         */
/*      3 Checksum de envio incorrecto.                           */
/*      4 Checksum de recepcion incorrecto.                       */
/*                                                                */
/******************************************************************/


int comunica(unsigned char *orden,unsigned char *respuesta)
{
  int i=0, largo=0;
  unsigned short out;

  out=CRC16(orden,6);
  orden[7]=(unsigned char) (0xFF & out);
  orden[6]=(unsigned char) (0xFF & (out >> 8));

  for(i=0;i<8;i++)  _bios_serialcom(_COM_SEND,COM3,orden[i]);

for (i=0;i<7;i++)
    {
    respuesta[i] = (unsigned char) _bios_serialcom(_COM_RECEIVE, COM3, 0);
    }
/*
  for (i=0;i<8;i++) printf(" %02X",orden[i]);
  printf("\n");
  for (i=0;i<7;i++) printf(" %02X",respuesta[i]);
  printf("\n");
*/
return 0;
}


/************************************************************************/
/*CRC Generation Function*/

/*The function takes two arguments:

*puchMsg ;     Puntero a la cadena de caracteres
               del mensaje que se quiere calcular CRC
usDataLen ;    Longitud del mensaje (n§ de bytes)

The function returns the CRC as a type unsigned short*/

unsigned short CRC16(unsigned char *puchMsg,unsigned short usDataLen)
{
       unsigned char uchCRCHi = 0xFF ;      /* Inicializacion high CRC byte. Todos los bits a
1*/
       unsigned char uchCRCLo = 0xFF ; /* Inicializacion low CRC byte. Todos los bits a 1*/
       unsigned uIndex ;               /* will index into CRC lookup*/

/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```



```
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;

/* Table of CRC values for low-order byte */
static unsigned/**/ char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;

       while (usDataLen--)             /* pass through message buffer */
              {
              uIndex = uchCRCHi ^ *puchMsg++ ;    /* calculate the CRC */
              uchCRCHi = (unsigned char) (uchCRCLo ^ auchCRCHi[uIndex]) ;
              uchCRCLo = auchCRCLo[uIndex] ;
              }
       return (unsigned short)((uchCRCHi << 8) | uchCRCLo) ;
}

/************************************************************************/
/* Envía una orden a TROVIS mediante protocolo MODBUS y recibe la respuesta.
   Par metros a enviar:

   direccion  ->  Identificativo del esclavo
   funcion    ->  Función Modbus a ejecutar
   param_a    ->  Primer par metro de la función
   param_b    ->  Segundo par metro de la función

   Códigos de retorno:

    0 -> Orden enviada y respuesta correcta
   -1 -> Orden enviada y respuesta incorrecta, cheksum no coincide
   -2 -> Orden enviada y recibido mensaje de error
   -3 -> Orden enviada, respuesta recibida con cheksum correcto, pero
         diferente a la orden enviada

************************************************************************/

int orden_MODBUS(int direccion, int funcion, int param_a, int param_b)
{
#define   RESPUESTAS(A,B)      for   (i=A;i<B;i++)      respuesta[i]   =   (unsigned   char)
_bios_serialcom(_COM_RECEIVE, COM3, 0)

int i=0;
unsigned short out;
unsigned char orden[8];
unsigned char respuesta[140];
memset(respuesta,0,sizeof(respuesta));

orden[0]= (unsigned char) (0xFF & direccion);          /* Dirección del esclavo */
```

```
orden[1]= (unsigned char) (0xFF & funcion);     /* Funci¢n a realizar */
orden[2]= (unsigned char) (0xFF & (param_a >> 8)); /* Par metro a High */
orden[3]= (unsigned char) (0xFF & param_a);     /* Par metro a Low */
orden[4]= (unsigned char) (0xFF & (param_b >> 8)); /* Par metro b High */
orden[5]= (unsigned char) (0xFF & param_b);     /* Par metro b Low */

out=CRC16(orden,6);
orden[6]= (unsigned char) (0xFF & (out >> 8));          /* Cheksum High */
orden[7]= (unsigned char) (0xFF & out);                 /* Cheksum Low */

for(i=0;i<8;i++)  _bios_serialcom(_COM_SEND,COM3,orden[i]); /* Env¡a orden */

switch(funcion){
  case 1:         /*** Read Coil Status ***/
    RESPUESTAS(0,3);                    /* Lee 3 primeros bytes */
    if (respuesta[1]!=orden[1])     /* Mensaje de error */
       {
       RESPUESTAS(3,5);                 /* Lee 2 bytes finales*/
       return -2;
       }
    RESPUESTAS(3,(3+respuesta[2]+2));          /* Lee resto de la respuesta */

    out=CRC16(respuesta,(unsigned short)(i-1)); /*Comprueba cheksum */
    if (  ((unsigned char) (0xFF & (out >> 8))) != respuesta[i-1] ||
          ((unsigned char) (0xFF & out)) != respuesta[i]) return -1;

    for (i=param_a;i<param_a+param_b;i++)    /* Asigna valores */
       coils[i]= (respuesta[3+(i/8)] >> (i-param_a-(i/8*8)) ) & 1;
  break;
  case 3:         /*** Read Holding Register ***/
    RESPUESTAS(0,3);                    /* Lee 3 primeros bytes */
    if (respuesta[1]!=orden[1])     /* Mensaje de error */
       {
       RESPUESTAS(3,5);                 /* Lee 2 bytes finales*/
       return -2;
       }

    RESPUESTAS(3,(3+respuesta[2]+2));          /* Lee resto de la respuesta */

    out=CRC16(respuesta,(unsigned short)(i-1)); /*Comprueba cheksum */
    if (  ((unsigned char) (0xFF & (out >> 8))) != respuesta[i-1] ||
          ((unsigned char) (0xFF & out)) != respuesta[i]) return -1;

    for (i=param_a;i<param_a+param_b;i++)    /* Asigna valores */
       registros[i]= (respuesta[3+2*(i-param_a)] << 8) |
                     respuesta[4+2*(i-param_a)];
  break;
  case 5:         /*** Force Single Coil ***/
    RESPUESTAS(0,3);                    /* Lee 3 primeros bytes */
    if (respuesta[1]!=orden[1])     /* Mensaje de error */
       {
       RESPUESTAS(3,5);                 /* Lee 2 bytes finales*/
       return -2;
       }
    RESPUESTAS(3,8);                    /* Lee resto de bytes */

    out=CRC16(respuesta,(unsigned short)(i-1)); /*Comprueba cheksum */
    if (  ((unsigned char) (0xFF & (out >> 8))) != respuesta[i-1] ||
          ((unsigned char) (0xFF & out)) != respuesta[i]) return -1;
    if ( respuesta[6]!=orden[6] || respuesta[7]!=orden[7] ) return -3;
  break;
  case 6:         /*** Preset Single Register ***/
    RESPUESTAS(0,8);                    /* Lee respuesta */

    out=CRC16(respuesta,(unsigned short)(i-1)); /*Comprueba cheksum */
    if (  ((unsigned char) (0xFF & (out >> 8))) != respuesta[i-1] ||
          ((unsigned char) (0xFF & out)) != respuesta[i]) return -1;
    if ( respuesta[6]!=orden[6] || respuesta[7]!=orden[7] ) return -3;
  break;
    }

printf("\nOrden: \n");
for (i=0;i<8;i++) printf(" %02X",orden[i]);
printf("\nRespuesta: \n");
for (i=0;i<20;i++) printf(" %02X",respuesta[i]);
printf(" ...\n");
```



```
return 0;
}


/*********************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
   Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST

void delay( clock_t wait )
{
    clock_t goal;

    goal = wait + clock();
    while( goal > clock() )
        ;
}

int main( int argc, char *argv[])
 {
int retorno;
int i;

if (argc==1)
    {
    printf("\nUtiliza: trovis funcion parametro_a parametro_b \n");
    return -1;
    }

if (inicio_comunica()){
       printf("\nComunicacion no establecida. Comprobar conexiones");
       return -1;
       }
else printf("\nComunicacion OK !!!\n");

delay(500);

retorno = orden_MODBUS(1, atoi(argv[1]), atoi(argv[2]), atoi(argv[3]) );

printf("\nValor de retorno de la funci¢n:  %i\n", retorno);
if (retorno!=0) return -2;
if (retorno==0) printf("\nOrden OK!!!\n");

if (atoi(argv[1])==1)
    {
     printf("Coils: \n");
     for (i=0;i<16;i++) printf (" %i", coils[i]);
    }
if (atoi(argv[1])==3)
    {
     printf("Registros: \n");
     for (i=0;i<58;i++) printf ("#%i %i  ",i, registros[i]);
    }

/*
orden_MODBUS(1,3,1,1);
orden_MODBUS(0xb,1,4,4);
orden_MODBUS(0xf,5,0xc,0xff00);
orden_MODBUS(1,3,0,1);
orden_MODBUS(0x12,6,9,0x64);
*/

return 0;
}


#endif
```

```
/*************************************************************************

   DEPURA.DAT - FICHERO DE CONFIGURACIÓN

*************************************************************************/

10                  INTERVALO DE CONTROL DE OXIGENO
600                 INTERVALO DE ESTIMACION DEL CONSUMO DE OXIGENO
14 16               INTERVALO DE DOSIFICACION  NITROGENO / CARBONO
300 300             INTERVALO LIMPIEZA DOSIFICADORA NITROGENO / CARBONO
5 5                 NUMERO DE PISTONADAS LIMPIEZA DOSIFICADORA NITROGENO / CARBONO
15                  INTERVALO ACTUACION SUPERVISION PLC
300                 INTERVALO TOMA DE DATOS EN FICHERO
500                 TIEMPO INTERVALO ENTRE PURGAS (M ximo 1000) + 4*1000
519                  / 10 DURACION DE LA PURGA     (SEGUNDOS)
60                  TIEMPO ACTUALIZACION DE LA GRAFICA EN PANTALLA (SEG)
300                 CAUDAL ENTRADA AGUA
600                 CAUDAL RECIRCULACION INTERNA (REATOR 3 -> REACTOR 1)
200                 CAUDAL RECIRCULACION EXTERNA (SEDIMENT->ANAEROBIO)
7                  TIEMPO FLOJETS ENCENDIDAS
23                 TIEMPO FLOJETS APAGADAS
30 50 50            VELOCIDAD AGITADORES (%)
0.0 3.0 3.0         SET POINT CONTROLADOR (PPM)
2.5 2.5 2.5         GANACIA CONTROLADOR
100 100 100         CONSTANTE DE TIEMPO INTEGRAL
0.002 0.002 0.002      CONSTANTE DE TIEMPO DERIVADO
1 1 1               PID ACTIVO (UNO) O INACTIVO (DOS)
4.0 12.0 12.0       SI EL PID ESTA INACTIVO, SALIDA mA VALVULA
-1.3443 0.9964 -1.7844 ORDENADA ORIGEN CORRELACION OXIGENO_MV
0.05288 0.05818 0.08119 PENDIENTE CORRELACION OXIGENO_MV


/*************************************************************************

   ALIMENTO.DAT - FICHERO DE CONFIGURACIÓN

*************************************************************************/

1           TIPO DE ALIMENTACION (0 CONSTANTE, 1 PERFIL)
HORAS
00 01 02 03 04 05 06 07 08 09 10 11  12 13 14 15 16 17 18 19 20 21 22 23
DOSIFICACIàN NITROGENO (CADENCIA EN SEGUNDOS)
28 35 65 100 200 200 160 100 45 25 18 13 10 8 7 6 6 7 7 8 10 12 15 23
DOSIFICACIàN CARBONO (CADENCIA EN SEGUNDOS)
30 37 67 102 202 202 162 102 47 27 20 15 11 10 9 7 7 8 8 9 11 14 16 24
CAUDAL AGUA (ML/MIN)
108 100 95  90  90  93  100 113 128 143 115 184
202 210 215 220 220 207 195 180 165 145 125 110
```



```
/*************************************************************************

   V_AGUA.DAT - FICHERO DE CONFIGURACIÓN

*************************************************************************/

9
4.000  0.000
6.000  13.333
8.000  36.430
10.000 92.166
12.000 156.658
14.000 223.048
16.000 295.567
18.000 370.370
20.000 441.176


/*************************************************************************

   BOMBA1.DAT - FICHERO DE CONFIGURACIÓN

*************************************************************************/

15
4.000    0.000
5.000    53.0
5.500    80.0
6.000    105.6
6.500    133.0
7.000    153.0
7.500    185.0
8.000    219.0
10.000   260.0
12.000   275.0
14.000   347.0
16.000   414.0
18.000   488.0
20.000   550.0


/*************************************************************************

   BOMBA2.DAT - FICHERO DE CONFIGURACIÓN

*************************************************************************/

11
4.000  0.000
5.000  170.0
6.000  340.0
8.000  690.0
10.000 925.0
12.000 1000.0
14.000 1180.0
16.000 1365.0
18.000 1540.0
19.900 1560.0
20.000 2000.0
```

2 PROGRAMA MONITORIZACIÓN Y CONTROL DEL SISTEMA ANALÍTICO

```
/*****************************************************************
   PROGRAMA PRINCIPAL DE GESTIO DELS ANALITZADORS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISIO: 4/1/99
*****************************************************************/

#include "fiacfa.h"

int main( int argc, char *argv[])
{
if (argc==2)   /* Lo hace si entramos 1 par metro */
    {
    if (!_stricmp(argv[1],"/offline"))      offline=1;
       /* Con este par metro funcionaremos offline, sin lecturas */
    if (!_stricmp(argv[1],"/solono3"))      solono3=1;
    if (!_stricmp(argv[1],"/sincal"))       sincal=1;
    }
if (access("c:\\fiacfa\\tmp\\secur.tmp",0) == 0) sincal=1;

inicia_sistema();

do
   {
   dat.t_actual=time(NULL);              /*Actualitzaci¢ temps FIA*/
   ftime(&t);
   dat.t_milli=t.time*1000+t.millitm;

 acciones_comunes();

 switch (estado)
 {
   case 0: /***** C…lcul l¡nia base FIA *****/

   if ((difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_base)
       && (difftime(dat.t_actual,dat.t_inici)<=dat.fi_lectura_base))
      {
        if (difftime(dat.t_milli,dat.t_milli_old)>=200)
        {
        dat.t_milli_old=t.time*1000+t.millitm;
        if (lectura_absor()==0)
           {
           for (i=0;i<=18; i++) mesur_abs[i]=mesur_abs[i+1];
           mesur_abs[19]=valor.absor;
           suma=0.0F;
           for (i=0; i<=19; i++) suma= suma + mesur_abs[i];
           dat.linea_base=suma/20.0F;
           }
        }
      }
    if (difftime(dat.t_actual,dat.t_inici)>dat.fi_lectura_base)
       {
       sprintf(mensaje,"Linia base calculada: %5.3f ",dat.linea_base);
       graf_texto(mensaje,1,1,594,37,1,1);
       estado++;
       }
    break;
    case 1:   /***** C…rrega del loop ******/
      reinicializa_puerto();
      comunica(load,retorno);
      reinicializa_puertob();
      valor.punt_nitrits=dat.punt_mostreig; /* Guardem punt de mostreig */
      valor.punt_nitrats=dat.punt_mostreig;

      if (dat.numero_injeccio>=dat.inj_entre_calibrats)
        {
        comunica(valv_sup_off,retorno);
        comunica(valv_inf_off,retorno);
        }
      estado++;
    break;
    case 2:
    if (difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_max1)
       {
       graf_texto("Iniciem detecci¢ Pic de nitrits",1,1,594,37,1,1);
       dat.t_milli_old=dat.t_milli;
       contador=0;
```



```
       estado++;
       }
    break;
    case 3:    /* Inici de la lectura del primer m…xim */
     if (difftime(dat.t_milli,dat.t_milli_old)>=200)
          {
          dat.t_milli_old=t.time*1000+t.millitm;
          if(lectura_absor()!=0) break;
                 /* Si no ha llegit b,, surt del switch*/
          if (valor.absor>max)
              {
              max=valor.absor;
              contador=0;
              }
          else (contador++);
          if(contador>50)
              {
              hpic1=max-dat.linea_base;
              concentracio_nitrits();
              max=0.0F;
              estado++;
              contador=0;
              sprintf(mensaje,"Pic de nitrits detectat, hpic1 = %6.4f ",hpic1);
              graf_texto(mensaje,1,1,594,37,1,1);
              }
          }
    break;
    case 4:
    if (difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_max2)
       {
       dat.t_milli_old=dat.t_milli;
       contador=0;
       graf_texto("Iniciem detecci¢ Pic de nitrats",1,1,594,37,1,1);
       estado++;
       }
    break;
    case 5:    /***** Detecci¢ segon m…xim *****/
     if (difftime(dat.t_milli,dat.t_milli_old)>=200)
          {
          dat.t_milli_old=t.time*1000+t.millitm;
          if(lectura_absor()!=0) break; /* Si no ha llegit b,, surt del switch*/
          if (valor.absor>max)
              {
              max=valor.absor;
              contador=0;
              }
          else (contador++);
          if(contador>50)
              {
              hpic2=max-dat.linea_base;
              concentracio_nitrats();
              max=0.0F;
              estado++;
              contador=0;
              sprintf(mensaje,"Pic de nitrats detectat, hpic2 = %6.4f ",hpic2);
              graf_texto(mensaje,1,1,594,37,1,1);
              }
          }
    break;
    case 6: /***** Injecci¢ de nova mostra *****/
      if (difftime(dat.t_actual,dat.t_inici)>=dat.t_injeccio)
      {
       dat.t_inici=time(NULL);
       /**** Desactivaci¢n del env¡o s¢lo del segundo valor
       if (dat.repeticion==0) dat.repeticion=1;
           else dat.repeticion=0; /*Para mandar un solo dato a la SUN*/

       reinicializa_puerto();
       comunica(inject,retorno);
       reinicializa_puertob();
       graf_texto("Nova injecci¢ de mostra",1,1,594,37,1,1);
       printf("\a\a\a");

       if(dat.numero_injeccio>=dat.inj_entre_calibrats)  /*  Posat  aqui  per  donar  temps  a
carregar */
       {
         dat.numero_injeccio=0;
```

```
        calibra_no3();
       }
       dat.numero_injeccio++;/* contador per calibrar nitrats */
       for (i=0;i<20;i++) mesur_abs[i]=0.0F;
       estado=0;
      }
   break;
 } /**Fin switch **/


 switch(estado2){
   case 0:
     if(difftime(dat.t_actual,dat.t_cicle)>=(dat.t_canvi_punt_mostreig-60))
       {
       valor.punt_amoni=dat.punt_mostreig;
       if (dat.mostra_manual==0)
            graf_texto("Perm¡s per mostreig manual (F8)",1,1,594,37,1,1);
       if (dat.mostra_manual==1)
            graf_texto("Perm¡s per mostreig autom…tic (F9)",1,1,594,37,1,1);
       estado2++;
       }
   break;
   case 1: /** Canvi del punt de presa de mostra (-17 min. desde t=0) **/
     if(difftime(dat.t_actual,dat.t_cicle)>=dat.t_canvi_punt_mostreig)
       {
       if (dat.mostra_manual==0) /* Mostra normal */
           {
/*         dat.punt_mostreig++;*/
/*         if(dat.punt_mostreig>=5) dat.punt_mostreig=0;*/

/*Cambio para tomar muestra s¢lo de los reactores (puntos 1-2-3)*/
/*         if(dat.punt_mostreig>=4) dat.punt_mostreig=1; */

/*Cambio para tomar muestra s¢lo de dos reactores (puntos 1-3)*/
/*         if(dat.punt_mostreig==1) dat.punt_mostreig=3;
           else dat.punt_mostreig=1;                    */

/*         presa_mostra(dat.punt_mostreig);
           sprintf(mensaje,"Canvi de presa de mostres al punt %i ",dat.punt_mostreig);
           graf_texto(mensaje,1,1,594,37,1,1); */
           /* Cambio para tomar solo salida */

           }
       estado2++;
       }
   break;
   case 2:/** Lectura CFA per obtenir concentraci¢ als 18 minuts desde t=0 **/
    if(difftime(dat.t_actual,dat.t_cicle)>=dat.t_lectura_amoni)
       {
       if(lectura_ammv()==0)
          {
          dat.numero_mostra_CFA++;
          concentracio_amoni();
          graf_texto("Nova concentraci¢ d'amoni calculada",1,1,594,37,1,1);
          estado2++;
          }
       }
   break;
   case 3: /** Finalitzaci¢ del cicle a un reactor **/
      if( difftime(dat.t_actual,dat.t_cicle)>= /*2**/ dat.t_injeccio
        && estado==0 )   /* Para sincronizar los dos tiempos */
      {
       dat.t_cicle=time(NULL);
       dat.t_inici=time(NULL);
       graf_texto("Finalizaci¢ d'un cicle dels analitzadors",1,1,594,37,1,1);

       /* Entrem a calibrat cada 24 hores */

/****   if(dat.numero_mostra_CFA>=72)
       {
         comunica2(amvalv_sup_on,retorno);
         comunica2(amvalv_inf_off,retorno);
         calibra_amoni();
         dat.numero_mostra_CFA=0;
       }   ************************* calibrado de amonio desactivado ***/

       estado2=0;
```



```
      }
    break;
  } /**Fin switch **/

}
 while(dat.fi_programa==0);

/************************ Sortida del programa */

comunica(pump_off, retorno);           /*Para la bomba*/

_setvideomode( _DEFAULTMODE );         /*Acaba con el modo gr fico, pasa a texto*/
remove("c:\\fiacfa\\tmp\\secur.tmp");

return 0;
}


/*********************************************************************
            Subrutina actuacions desde teclat                */

void selectmenu(void)
{
  int tecla;
  int retorno[40];

tecla=_bios_keybrd(_NKEYBRD_READY);  /*Comprueba si se ha presionado una tecla*/
if(tecla==0) return;
_bios_keybrd(_NKEYBRD_READ); /*Vac¡a el buffer, lee el £ltimo caracter*/

  switch(tecla){
       case F1:
           reinicializa_puerto();
           comunica(load,retorno);
           reinicializa_puertob();
       break;
       case F2:
           comunica(pump_on,retorno);
       break;
       case F3:
           comunica(pump_off,retorno);
       break;
       case F4:
           reinicializa_puerto();
           comunica(inject,retorno);
           reinicializa_puertob();
       break;
       case F5:
           reinicializa_puerto();
           comunica(valv_inj,retorno);
           reinicializa_puertob();
       break;
       case F6:
           comunica(valv_sup_off,retorno);
       break;
       case F7:
           comunica(valv_sup_on,retorno);
       break;
       case F8:
           comunica(valv_inf_on,retorno);  /* actuacio manual */
           dat.mostra_manual=1;
           graf_texto("Mostreig manual activat",1,1,594,37,1,1);
       break;
       case F9:
           comunica(valv_inf_off,retorno); /* Fi actuaci¢ manual */
           dat.mostra_manual=0;
           graf_texto("Mostreig autom…tica activat",1,1,594,37,1,1);
       break;
       case F10:
            dat.fi_programa=1;       /* parar programa */
       break;
       case F11:
            dat.fi=1;                        /* parar calibrat */
       break;
       case F12:
            lectura_configuracio(); /* lectura fitxer dades */
```

```
        break;
        case ALTF2:
            comunica2(ampump_on,retorno);
        break;
        case ALTF3:
            comunica2(ampump_off,retorno);
        break;
        case ALTF6:
            comunica2(amvalv_sup_off,retorno);
        break;
        case ALTF7:
            comunica2(amvalv_sup_on,retorno);
        break;
        case ALTF8:
            comunica2(amvalv_inf_on,retorno);  /*actuacio manual*/
        break;
        case ALTF9:
            comunica2(amvalv_inf_off,retorno);
        break;
        case ALTQ:
          /*graf_texto("Inventando valores de la gr fica",1,1,594,37,1,1);*/
            /*inventa_valores();*/
        break;
        case CTRLC:
            /* calibracio_manual=1;*/
        break;
        case CTRLQ:
           _setvideomode( _DEFAULTMODE );
           exit(0);
        break;
        case ALTD:
             salir_al_dos();
        break;
        case ALTH:
            graf_ayuda_general();
        break;
        case ESC:
            graf_presentacion();
        break;
        case RETURN:
        break;
        case BARRA:
            dat.t_grafica=time(NULL);
            boton( 1, "Barra", 603, 3, 636, 20, 1);
            grafica_variable(0);
            grafica_variable(1);
/*          graf_valores_display();*/
            graf_valor_absor();
            graf_valor_mv();
            graf_hora();
            boton( 0, "Barra", 603, 3, 636, 20, 1);
        break;
        case AMUNT:
            boton( 1, "Ar", 610, 30, 629, 50, 2);
            seleccionar_grafico(1);
            boton( 0, "Ar", 610, 30, 629, 50, 2);
        break;
        case ABAIX:
            boton( 1, "Ab", 610, 60, 629, 80, 2);
            seleccionar_grafico(2);
            boton( 0, "Ab", 610, 60, 629, 80, 2);
        break;
        case DRETA:
            boton( 1, "De", 610, 90, 629, 110, 2);
            seleccionar_grafico(3);
            boton( 0, "De", 610, 90, 629, 110, 2);
        break;
        case ESQUERRA:
            boton( 1, "Iz", 610, 120, 629, 140, 2);
            seleccionar_grafico(4);
            boton( 0, "Iz", 610, 120, 629, 140, 2);
        break;
        case CTRLBS:
        break;
        default:
        break;
      }
```



```
return;
}

/*************************************************************************
                    Inicialitzaci¢ de temps                ***************/

void inicializar_tiempos(void)
{
dat.t_actual=time(NULL);
dat.t_cicle=dat.t_actual;
dat.t_inici=dat.t_actual;
dat.t_grafica=dat.t_actual;
dat.t_lectura=dat.t_actual;
dat.t_milli=dat.t_actual;
dat.t_reloj=dat.t_actual;
dat.t_estab=dat.t_actual;
valor.t_amoni=dat.t_actual;
valor.t_nitrats=dat.t_actual;
valor.t_nitrits=dat.t_actual;

}

/*************************************************************************
                Inicialitzaci¢ de les variables          *********/

void inicializar_variables(void)
{
dat.control=1;
/* Cambio reactores   dat.punt_mostreig=0;*/
 /*dat.punt_mostreig=1;*/

dat.punt_mostreig=4;  /*Solo salida*/
dat.numero_injeccio=0;
dat.numero_mostra_CFA=0;
dat.fi=0;
dat.fi_programa=0;
dat.inicio_graf=3;
dat.fin_graf=4;
dat.mostra_manual=0;
dat.posada_en_marxa=1;
dat.conexion_SUN=1;
/*dat.repeticion=0;*/
dat.repeticion=1; /* Para enviar todas las medidas */
}

/*************************************************************************
          Calcul i arxiu de la concentracio d'amoni           **********/

void concentracio_amoni(void)
{
char medida[25];

  valor.amoni= (14.0F/18.0F)*(float)pow(10.,(valor.mv-dat.ordenada)/dat.pendent);
  valor.t_amoni=time(NULL);
  if (dat.mostra_manual==1) valor.punt_amoni=-1; /* La mostra ,s off-line */
  if (dat.mostra_manual==0)
     medida_a_SUN(3, (char)valor.punt_amoni, valor.amoni);
  sprintf(medida,"%6.4f \t%2i  \t",valor.amoni,valor.punt_amoni);
  guarda_medida( medida, "c:\\fiacfa\\datos\\amoni.his");
  sprintf(medida,"%8.4f \t%2i  \t",valor.mv,valor.punt_amoni);
  guarda_medida( medida, "c:\\fiacfa\\datos\\mv.his");
  graf_valores_display();

return;
}

/*************************************************************************
          Calcul de la concentracio de nitrits           **********/

void concentracio_nitrits(void)
{

  valor.nitrits= (dat.a+dat.b*hpic1)*14.0F/46.0F;
  if ( valor.nitrits<0.0F ) valor.nitrits=0.0F;
  valor.t_nitrits=time(NULL);
  if (dat.mostra_manual==1) valor.punt_nitrits=-1; /* La mostra ,s off-line */
  if (dat.mostra_manual==0 && dat.repeticion==1)
```

```
     medida_a_SUN(1, (char)valor.punt_nitrits, valor.nitrits);
  graf_valores_display();

return;
}
/*************************************************************************
          Calcul i arxiu de la concentracio de nitrats            **********/

void concentracio_nitrats(void)
{
char medida[25];

  valor.nitrats= (dat.e+dat.f*((dat.d*hpic2-(dat.a+dat.b*hpic1)+dat.c)/dat.d))*14.0F/62.0F;
  if ( valor.nitrats<0.0F ) valor.nitrats=0.0F;
  valor.t_nitrats=time(NULL);
  if (dat.mostra_manual==1) valor.punt_nitrats=-1; /* La mostra ,s off-line */
  if (dat.mostra_manual==0 && dat.repeticion==1)
     medida_a_SUN(2, (char)valor.punt_nitrats, valor.nitrats);

  sprintf(medida,"%6.4f \t%6.4f \t%2i   \t",
          valor.nitrits,valor.nitrats,valor.punt_nitrats);
  guarda_medida( medida, "c:\\fiacfa\\datos\\nitr.his");
  sprintf(medida,"%6.4f \t%6.4f \t%2i   \t",
          hpic1, hpic2, valor.punt_nitrats);
  guarda_medida( medida, "c:\\fiacfa\\datos\\hpic.his");
  graf_valores_display();

return;
}

/*************************************************************************
     Acciones comunes del programa utilizadas en varias funciones   *****/

void acciones_comunes(void)
{

   selectmenu();
   lectura_ammv();

/***** Actualiza reloj gr fico *****/

   if((int)difftime(dat.t_actual,dat.t_reloj)>=10)
      {
      dat.t_reloj=time(NULL);
      graf_hora();
      }

/***** Lectures per fer gr…fiques *****/

   if ((int)difftime(dat.t_actual,dat.t_lectura)>=4)
      {
      if ( lectura_absor()==0/* && lectura_ammv()==0*/ )   /* Si lee bien */
        {
        dat.t_lectura=time(NULL);
        lectura_ammv();
        lectura_ammv();
        actualizar_matrices_graficas();
        graf_valor_absor();
        graf_valor_mv();
        delay(100);
        }
      }
/***** Gr…fiques per pantalla *****/

   if ((int)difftime(dat.t_actual,dat.t_grafica)>=20)
      {
      dat.t_grafica=time(NULL);
      grafica_variable(0);
      grafica_variable(1);
      }
}

/*************************************************************************
              Inicia tot el sistema FIA CFA                  *****/


void inicia_sistema()
```



```
{
FILE *obert;

inicializar_variables();
inicializar_tiempos();
lectura_configuracio();

      /* Lectura de los £ltimos calibrados */
if (lee_calibrado( "c:\\fiacfa\\datos\\calilno2.his", &dat.a, &dat.b)!=0)
       { dat.a = -0.009499F; dat.b = 8.355F;}
    if (dat.b==0.0F)
       {
       printf("\nError: Fitxer de calibrat calilno2.his incorrecte\n");
       exit(1);
       }
if (lee_calibrado( "c:\\fiacfa\\datos\\cali2no2.his", &dat.c, &dat.d)!=0)
       { dat.c = 0.20377F; dat.d = 76.2797F;}
    if (dat.d==0.0F)
       {
       printf("\nError: Fitxer de calibrat cali2no2.his incorrecte\n");
       exit(1);
       }
if (lee_calibrado( "c:\\fiacfa\\datos\\calibno3.his", &dat.e, &dat.f)!=0)
       { dat.e = -0.2976F; dat.f = 132.2938F;}
    if (dat.f==0.0F)
       {
       printf("\nError: Fitxer de calibrat calibno3.his incorrecte\n");
       exit(1);
       }
if (lee_calibrado( "c:\\fiacfa\\datos\\calibnh4.his",
                   &dat.ordenada, &dat.pendent)!=0)
       { dat.ordenada = -79.954F; dat.pendent = -66.296F;}
    if (dat.ordenada==0.0F || dat.pendent==0.0F)
       {
       printf("\nError: Fitxer de calibrat calibnh4.his incorrecte\n");
       exit(1);
       }

modo_grafico();
graf_base();
graf_presentacion();

reinicializa_puertob();

if (sincal == 1)
    {
      comunica(valv_inf_off,retorno);
      comunica(valv_sup_on,retorno);
      comunica2(amvalv_inf_off,retorno);
      comunica2(amvalv_sup_off,retorno);
    }

if (sincal == 0 && solono3==0)
    {
      comunica(valv_inf_on,retorno);   /*necessari per entrar i calibrar no2-*/
      comunica(valv_sup_off,retorno);
      comunica2(amvalv_inf_off,retorno);    /*necessari  per  entrar  passant  patr¢  50  ppm
amoni*/
      comunica2(amvalv_sup_on,retorno);
    }

if (solono3==1)
    {
      comunica(valv_inf_off,retorno);       /*necessari per entrar i calibrar no3-*/
      comunica(valv_sup_off,retorno);
      comunica2(amvalv_inf_off,retorno);    /*necessari per entrar mostra*/
      comunica2(amvalv_sup_off,retorno);
    }

reinicializa_puerto();
comunica(valv_inj,retorno);
delay(500);
comunica(load,retorno);           /*Posicio de c…rrega a l'inici*/
reinicializa_puertob();
comunica(pump_on, retorno);
comunica2(ampump_on,retorno);
```

```
presa_mostra(dat.punt_mostreig);  /* comencem amb posici¢ de mostreig de T0 */

if (sincal == 0 && solono3==0)
    {
    graf_texto("Espereu uns minuts, el sistema est… estabilitzant-se...",1,1,594,12,1,1);
    graf_texto("ATENCIO, si no voleu que m'enfadi, cal que ara mateix poseu el patr¢ de 3 PPM
NITRITS per calibrar el sistema.",1,13,594,24,0,1);
    graf_texto("NO MARXEU ENCARA, espereu a que aparegui el missatge: 'Introduiu patr¢ 7 PPM
NITRITS', que ja hauries de tenir preparat.",1,25,594,37,0,1);
    }

while(difftime(dat.t_actual,dat.t_estab)<=dat.t_estabilit)
      {
      dat.t_actual=time(NULL);
      selectmenu();
      }

reinicializa_puertob();
/*****
comunica(colorim[0], retorno);         /*Reset del colorimetre*/
/***delay(1000);*****/

/*Calibracio del colorimetre*/
/**  y una M.
      for (i=1;i<=3;i++)
   {
    delay(1000);
    comunica(colorim[1], retorno);
   }
****/
delay(8500);
reinicializa_puerto();
comunica(inject,retorno);  /*Injeccio a temps 0 */
reinicializa_puertob();

inicializar_tiempos();

dat.control=0;             /* permetem carrega del loop als calibrats */

if (sincal == 0 && solono3==0)
     {
     dat.posada_en_marxa=0;  /*calibrem amoni 50 amb nitrits pic 1
                          i calibrem amoni 5 amb nitrats sols a l'inici*/

     calibra_pic1_no2();      /*comencem calibrant el sistema*/
     calibra_pic2_no2();
     calibra_no3();
     graf_texto("Calibrats finalitzats",1,1,594,37,1,1);
     }

if (solono3==1)
     {
     dat.posada_en_marxa=1;  /* no calibrem amoni ni nitrits*/
     calibra_no3();
     graf_texto("Calibrat Nitrats finalitzat",1,1,594,37,1,1);
     }

/* En cas de reboot evita la posada en marxa amb calibrats */
obert=fopen("c:\\fiacfa\\tmp\\secur.tmp","wt");
fprintf(obert,"Hola\n");
fclose(obert);

}
```



```
/*****************************************************************
   DEFINICIONS DEL PROGRAMA PRINCIPAL DE GESTIO DELS ANALITZADORS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISIO: 4/1/99
*****************************************************************/

#include <time.h>
#include <bios.h>
#include <sys\timeb.h>
#include <stdio.h>
#include <graph.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <io.h>

#include "nombres.h"
#include "teclas.h"


void selectmenu(void);
void inicializar_tiempos(void);
void inicializar_variables(void);
void concentracio_amoni(void);
void concentracio_nitrits(void);
void concentracio_nitrats(void);
void acciones_comunes(void);
void inicia_sistema(void);

extern void modo_grafico(void);
extern void graf_base(void);
extern void graf_hora(void);
extern graf_presentacion(void);
extern void grafica_variable(int variable);
extern int  boton ( int on_off, char *nombre, short x1, short y1,
     short x2, short y2, short font);
extern int graf_texto (char *nombre, short x1, short y1,
     short x2, short y2, int font, int fondo);
extern void salir_al_dos(void);
extern void seleccionar_grafico(int orden);
extern void graf_valor_absor(void);
extern void graf_valor_mv(void);
extern void actualizar_matrices_graficas(void);
extern void graf_valores_display(void);
extern void graf_ayuda_general(void);

extern void reinicializa_puerto(void);
extern void reinicializa_puertob(void);
extern int comunica(int *frase, int *salida);
extern int comunica2(int *frase, int *salida);
extern int lectura_absor(void);
extern int lectura_ammv(void);

extern int calibra_pic1_no2(void);
extern int calibra_pic2_no2(void);
extern int calibra_no3(void);
extern int calibra_amoni(void);

extern void presa_mostra(int la_mostra);

extern void guarda_medida( char *mensaje, char *fichero);
extern void lectura_configuracio(void);
extern int lee_calibrado( char *fichero, float *param_a, float *param_b);

extern void delay( clock_t wait );

extern void medida_a_SUN(char parametro, char punto, float medida);

datos_fia     dat;
valores_fia   valor;

struct timeb t;       /* Estructura de temps en mil.lisegons */
int offline=0; /* Opciones de inicio */
int solono3=0;
int sincal=0;
int retorno[40];
```

```
float mesur_abs[20];
float suma;
int i;
int estado=0;
int estado2=0;
float max;
int contador =0;
float hpic1;
float hpic2;
char mensaje[200];

/****************************************
Instruccions per actuar sobre modul EASI
****************************************/

int valv_inj[20]={'\r','/','1','Z','6','R','\r','□'};        /*Inicialitza valvula injeccio*/
int pump_on[20]={'\r','0','1','M','O','1','\r','□'};         /* Posa bomba en marxa*/
int pump_off[20]={'\r','0','1','M','O','0','\r','□'};        /* Para bomba*/
int colorim[3][20]={{'\r','0','3','R','S','T','\r','□'},    /* Reset color¡metre*/
                    {'\r','0','3','T','C','A','\r','□'}, /* Calibraci¢ color¡metre*/
                    {'\r','0','3','R','D','I','\r','□'}}; /* Lectura color¡metre*/
int valv_inf_on[20]={'\r','0','2','V','I','1','\r','□'};     /* Encen v…lvula inferior*/
int valv_inf_off[20]={'\r','0','2','V','I','0','\r','□'};    /* Apaga v…lvula inferior*/
int load[15]={'\r','/','1','Y','2','A','1','R','\r','□'};    /* C…rrega del loop*/
int inject[15]={'\r','/','1','Y','1','A','2','R','\r','□'};  /* Injecci¢ de mostra*/
int valv_sup_on[20]={'\r','0','2','V','S','1','\r','□'};      /* Encen v…lvula inferior*/
int valv_sup_off[20]={'\r','0','2','V','S','0','\r','□'};     /* Apaga v…lvula inferior*/

int ampump_on[20]={'\r','1','1','M','O','1','\r','□'};              /* Posa bomba en marxa*/
int ampump_off[20]={'\r','1','1','M','O','0','\r','□'};               /* Para bomba*/
int amvalv_inf_on[20]={'\r','1','2','V','I','1','\r','□'};                 /*   Encen   v…lvula
inferior*/
int amvalv_inf_off[20]={'\r','1','2','V','I','0','\r','□'};                /*   Apaga   v…lvula
inferior*/
int amvalv_sup_on[20]={'\r','1','2','V','S','1','\r','□'};                 /*   Encen   v…lvula
inferior*/
int amvalv_sup_off[20]={'\r','1','2','V','S','0','\r','□'};        /* Apaga v…lvula inferior*/
int ampotenciometre[20]={'\r','1','5','R','D','I','\r','□'};       /* Lectura de mV*/
int   ampotenciometre1[20]={'\r','1','5','R','S','T','\r','□'};               /*   Reset   del
potenciometre*/
int amcorreccio[20]={'\r','1','5','T','U','P','\r','□'};           /* Tecla up */
```



```
/*****************************************************************************

   NOMBRES.H - DEFINICION DE ESTRUCTURAS COMUNES PARA EL PROGRAMA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 15/4/97

*****************************************************************************/


typedef struct{
       time_t      t_actual;
       time_t      t_cicle;
       time_t      t_inici;
       time_t      t_grafica;
       time_t      t_lectura;
       time_t      t_reloj;
       time_t      t_milli;
       time_t      t_milli_old;
       time_t      t_estab;
       int         fi;
       int         fi_programa;
       float       a;
       float       b;
       float       r1;
       float       c;
       float       d;
       float       r2;
       float       e;
       float       f;
       float       r3;
       float       ordenada;
       float       pendent;
       float       linea_base;
       int         t_estabilit;
       int         inici_lectura_base;
       int         fi_lectura_base;
       int         inj_entre_calibrats;
       int         inici_lectura_max1;
       int         inici_lectura_max2;
       int         t_carrega_loop;
       int         t_injeccio;
       int         t_canvi_punt_mostreig;
       int         t_lectura_amoni;
       int         delta_t[4];
       int         delta_pic1[3];
       int         delta_pic2[3];
       int         inici_lectura_base_calibratno3;
       int         fi_lectura_base_calibratno3;
       float       diferencia_entre_pics;
       int         posada_en_marxa;
       int         control;
       float       ammv5ppm;
       float       ammv50ppm;
       int         inicio_graf;
       int         fin_graf;
       int         punt_mostreig;
       int         numero_injeccio;
       int         numero_mostra_CFA;
       int         mostra_manual;
       int         conexion_SUN;
       int         repeticion;
}datos_fia;


typedef struct {                                /*Estructura   per   parametres   referents   als
controladors*/
       float       amoni;              /*Valor mesurat de la variable*/
       time_t      t_amoni;
       int         punt_amoni;
       float       nitrats;            /*Valor mesurat de la variable*/
       time_t      t_nitrats;
       int         punt_nitrats;
       float       nitrits;            /*Valor mesurat de la variable*/
       time_t      t_nitrits;
       int         punt_nitrits;
       float       absor;
```

```
        time_t      t_absor;
        float       mv;
        time_t      t_mv;
}valores_fia;

typedef struct {
        float       absor;
        float       mv;
        time_t      t_absor;
        time_t      t_mv;
}matriz_grafica;
```



```
/*****************************************************************************

   TECLAS.H - DEFINICION DE CODIGOS DE TECLADO

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 14/11/96

*****************************************************************************/

#define ESC       0x011B
#define CTRLC     0x2E03
#define CTRLQ     0x1011
#define CTRLBS    0x0E7F
#define CTRLRET   0x1C0A
#define ALTQ      0x1000
#define ALTP      0x1900
#define ALTS      0x1F00
#define ALTD      0x2000
#define ALTF      0x2100
#define ALTG      0x2200
#define ALTH      0x2300
#define ALTC      0x2E00
#define ALTN      0x3100
#define ALTM      0x3200

#define RETURN    0x1C0D
#define INTRO     0xE00D
#define BACKSP    0x0E08
#define TAB       0x0F09
#define BARRA     0x3920

#define AMUNT     0x48E0
#define ABAIX     0x50E0
#define DRETA     0x4DE0
#define ESQUERRA  0x4bE0

#define INSERT    0x52E0
#define SUPR      0x53E0
#define INICIO    0x47E0
#define FIN       0x4FE0
#define REPAG     0x51E0
#define AVPAG     0x49E0

#define F1        0x3B00
#define F2        0x3C00
#define F3        0x3D00
#define F4        0x3E00
#define F5        0x3F00
#define F6        0x4000
#define F7        0x4100
#define F8        0x4200
#define F9        0x4300
#define F10       0x4400
#define F11       0x8500
#define F12       0x8600
#define ALTF1     0x6800
#define ALTF2     0x6900
#define ALTF3     0x6A00
#define ALTF4     0x6B00
#define ALTF5     0x6C00
#define ALTF6     0x6D00
#define ALTF7     0x6E00
#define ALTF8     0x6F00
#define ALTF9     0x7000
#define ALTF10    0x7100

/*
#define o         0x186F
#define O         0x184F
#define p         0x1970
#define P         0x1950
#define m         0x326D
#define M         0x324D
#define t         0x1474
#define T         0x1454
#define e         0x1265
#define E         0x1245
```

```
#define a        0x1E61
#define A        0x1E41
#define r        0x1372
#define R        0x1352
#define h       0x2368
#define H        0x2348
#define d       0x2064
#define D       0x2044
*/

#define NUM1     0x0231
#define NUM2     0x0332
#define NUM3     0x0433
#define NUM1E    0x4F00
#define NUM2E    0x5000
#define NUM3E    0x5100
```



```
/*****************************************************************************

   CALCULO.C - FUNCIONES PARA REALIZAR CALCULOS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 9/11/98

*****************************************************************************/

#include <stdio.h>

/*#define TEST*/

int regresion_lineal( float *x, float *y, int n,
                      float *a, float *b, float *r);


/*****************************************************************************/
/* Calcula la regresi¢n lineal con los datos de x e y.
   Devuelve ordenada (a), pendiente (b) y coeficiente correlaci¢n (r)  */


int regresion_lineal( float *x, float *y, int n,
                      float *a, float *b, float *r)
{
    if (n<1) return -1;  /****temporal****/
/*
float sumx=0.0F,
      sumy=0.0F,
      sumxy=0.0F,
      sumx2=0.0F,
      sumy2=0.0F;
int i;

if (n<2) return -1;

for (i=0;i<n;i++)
    {
    sumx=sumx+x[i];
    sumy=sumy+y[i];
    sumxy=sumxy+x[i]*y[i];
    sumx2=sumx2+x[i]*x[i];
    sumy2=sumy2+y[i]*y[i];
    }

*b=(n*sumxy-sumx*sumy)/(n*sumx2-sumx*sumx);
*a=(sumy-(*b)*sumx)/n;
*r=(n*sumxy-sumx*sumy)*(n*sumxy-sumx*sumy)/
   ((n*sumx2-sumx*sumx)*(n*sumy2-sumy*sumy));
*/

/****temporal****/
/*Solo utiliza el punto de mayor concentraci¢n !!!!!!!!!*/

if (x[0]==0.0F) return -2;

*b=y[0]/x[0];
*a=0.0F;
*r=1.0F;

return 1;
}

/*****************************************************************************/
#ifdef TEST
void main(void)
{
float misx[5]={1.0F,2.0F,3.0F,4.0F,5.0F};
float misy[5]={2.0F,4.0F,6.0F,8.0F,10.0F};
float a,b,r;

regresion_lineal( misx, misy, 5, &a, &b, &r);

printf("\ta: %f\tb: %f\tr: %f\n",a,b,r);


}
```

```
#endif
/*****************************************************************************/
```



```
/*****************************************************************************

   CALIBRA.C - FUNCIONES PARA EL CALIBRADO DEL SISTEMA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 4/1/99

*****************************************************************************/

/********************** Librerias  a incluir ************************/

#include <time.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <math.h>
#include <string.h>
#include <stdio.h>
#include "nombres.h"

/************************ Definiciones *********************************/

/********************** Subrutinas del fichero **********************/

int calibra_pic1_no2(void);
int calibra_pic2_no2(void);
int calibra_no3(void);
int calibra_amoni(void);
void recta_amoni(void);

/********************* Subrutinas externas necesarias *****************/

extern int comunica(int *frase, int *salida);            /* Comunicacio EASI-PC*/
extern int comunica2(int *frase, int *salida);
extern int regresion_lineal( float *x, float *y, int n,
                     float *a, float *b, float *r);
extern void reinicializa_puerto(void);
extern void reinicializa_puertob(void);
extern void delay( clock_t wait );
extern int lectura_absor(void);
extern int lectura_ammv(void);
extern void acciones_comunes(void);
extern int graf_texto (char *nombre, short x1, short y1,
          short x2, short y2, int font, int fondo);
extern void concentracio_amoni(void);
extern void concentracio_nitrits(void);
extern void concentracio_nitrats(void);
extern void guarda_medida( char *mensaje, char *fichero);
extern void graf_valores_display(void);

/****************** Variables comunes del programa ******************/

extern int
   valv_inj[20], pump_on[20], pump_off[20], colorim[3][20],
   valv_inf_on[20], valv_inf_off[20], load[15], inject[15],
   valv_sup_on[20], valv_sup_off[20], ampump_on[20], ampump_off[20],
   amvalv_inf_on[20], amvalv_inf_off[20], amvalv_sup_on[20],
   amvalv_sup_off[20], ampotenciometre[20], ampotenciometre1[20],
   amcorreccio[20];

extern datos_fia      dat;
extern valores_fia    valor;
float hpic;
extern float hpic1;
extern float hpic2;

/*****************************************************************************
                    Calibraci¢ pic 3 ppm de nitrit
*****************************************************************************/

int calibra_pic1_no2(void)
{
  int i;
  float mesur_abs[20];
  float suma;
  int estado=0;
  float max=0.0F;
  int contador=0;
```

```
  time_t t_maxim=0;
  struct timeb t;     /*Estructura de temps amb mil.lisegons*/
  float h_pic[4]={0.0F,0.0F,0.0F,0.0F};
  float c_pic[4]={3.0F,2.0F,1.0F,0.5F};
  int retorno[40];
  int pseudo_pico=0;
  char mensaje[200];

dat.mostra_manual=1; /*No envia datos a SUN*/

graf_texto("Calibraci¢ electr•nica PIC 1 DE NITRITS...",1,1,594,37,1,1);

for (i=0;i<20;i++) mesur_abs[i]=0.0F;

do
{
      dat.t_actual=time(NULL);                  /*Actualitzaci¢ dels temps */
      ftime(&t);
      dat.t_milli=t.time*1000+t.millitm;
      acciones_comunes();

/* Valor del patr¢ de 50 ppm amoni
      ( calculat sols a l'inici del programa )*/
      /*Llegit 1 minut abans de tornar a injectar */

      if((difftime(dat.t_actual,dat.t_inici)>=(dat.t_injeccio-60))
         && (difftime(dat.t_actual,dat.t_inici)<=dat.t_injeccio)
         && (dat.posada_en_marxa==0))
      {
       comunica2(amvalv_sup_on,retorno);   /* canviem a patr¢ de 5 ppm amoni*/
       comunica2(amvalv_inf_on,retorno);
       if (lectura_ammv()==0)
           {
           dat.ammv50ppm= valor.mv;
           valor.punt_amoni=0;
           concentracio_amoni();
           dat.posada_en_marxa=1;
           }
      }

/************ Calibrat ***************/

switch(estado){
 case 0:     /* Calcul linia base */
      if ((difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_base)
       && (difftime(dat.t_actual,dat.t_inici)<=dat.fi_lectura_base))
      {
        if (difftime(dat.t_milli,dat.t_milli_old)>=200)
        {
        dat.t_milli_old=t.time*1000+t.millitm;
        if (lectura_absor()==0)
           {
           for (i=0;i<=18; i++) mesur_abs[i]=mesur_abs[i+1];
           mesur_abs[19]=valor.absor;
           suma=0.0F;
           for (i=0; i<=19; i++) suma= suma + mesur_abs[i];
           dat.linea_base=suma/20.0F;
           }
        }
      }
    if (difftime(dat.t_actual,dat.t_inici)>dat.fi_lectura_base)
       {
       sprintf(mensaje,"Linia base calculada: %5.3f ",dat.linea_base);
       graf_texto(mensaje,1,1,594,37,1,1);
       estado++;
       }
 break;
 case 1:   /* Carrega del loop */
      if (difftime(dat.t_actual,dat.t_inici)>=dat.t_carrega_loop)
        {
        reinicializa_puerto();
        comunica(load,retorno);
        reinicializa_puertob();
        estado++;
        }
 break;
 case 2:
```



```
     if (difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_max1) estado++;
 break;
 case 3:     /* Inici de la lectura del primer m…xim */
     if (difftime(dat.t_milli,dat.t_milli_old)>=200)
          {
          dat.t_milli_old=t.time*1000+t.millitm;
          if(lectura_absor()!=0) break; /* Si no ha llegit b,, surt del switch*/
          if (valor.absor>max)
               {
               max=valor.absor;
               contador=0;
               t_maxim=dat.t_actual;
               }
          else (contador++);
          if(contador>20)
               {
               h_pic[0]=max-dat.linea_base;
               sprintf(mensaje,"Pic 1 de nitrits 3 ppm detectat, hpic = %6.4f ",h_pic[0]);
               graf_texto(mensaje,1,1,594,37,1,1);
               estado++;
               }
          }
 break;
 case 4:   /* Lectura de los pseudo picos */
    /************ Desactivamos el calibrado electronico
     if (difftime(dat.t_actual,t_maxim+dat.delta_pic1[pseudo_pico])>=0)
       {
       if (lectura_absor()==0)
            {
            h_pic[pseudo_pico+1]=valor.absor-dat.linea_base;
            pseudo_pico++;
            if (pseudo_pico>=3) estado++;
            }
       }
    ***********************************************/
    estado++;
 break;
 case 5:
    regresion_lineal( h_pic, c_pic, 4, &dat.a, &dat.b, &dat.r1);
    sprintf(mensaje,"%6.4f\t%6.4f\t%6.4f\t",dat.a,dat.b,dat.r1);
    guarda_medida( mensaje, "c:\\fiacfa\\datos\\calilno2.his");

    hpic=h_pic[0];

    hpic1=hpic;
    concentracio_nitrits();

    sprintf(mensaje,"%6.4f \t%6.4f \t%6.4f \t%6.4f \t",h_pic[0],h_pic[1],h_pic[2],h_pic[3]);
    guarda_medida( mensaje, "c:\\fiacfa\\datos\\hno21.his");

    estado++;
 break;
 case 6:   /* Injecci¢ nova mostra */
      if (difftime(dat.t_actual,dat.t_inici)>=dat.t_injeccio)
         {
         dat.t_inici=time(NULL);
         reinicializa_puerto();
         comunica(inject,retorno);
         reinicializa_puertob();
         estado++;
         }
 break;
 }/*Fin switch*/

}  while(estado!=7 && dat.fi==0);

dat.mostra_manual=0; /*Envia datos a SUN*/

dat.fi=0;
dat.posada_en_marxa=0;

return 0;
}

/*****************************************************************************
                    Calibraci¢ pic 7 ppm de nitrit
*****************************************************************************/
```

```
int calibra_pic2_no2(void)
{
  int estado=0;
  float max=0.0F;
  int contador=0;
  time_t t_maxim=0;
  struct timeb t;     /*Estructura de temps en mil.lisegons*/
  float h_pic[4]={0.0F,0.0F,0.0F,0.0F};
  float c_pic[4]={7.0F,5.0F,3.0F,1.0F};
  int retorno[40];
  int pseudo_pico=0;
  char mensaje[200];

dat.mostra_manual=1; /*No envia datos a SUN*/

graf_texto("Calibraci¢ electr•nica PIC 2 DE NITRITS...",1,1,594,37,1,1);

do
{
      dat.t_actual=time(NULL);                  /*Actualitzaci¢ dels temps */
      ftime(&t);
      dat.t_milli=t.time*1000+t.millitm;

      acciones_comunes();

/************ Calibrat ***************/

switch(estado){
 case 0:     /* Calcul linia base */
    /* Asumim que es la mateixa del calibrat anterior */
      estado++;
 break;
 case 1:   /* Carrega del loop */
      if (difftime(dat.t_actual,dat.t_inici)>=dat.t_carrega_loop)
        {
        reinicializa_puerto();
        comunica(load,retorno);
        reinicializa_puertob();
        if(dat.posada_en_marxa==1)
          {
            comunica(valv_sup_on,retorno);
            comunica(valv_inf_off,retorno);
          }
        if(dat.posada_en_marxa==0) /*per tenir temps de carregar patr¢ NO3- sols a l'inici del
programa*/
          {
            comunica(valv_sup_off,retorno);
            comunica(valv_inf_off,retorno);
          }
        estado++;
        }
 break;
 case 2:
     if (difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_max2) estado++;
 break;
 case 3:     /* Inici de la lectura del primer m…xim */
     if (difftime(dat.t_milli,dat.t_milli_old)>=200)
          {
          dat.t_milli_old=t.time*1000+t.millitm;
          if(lectura_absor()!=0) break; /* Si no ha llegit b,, surt del switch*/
          if (valor.absor>max)
               {
               max=valor.absor;
               contador=0;
               t_maxim=dat.t_actual;
               }
          else (contador++);
          if(contador>20)
               {
               h_pic[0]=max-dat.linea_base;
               sprintf(mensaje,"Pic 2 de nitrits 7 ppm detectat, hpic = %6.4f ",h_pic[0]);
               graf_texto(mensaje,1,1,594,37,1,1);
               estado++;
               }
          }
 break;
```



```
 case 4:
    /************ Desactivamos el calibrado electronico
     if (difftime(dat.t_actual,t_maxim+dat.delta_pic2[pseudo_pico])>=0)
        {
          if (lectura_absor()==0)
              {
              h_pic[pseudo_pico+1]=valor.absor-dat.linea_base;
              pseudo_pico++;
              if (pseudo_pico>=3) estado++;
              }
        }
    ***********************************************/
    estado++;
 break;
 case 5:
    regresion_lineal( h_pic, c_pic, 4, &dat.c, &dat.d, &dat.r2);
    sprintf(mensaje,"%6.4f \t%6.4f \t%6.4f \t",dat.c,dat.d,dat.r2);
    guarda_medida( mensaje, "c:\\fiacfa\\datos\\cali2no2.his");

    sprintf(mensaje,"%6.4f\t%6.4f\t%6.4f\t%6.4f\t",h_pic[0],h_pic[1],h_pic[2],h_pic[3]);
    guarda_medida( mensaje, "c:\\fiacfa\\datos\\hno22.his");

    estado++;
 break;
 case 6:   /* Injecci¢ nova mostra */
      if (difftime(dat.t_actual,dat.t_inici)>=dat.t_injeccio)
          {
          dat.t_inici=time(NULL);
          reinicializa_puerto();
          comunica(inject,retorno);
          reinicializa_puertob();
          estado++;
          }
 break;
 }/*Fin switch*/

}  while(estado!=7 && dat.fi==0);

dat.mostra_manual=0; /*Envia datos a SUN*/

dat.fi=0;

return 0;
}

/*****************************************************************************
                    Calibraci¢ electronica de nitrats          ******************/

int calibra_no3(void)
{
  float mesur_abs[20];
  float suma;
  int i;
  int estado=0;
  float max=0.0F;
  int contador=0;
  time_t t_maxim=0;
  struct timeb t;     /*Estructura de temps en mil.lisegons*/
  float h_pic[5]={0.0F,0.0F,0.0F,0.0F,0.0F};
  float c_pic[5]={50.0F,30.0F,15.0F,5.0F,1.0F};
  float h_pic_old=0;  /* Variable per comprobar la repetitivitat del pic */
  int retorno[40];
  int pseudo_pico=0;
  char mensaje[200];

dat.mostra_manual=1; /*No envia datos a SUN*/
graf_texto("Calibraci¢ electr•nica de NITRATS...",1,1,594,37,1,1);

ftime(&t);
dat.t_milli=t.time*1000+t.millitm;
dat.t_milli_old=dat.t_milli;
for (i=0;i<20;i++) mesur_abs[i]=0.0F;

do
{
      dat.t_actual=time(NULL);                  /*Actualitzaci¢ dels temps */
      ftime(&t);
```

```
      dat.t_milli=t.time*1000+t.millitm;

      acciones_comunes();

/* Valor del patr¢ de 5 ppm amoni */
      /*Llegim quan calibrem nitrats sols a l'arrencar */

      if((difftime(dat.t_actual,dat.t_inici)>=60)
         && (difftime(dat.t_actual,dat.t_inici)<=80)
         && (dat.posada_en_marxa==0))
      {
       if (lectura_ammv()==0)
           {
           comunica2(amvalv_sup_off,retorno);       /* canviem a mostra real*/
           comunica2(amvalv_inf_off,retorno);
           dat.ammv5ppm= valor.mv;
           valor.punt_amoni=0;
           dat.posada_en_marxa=1;
            /*Evita lectura amoni en calibrats de nitrats excepte a l'inici*/
           concentracio_amoni();
           recta_amoni();
           sprintf(mensaje,"%6.4f\t%6.4f\t",dat.ordenada,dat.pendent);
           guarda_medida( mensaje, "c:\\fiacfa\\datos\\calibnh4.his");
           }
      }

/* Carrega del loop */

      if((difftime(dat.t_actual,dat.t_inici)>=dat.t_carrega_loop)
         && (dat.control==0))
      {
         reinicializa_puerto();
         comunica(load,retorno);
         reinicializa_puertob();
         dat.control=1;
      }

      if (difftime(dat.t_actual,dat.t_inici)>=dat.t_injeccio)
      {
         dat.t_inici=time(NULL);
         dat.t_cicle=time(NULL);
         reinicializa_puerto();
         comunica(inject,retorno);
         reinicializa_puertob();
         dat.control=0;
         if (estado==6) estado++;
           /** El calibrat ha finalitzat i compleix la repetitivitat */
         else if (estado == -1) estado=0; /*Vuelve a calibrar*/
      }

/************ Calibrat ***************/

switch(estado){
 case 0:     /* Calcul linia base */
      if ((difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_base_calibratno3)
       && (difftime(dat.t_actual,dat.t_inici)<=dat.fi_lectura_base_calibratno3))
      {
        if (difftime(dat.t_milli,dat.t_milli_old)>=200)
        {
        dat.t_milli_old=t.time*1000+t.millitm;
        if (lectura_absor()==0)
           {
           for (i=0;i<=18; i++) mesur_abs[i]=mesur_abs[i+1];
           mesur_abs[19]=valor.absor;
           suma=0.0F;
           for (i=0; i<=19; i++) suma= suma + mesur_abs[i];
           dat.linea_base=suma/20.0F;
           }
        }
      }
    if (difftime(dat.t_actual,dat.t_inici)>dat.fi_lectura_base_calibratno3)
       {
       sprintf(mensaje,"Linia base calculada: %5.3f ",dat.linea_base);
       graf_texto(mensaje,1,1,594,37,1,1);
       estado++;
       }
 break;
```



```
case 1:
    if (difftime(dat.t_actual,dat.t_inici)>=dat.inici_lectura_max2)
      {
      estado++;
      graf_texto("Iniciem detecci¢ Pic de nitrats",1,1,594,37,1,1);
      }
break;
case 2:     /* Inici de la lectura del segon m…xim */
    if (difftime(dat.t_milli,dat.t_milli_old)>=200)
        {
        dat.t_milli_old=t.time*1000+t.millitm;
        if(lectura_absor()!=0) break; /* Si no ha llegit b,, surt del switch*/
        if (valor.absor>max)
            {
            max=valor.absor;
            contador=0;
            t_maxim=dat.t_actual;
            }
        else (contador++);
        if(contador>20)
            {
            h_pic[0]=max-dat.linea_base;
            max=0.0F;
            sprintf(mensaje,"Pic de nitrats 50 ppm detectat, hpic = %6.4f ",h_pic[0]);
            graf_texto(mensaje,1,1,594,37,1,1);
            delay(1000);
            contador=0;
            estado++;
            }
        }
break;
case 3:
    if ((float)fabs(h_pic[0]-h_pic_old)<=dat.diferencia_entre_pics)
      {
      comunica(valv_sup_on,retorno);        /* Per entrar mostra real */
      comunica(valv_inf_off,retorno);
      graf_texto("Pic de nitrats repetitiu",1,1,594,37,1,1);
      estado++;
      }
    else
      {
      h_pic_old=h_pic[0];
      graf_texto("Pic de nitrats no repetitiu",1,1,594,37,1,1);
      estado =-1; /* El pic no ,s repetitiu,
                     s'ha de repetir el proc,s en una injecci¢ posterior */
      }
    sprintf(mensaje," %6.4f ",h_pic[0]);
    guarda_medida( mensaje, "c:\\fiacfa\\datos\\hpicno3.his");

    hpic=h_pic[0];
    valor.nitrats= dat.e+dat.f*((dat.d*hpic-0.0F+dat.c)/dat.d);
    valor.t_nitrats=time(NULL);
    valor.punt_nitrats=0;
    graf_valores_display();
    pseudo_pico=0;      /* Reinicialitzaci¢ de variables */

break;
case 4:
   /************ Desactivamos el calibrado electronico
    if (difftime(dat.t_actual,t_maxim+dat.delta_t[pseudo_pico])>=0)
      {
      if (lectura_absor()==0)
          {
          sprintf(mensaje,"Detectando pseudo pico: %i ",pseudo_pico);
          graf_texto(mensaje,1,1,594,37,1,1);
          h_pic[pseudo_pico+1]=valor.absor-dat.linea_base;
          pseudo_pico++;
          if (pseudo_pico>=4) estado++;
          }
      }
   ***********************************************/
   estado++;
break;
case 5:
   regresion_lineal( h_pic, c_pic, 4, &dat.e, &dat.f, &dat.r3);
   sprintf(mensaje,"%6.4f \t%6.4f \t%6.4f \t",dat.e,dat.f,dat.r3);
   guarda_medida( mensaje, "c:\\fiacfa\\datos\\calibno3.his");
```

```
    graf_texto("Nou calibrat de nitrats calculat",1,1,594,37,1,1);

sprintf(mensaje,"%6.4f\t%6.4f\t%6.4f\t%6.4f\t%6.4f\t",h_pic[0],h_pic[1],h_pic[2],h_pic[3],h_pi
c[4]);
    guarda_medida( mensaje, "c:\\fiacfa\\datos\\hno3.his");

    estado++;
 break;
 }/*Fin switch*/

}  while(estado!=7 && dat.fi==0);

dat.mostra_manual=0; /*Envia datos a SUN*/
dat.fi=0;
return 0;
}

/*************************************************************************
                          Calibraci¢ d'amoni          ******************/

int calibra_amoni(void)
{
int estado=0;
int retorno[40];
char mensaje[200];

graf_texto("Calibraci¢ d' AMONI...",1,1,594,37,1,1);
dat.mostra_manual=1; /*No envia datos a SUN*/

do
{
      dat.t_actual=time(NULL);                /*Actualitzaci¢ dels temps */

      acciones_comunes();

/************* Calibrat ***************/

switch(estado){
  case 0: /** Canvi per fer passar patr¢ 5 ppm **/
    if(difftime(dat.t_actual,dat.t_cicle)>=660)
      {
      comunica2(amvalv_sup_on,retorno);
      comunica2(amvalv_inf_on,retorno);
      estado++;
      }
  break;
  case 1: /** Lectura CFA per patr¢ 50 ppm als 11 minuts **/
    if( difftime(dat.t_actual,dat.t_cicle)>=665 )
      {
      if (lectura_ammv()==0)
            {
            dat.ammv50ppm= valor.mv;
            valor.punt_amoni=0;
            concentracio_amoni();
            estado++;
            }
      }
  break;
  case 2: /** Canvi per fer passar mostra al sortir de la subrutina **/
    if( difftime(dat.t_actual,dat.t_cicle)>=1680 )
      {
      comunica2(amvalv_sup_off,retorno);
      comunica2(amvalv_inf_off,retorno);
      estado++;
      }
  break;
  case 3: /** Lectura CFA per patr¢ 5 ppm als 30 minuts **/
    if((difftime(dat.t_actual,dat.t_cicle)>=1790))
      {
      if (lectura_ammv()==0)
            {
            dat.ammv5ppm= valor.mv;
            valor.punt_amoni=0;
            concentracio_amoni();
            recta_amoni();
            sprintf(mensaje,"%6.4f \t%6.4f \t",dat.ordenada,dat.pendent);
```



```
            guarda_medida( mensaje, "c:\\fiacfa\\datos\\calibnh4.his");
            estado++;
            }
      }
  break;
  case 4:
    if( difftime(dat.t_actual,dat.t_cicle)>=1800 )
      {
      comunica2(amvalv_sup_off,retorno);     /* canviem a mostra real*/
      comunica2(amvalv_inf_off,retorno);
      dat.t_cicle=time(NULL);
      dat.t_inici=time(NULL);
      reinicializa_puerto();
      comunica(inject,retorno);     /* Injeccio per FIA */
      reinicializa_puertob();
      estado++;
      }
  break;
 }/*Fin switch*/

}  while(estado!=5 && dat.fi==0);

dat.mostra_manual=0; /*Envia datos a SUN*/

dat.fi=0;
return 0;
}

/*************************************************************************/

void recta_amoni(void)
{
float amoni[2],
      h_amoni[2];

amoni[0]=(float)log10(50.0F);
amoni[1]=(float)log10(5.0F);
h_amoni[0]=dat.ammv50ppm;
h_amoni[1]=dat.ammv5ppm;

/* Y1=a+b*X1, Y2=a+b*X2. b=(Y1-Y2)/(X1-X2). a=Y1-b*X1. */

dat.pendent = (h_amoni[0]-h_amoni[1])/(amoni[0]-amoni[1]);
dat.ordenada = h_amoni[0]-dat.pendent*amoni[0];

}
```

```
/*****************************************************************************

   CLIENTE.C - FUNCIONES PARA LA COMUNICACION CON LA SUN

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 28/8/97

*****************************************************************************/

/********************* Librerias  a incluir ***************************/

#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <process.h>
#include <stdlib.h>
#include <io.h>
#include <conio.h>
#include <time.h>
#include "nombres.h"

#include "c:\sockets\devkit\include\pctcp\types.h"
#include "c:\sockets\devkit\include\sys\types.h"
#include "c:\sockets\devkit\include\sys\ioctl.h"
#include "c:\sockets\devkit\include\pctcp\errno.h"
#include "c:\sockets\devkit\include\sys\socket.h"
#include "c:\sockets\devkit\include\netinet\in.h"
#include "c:\sockets\devkit\include\netdb.h"
/*#include "c:\sockets\devkit\include\arpa\inet.h"*/

#include "c:\sockets\devkit\include\pctcp\pctcp.h"
#include "c:\sockets\devkit\include\pctcp\ipconfig.h"
#include "c:\sockets\devkit\include\pctcp\sockets.h"
#include "c:\sockets\devkit\include\pctcp\options.h"
#include "c:\sockets\devkit\include\fcntl.h"

#include <sys\time.h>
/************************ Definiciones ******************************/

/*#define DEPURA*/
/*#define TEST*/

/*#define svrName "eq3.uab.es"  */      /* Direcci¢n SUN */
#define svrName "158.109.13.2"          /* Direcci¢n num,rica SUN */
#define Port (ushort) 32323             /* Numero de puerto del socket */
#define TAMANY 512                      /* Tama¤o del buffer */
#define TAMANY_M   100                  /* Tama¤o del mensaje cliente */

typedef unsigned short ushort;          /* Definicion de tipo de variables */
typedef unsigned long ulong;

/************************ Subrutinas del fichero ************************/

int sockets_pasa_datos(void);
void medida_a_SUN(char parametro, char punto, float medida);
int mensaje_a_SUN(char *mensaje);
int connectToServer(char *serverName, ushort port);
int disconnectToServer(int toServerSocket);

void error_send(int errno);
void error_socket(int errno);
void error_connect(int errno);

#ifdef TEST
void main(void);
#define extern
#endif

/********************* Subrutinas externas necesarias ******************/

extern ulong inet_addr(char *servidor); /**JB**/
extern int close(int elsocket);         /**JB**/
extern int variar_velocidad(double velocidad, int variador);
extern void guarda_mensaje( char *mensaje);
extern void imprime(char *mensaje);

/******************** Variables comunes del programa *******************/
```



```
char buffer[TAMANY];              /* Variable a transmitir o recibir */

extern datos_fia     dat;
extern valores_fia   valor;

/*****************************************************************************/
/* Funci¢n para establecer la comunicaci¢n y controlar el env¡o y la
   recepci¢n de datos */


int sockets_pasa_datos(void)
{
int toServerSocket = -1;
int i;                            /* Variables de control */
char caracter[1]={'A'};

/* Si la conexi¢n anterior no ha sido correcta, vuelve */
if (dat.conexion_SUN == 0) return -2;

/* Conexi¢n con la SUN */
toServerSocket = connectToServer(svrName, Port);

/* Si la conexi¢n no es correcta, devuelve -1 */
if (toServerSocket==-1)
    {
#ifdef DEPURA
    printf ("Error en la conexi¢n\n");
#endif
    dat.conexion_SUN=0;
    return -1;
    }

#ifdef DEPURA
printf("CONEXIO A %s REALITZADA, %i\n",svrName,toServerSocket);
#endif

/* Mandamos una A, queremos establecer los valores de los analizadores */
send(toServerSocket, caracter, sizeof(caracter), 0);

/* Mandamos el buffer */
i = send(toServerSocket, buffer, sizeof(buffer), 0);
#ifdef DEPURA
if (i==-1) error_send(errno);
#endif

/* Recibimos respuesta */
recv(toServerSocket, caracter, sizeof(caracter), 0);
/* Lo env¡a de nuevo si no ha sido bien recibido */
if (caracter[0]!='O')
    {
#ifdef DEPURA
    printf("Recibimos mal\n");
#endif
    caracter[0]='A';
    send(toServerSocket, caracter, sizeof(caracter), 0);

    send(toServerSocket, buffer, sizeof(buffer), 0);
    recv(toServerSocket, caracter, sizeof(caracter), 0);
    if (caracter[0]!='O')       /* Si lo recibe mal de nuevo, desconecta */
       {
#ifdef DEPURA
printf("Recibimos mal de nuevo\n");
#endif
       disconnectToServer(toServerSocket);
       return -3;
       }
    }

/* Pedimos desconexi¢n */
caracter[0]='F';
send(toServerSocket, caracter, sizeof(caracter), 0);
disconnectToServer(toServerSocket);

return 0;
}
```

```
/*************************************************************************/
/* Introduce los datos de la estructura medidas en el buffer,
   a¤adiendo un cheksum y llama a otra funci¤n que manda los datos */

void medida_a_SUN(char parametro, char punto, float medida)
{
struct sexta{
       char punto;     /* 0 R-Anaerobio, 1 R-1, 2 R-2, 3 R-3, 4 Salida */
       char parametro;  /* 1 nitrato, 2 nitrito, 3 amonio */
        char valor[10];
}dato_analisis;

int i;
unsigned long chequeo=0L;

memset(buffer, 0, sizeof(buffer));   /* Borra las estructuras */
memset(&dato_analisis, 0, sizeof(dato_analisis));

/* Introducimos las medidas en la estructura que pasamos a la SUN */

dato_analisis.parametro = parametro;
dato_analisis.punto = punto;
sprintf(dato_analisis.valor,"%7.4f", medida);

/* Copiamos la estructura de datos al buffer */
memcpy(buffer, &dato_analisis, sizeof(dato_analisis));

/* Calculamos el cheksum del buffer */
for (i=0;i<(TAMANY-12);i++) chequeo+=buffer[i];

/* Introducimos el cheksum como cadena de caracteres en la posici¤n 500 */
sprintf(&buffer[500],"%lllu",chequeo);

/* Pasamos los datos a la SUN */
sockets_pasa_datos();

}

/*************************************************************************/

int mensaje_a_SUN(char *mensaje)
{
int toServerSocket = -1;
int i;                          /* Variables de control */
unsigned long chequeo=0L;
char caracter[1]={'M'};
time_t ltime;
char mensaje_cliente[TAMANY_M];

/*if (dat.conexion_SUN == 0) return -2;*/

memset(buffer, 0, sizeof(buffer));   /* Borra las estructuras */
memset (mensaje_cliente, 0, sizeof(mensaje_cliente));

time (&ltime);
sprintf(mensaje_cliente,"%s%s", mensaje, ctime( &ltime ));

/* Copiamos el mensaje al buffer */
memcpy(buffer, mensaje_cliente, sizeof(mensaje_cliente));

/* Calculamos el cheksum del buffer */
for (i=0;i<(TAMANY-12);i++) chequeo+=buffer[i];

/* Introducimos el cheksum como cadena de caracteres en la posici¤n 500 */
sprintf(&buffer[500],"%lllu",chequeo);


/* Conexi¤n con la SUN */
toServerSocket = connectToServer(svrName, Port);

/* Si la conexi¤n no es correcta, devuelve -1 */
if (toServerSocket==-1)
    {
#ifdef DEPURA
    printf ("Error en la conexi¤n\n");
#endif
    dat.conexion_SUN=0;
```



```
    return -1;
    }
else dat.conexion_SUN=1;

#ifdef DEPURA
printf("CONEXIO A %s REALITZADA, %i\n",svrName,toServerSocket);
#endif

/* Mandamos una M, queremos mandar mensaje */
send(toServerSocket, caracter, sizeof(caracter), 0);

/* Mandamos el buffer */
send(toServerSocket, buffer, sizeof(buffer), 0);

/* Recibimos respuesta */
recv(toServerSocket, caracter, sizeof(caracter), 0);
/* Lo env¡a de nuevo si no ha sido bien recibido */
if (caracter[0]!='O')
    {
    caracter[0]='M';
    send(toServerSocket, caracter, sizeof(caracter), 0);

    send(toServerSocket, buffer, sizeof(buffer), 0);
    recv(toServerSocket, caracter, sizeof(caracter), 0);
    if (caracter[0]!='O')       /* Si lo recibe mal de nuevo, desconecta */
       {
       disconnectToServer(toServerSocket);
       return -3;
       }
    }

/* Pedimos desconexi¤n */
caracter[0]='F';
send(toServerSocket, caracter, sizeof(caracter), 0);
disconnectToServer(toServerSocket);

return 1;
}


/*************************************************************************/
/*************************************************************************/
/*  connectToServer es usada por el cliente para abrir una conexi¤n     */
/*  con el servidor                                                      */
/*************************************************************************/

int connectToServer(char *serverName, ushort port)
{
struct sockaddr_in serverSockAddr;    /* Variable direcci¤n del host remoto */
struct hostent *serverHostEnt;        /* Puntero direcci¤n del host remoto */
int toServerSocket = (-1);            /* Descriptor del socket local */
ulong hostAddr;                       /* Direcci¤n del host */
int result = (-1);                    /* Supone fallo */
int error=0;
struct linger{
    int l_onoff;
    int l_linger;
}mi_linger;

/*************************************************************************/
/*      Borra y establece la estructura de la direcci¤n del servidor     */

memset(&serverSockAddr, 0, sizeof(serverSockAddr) );

hostAddr = inet_addr(serverName);       /* Devuelve la direcci¤n */
if((long)hostAddr != (long)(-1))        /* Hay direcci¤n */
       {
#ifdef DEPURA
printf("\nconnectToServer: se ha obtenido la direcci¤n de %s i es %lx \n",  serverName,
hostAddr);
#endif
       memcpy( &(serverSockAddr.sin_addr), &hostAddr,sizeof(hostAddr));
       }
       else                                    /* No hay direcci¤n */
       {
#ifdef DEPURA
printf("\nconnectToServer: no se ha obtenido directamente la direcci¤n \n" );
```

```
#endif

/* Se consulta al servidor de nombres para determinar la direcci¢n del host */

       serverHostEnt =  gethostbyname(serverName); /* Asignaci¢n de la direcci¢n remota */
       if(NULL == serverHostEnt)
            {
#ifdef DEPURA
            fprintf(stderr, "Imposible localizar host \" %s \"\n", serverName);
#endif
            goto egress;
            }
/* Se copia la direcci¢n de entrada del host a la estructura del socket */

       memcpy(&(serverSockAddr.sin_addr), serverHostEnt->h_addr,serverHostEnt->h_length);
       };

serverSockAddr.sin_family = AF_INET;           /* Asignaci¢n de direcciones */
serverSockAddr.sin_port = (ushort)htons(port); /* Asignaci¢n del puerto */

/*************************************************************************/
/*      Se crea el socket                                                */

error = 0;

toServerSocket = socket( AF_INET, SOCK_STREAM, 0);
if (toServerSocket == -1)
       {
#ifdef DEPURA
       error_socket(errno);
#endif
       goto egress;
       }
#ifdef DEPURA
printf("El socket vale %i \n",toServerSocket);
#endif

if (setsockopt (toServerSocket, SOL_SOCKET, SO_REUSEADDR, "sisi", 4)==-1)
    printf ("Error en setsockopt, reusa\n");

mi_linger.l_onoff=1;
mi_linger.l_linger=10;

if (setsockopt (toServerSocket, SOL_SOCKET, SO_LINGER, (char *) &mi_linger,
              sizeof(mi_linger))==-1)
    printf ("Error en setsockopt, linger\n");

if (connect(toServerSocket, &serverSockAddr, sizeof(serverSockAddr)) == -1)
    {
    error = 1;
#ifdef DEPURA
    error_connect(errno);
#endif
    goto egress;
    }
result = toServerSocket;

egress:
return result;
}

/*************************************************************************/
/* Funci¢n para cerrar la conexi¢n con el servidor */

int disconnectToServer(int toServerSocket)
{
int shut,clo;

shut=shutdown(toServerSocket, 2);
clo=close(toServerSocket);

#ifdef DEPURA
if (shut== -1)   printf(" Error en shutdown toServerSocket.\n");
else             printf("Shutdown toServerSocket ok.\n");

if(clo == -1)    printf("Error en close toServerSocket.\n");
#endif
```



```
return clo;
}

/*************************************************************************/

void error_send(int errno)
{
switch(errno)
              {
              case EACCES:
              printf("ERROR: EACCES\n");
              break;
              case EFAULT:
              printf("ERROR: EFAULT\n");
              break;
              case ENOBUFS:
              printf("ERROR: ENOBUFS\n");
              break;
              case ENOTCONN:
              printf("ERROR: ENOTCONN\n");
              break;
              case ENOTSOCK:
              printf("ERROR: ENORSOCK\n");
              break;
              case ESHUTDOWN:
              printf("ERROR: ESHUTDOWN\n");
              break;
              case EWOULDBLOCK:
              printf("ERROR: EWOLULDBLOCK\n");
              break;
              default: printf("ERROR: default\n");
              }
}

/*************************************************************************/

void error_socket(int errno)
{
switch(errno){
              case EPROTONOSUPPORT:
                     printf("FAIL: EPROTONOSUPPORT\n");
              break;
              case EMFILE:
                     printf("FAIL: EMFILE\n");
              break;
              case ENFILE:
                     printf("FAIL: ENFILE\n");
              break;
              case ENOBUFS:
                     printf("FAIL: ENOBUFS\n");
              break;
              case EPROTOTYPE:
                     printf("FAIL: PROTOTYPE\n");
              break;
              default:
                     printf("FAIL: default\n");
             }
}

/*************************************************************************/

void error_connect(int errno)
{
switch(errno){
              case EADDRINUSE:
                     printf("ERROR: EADDRINUSE\n");
              break;
              case EADDRNOTAVAIL:
                     printf("ERROR: EADDRNOTAVAIL\n");
              break;
              case EAFNOSUPPORT:
                     printf("ERROR: EAFNOSUPPORT\n");
              break;
              case ECONNREFUSED:
                     printf("ERROR: ECONNREFUSED\n");
              break;
```

```
                case EDESTADDRREQ:
                        printf("ERROR: EDESTADDRREQ\n");
                break;
                case EFAULT:
                        printf("ERROR: EFAULT\n");
                break;
                case EINPROGRESS:
                        printf("ERROR: EDESTADDRREQ\n");
                break;
                case EINVAL:
                        printf("ERROR: EINVAL\n");
                break;
                case EISCONN:
                        printf("ERROR: EISCONN\n");
                break;
                case EMFILE:
                        printf("ERROR: EMFILE\n");
                break;
                case ENETUNREACH:
                        printf("ERROR: ENETUNREACH\n");
                break;
                case ENOBUFS:
                        printf("ERROR: ENOBUFS\n");
                break;
                case ENOTCONN:
                        printf("ERROR: ENOTCONN\n");
                break;
                case ENOTSOCK:
                        printf("ERROR: ENOTSOCK\n");
                break;
                case ETIMEDOUT:
                        printf("ERROR: ETIMEDOUT\n");
                break;
                default:
                        printf("default. errno: %d\n", errno);
                 }

}

/*****************************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
    Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST


void main(void)
{
int i,j;

dat.conexion_SUN = 1;

for (i=1;i<4;i++)    for (j=0;j<5;j++)  medida_a_SUN((char)i,(char) j, 1.0F);
getch();

for (i=1;i<4;i++)    for (j=0;j<5;j++)  medida_a_SUN((char)i,(char) j, 0.0F);

}

#endif

/*****************************************************************************/
```



```
/*****************************************************************************

   COMUNICA.C -        FUNCIONES PARA COMUNICAR CON EASI

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 4/1/99

*****************************************************************************/

#include <stdio.h>
#include <bios.h>
#include <conio.h>
#include <time.h>
#include <dos.h>
#include <memory.h>
#include <math.h>
#include "nombres.h"

/*#define TEST*/
#ifdef TEST
 #define extern
#endif

/***********************************
Par…metres de comunicaci¢ via COM1
***********************************/
#define DTR          0x01    /*Data Terminal Ready*/
#define RTS          0x02    /*Ready To Send*/
#define COM1PORT     0x0000  /*Pointer to Location of COM1 port*/
#define COM1         0
#define DCOM1        0x3F8

#define COM2         1
#define DCOM2        0x2F8

#define DATA_READY   0x100
#define FALSE        0
#define TRUE         !FALSE
#define SETTINGS ( 0xE0 | 0x00 | 0x02 | 0x00)  /* 9600,N,7,1 */
#define SETTINGS2 ( 0xE0 | 0x00 | 0x03 | 0x00)  /* 9600,N,8,1 */


int comunica(int *frase, int *salida);
int comunica2(int *frase, int *salida);
void reinicializa_puerto(void);
void reinicializa_puertob(void);
void reinicializa_puerto2(void);
int lectura_absor(void);
int lectura_ammv(void);

extern void delay( clock_t wait );

extern valores_fia  valor;
extern datos_fia     dat;
extern int colorim[3][20], ampotenciometre[20];
extern int offline;

/*****************************************************************************/
/*  Subrutina per enviar/rebre dades bit a bit a trav,s del COM1      */

int comunica(int *frase, int *salida)
{
    int in, m=0,i=0, j=0, k=0, status, DONE = FALSE;

if (offline==1) return -3;

/*    delay(10);*/

    status = _bios_serialcom(_COM_STATUS, COM1, 0); /*Comprueba estado*/
    while ( (status & DATA_READY)!=0 )
     {
       _bios_serialcom(_COM_RECEIVE, COM1, 0);
       status = _bios_serialcom(_COM_STATUS, COM1, 0);
       m=1;
     }
     if (m==1)  return -2;
```

```
    salida[0]=48;
    while ( !DONE )
    {
        k++;
        _outp ( DCOM1 + 4, DTR | RTS );
        if ( (in = frase[i]) != '□' )
            {
            i++;
            status = _bios_serialcom(_COM_STATUS, COM1, 0);
            _bios_serialcom(_COM_SEND, COM1, in);
            while ( (status & DATA_READY)==0 )
                    status = _bios_serialcom(_COM_STATUS, COM1, 0);
            salida[j] = _bios_serialcom(_COM_RECEIVE, COM1,0) & 0x7F;
            }
        status = _bios_serialcom(_COM_STATUS, COM1, 0);
        if(status & DATA_READY)
           {
           j++;
           salida[j] = _bios_serialcom(_COM_RECEIVE, COM1, 0) & 0x7F ;
           }
        if ( (frase[i] == '□') && (j >3) && (salida[j] == '\r'))  DONE=TRUE;
        if (k>25000) return -1;      /* retorna error i no agafa lectura */
        if (j>30) return -2;
    }
return 0;
}

/*************************************************************************/
/*  Subrutina per enviar/rebre dades bit a bit a trav,s del COM2      */

int comunica2(int *frase, int *salida)
{
    int in, m=0,i=0, j=0, k=0, status, DONE = FALSE;

if (offline==1) return -3;

/*    delay(10);*/

    status = _bios_serialcom(_COM_STATUS, COM2, 0); /*Comprueba estado*/
    while ( (status & DATA_READY)!=0 )
     {
       _bios_serialcom(_COM_RECEIVE, COM2, 0);
       status = _bios_serialcom(_COM_STATUS, COM2, 0);
       m=1;
     }
     if (m==1)  return -2;

    salida[0]=48;
    while ( !DONE )
    {
        k++;
        _outp ( DCOM2 + 4, DTR | RTS );
        if ( (in = frase[i]) != '□' )
            {
            i++;
            status = _bios_serialcom(_COM_STATUS, COM2, 0);
            _bios_serialcom(_COM_SEND, COM2, in);
            while ( (status & DATA_READY)==0 )
                    status = _bios_serialcom(_COM_STATUS, COM2, 0);
            salida[j] = _bios_serialcom(_COM_RECEIVE, COM2,0) & 0x7F;
            }
        status = _bios_serialcom(_COM_STATUS, COM2, 0);
        if(status & DATA_READY)
           {
           j++;
           salida[j] = _bios_serialcom(_COM_RECEIVE, COM2, 0) & 0x7F ;
           }
        if ( (frase[i] == '□') && (j >3) && (salida[j] == '\r'))  DONE=TRUE;
        if (k>25000) return -1;      /* retorna error i no agafa lectura */
        if (j>30) return -2;
    }
return 0;
}

/*************************************************************************/

void reinicializa_puerto(void)
```



```
{
_bios_serialcom(_COM_INIT, COM1, SETTINGS2);
}

/*************************************************************************/

void reinicializa_puertob(void)
{
_bios_serialcom(_COM_INIT, COM1, SETTINGS);
}

/*************************************************************************/

void reinicializa_puerto2(void)
{
_bios_serialcom(_COM_INIT, COM2, SETTINGS);
}

/*************************************************************************
                    Subrutina lectura d'absorb…ncia                    *******/

int lectura_absor(void)
{
int error=0, j=0;
char tempor[10];
float absor;
int retorno[40];

    do
    {
      j++;
      memset(retorno,0,sizeof(retorno));              /*Borra la variable*/
      error=comunica(colorim[2],retorno);
    }
    while(error!=0 && j<4);
/*for (j=0;j<25;j++) printf("%c",retorno[j]);*/

    if (error==0)
      {
      if (retorno[0]!=13 || retorno[2]!='3'|| retorno[8]!='A' ||
          retorno[7]!='"'|| retorno[24]!='"' ) return -2;
                    /*Ha le¡do pero no es correcto*/
      memset(tempor,0,sizeof(tempor));                 /*Borra la variable*/
      for (j=0;j<7;j++) tempor[j]=(char) retorno[12+j];    /*Asigna los caracteres*/
      absor=(float)atof(tempor);                /*Calcula el valor*/
      if ((absor>-0.1F) && (absor<1.0F))        /*Comprueba rango*/
       if ((absor<(valor.absor+0.60F)) && (absor>(valor.absor-0.60F)))
          {                                  /*Comprueba variaci¢n respecto anterior*/
          valor.absor=absor;
          valor.t_absor=time(NULL);
          return 0;
          }
      }
return -1;
}

/*************************************************************************
                    Subrutina lectura de mv                          ******/

int inicio=1;

int lectura_ammv(void)
{
int amerror=0, j=0;
char tempor[10];
int amretorno[40];
float temp;

memset(amretorno,0,sizeof(amretorno));                 /*Borra la variable*/
    do
    {
      j++;
      amerror=comunica2(ampotenciometre,amretorno);
    }
    while(amerror!=0 && j<4);

/*for (j=0;j<25;j++) printf("%c",amretorno[j]);*/
```

```
if (amerror==0)
       {
         if (amretorno[0]!=13 || amretorno[2]!='5' ||
             amretorno[7]!='"'|| amretorno[24]!='"' ) return -2;
                 /*Ha le¡do pero no es correcto*/
         memset(tempor,0,sizeof(tempor));              /*Borra la variable*/
         for (j=0;j<6;j++) tempor[j]=(char)amretorno[8+j]; /*Asigna los caracteres*/
         temp=(float)atof(tempor);                  /*Calcula el valor*/

         if ((temp>-400.0F) && (temp<100.0F))            /*Comprueba rango*/
           if((temp<(valor.mv+50.0F)) && (temp>(valor.mv-50.0F)) ||
              difftime(dat.t_actual,valor.t_mv)>60)
           {                              /*Comprueba variaci¢n respecto anterior*/
           valor.mv=temp;
           valor.t_mv=time(NULL);
           return 0;
           }
         if (inicio==1)
              if ((temp>-300.0F) && (temp<100.0F) && (temp!=0.0F) )
              {
              valor.mv=temp;
              valor.t_mv=time(NULL);
              inicio=0;
              }
       }
return -1;
}


/*****************************************************************************/
/*****************************************************************************/

#ifdef TEST
void main(void)
{
int i;
int tecla;

reinicializa_puertob();
reinicializa_puerto2();

    for (i=0;i<10000;i++)
    {
tecla=_bios_keybrd(_NKEYBRD_READY);  /*Comprueba si se ha presionado una tecla*/
if(tecla!=0) break;
    lectura_ammv();
    printf("\t%f\t%i\n",valor.mv,i);

    lectura_absor();
    printf("\t%f\t%i\n",valor.absor,i);
    delay(100);
    }

}
void delay( clock_t wait )
{
    clock_t goal;

    goal = wait + clock();
    while( goal > clock() )
        ;
}

int colorim[3][20]={{'\r','0','3','R','S','T','\r','¤'},    /* Reset color¡metre*/
                    {'\r','0','3','T','C','A','\r','¤'}, /* Calibraci¢ color¡metre*/
                    {'\r','0','3','R','D','I','\r','¤'}}; /* Lectura color¡metre*/
int ampotenciometre[20]={'\r','1','5','R','D','I','\r','¤'};         /* Lectura de mV*/

#endif
/*****************************************************************************/
```



```
/*****************************************************************************

   DELAY.C      FUNCION PARA ESPERAR LOS MILISEGUNDOS DETERMINADOS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 23/6/95

*****************************************************************************/

/********************* Libreriras  a incluir ************************/

#include <time.h>
#include <conio.h>
#include <stdio.h>

/*********************** Subrutinas del fichero **********************/

void delay( clock_t wait );

/*****************************************************************************/

/* Pauses for a specified number of microseconds. */
void delay( clock_t wait )
{
    clock_t goal;

    goal = wait + clock();
    while( goal > clock() )
        ;
}

/*****************************************************************************/

#ifdef  TEST
void main (void)
{
    int duracion=1000;
    clock_t  cstart, cend;

    cstart = clock();

    delay ( duracion );

    cend = clock();
    printf( "\nTiempo transcurrido:\t\t%6.4f seconds \n",
          ((float)cend - cstart) / CLOCKS_PER_SEC );
}
#endif
```

```
/*************************************************************************

   FICHEROS.C -        FUNCIONES PARA LA LECTURA Y ESCRITURA DE FICHEROS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 15/4/97

*************************************************************************/

/************************ Definiciones ******************************/


/********************** Librerias  a incluir ************************/

#include <io.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include  "nombres.h"

/*#define TEST*/
#ifdef TEST
#define extern
#endif

/********************** Subrutinas del fichero **********************/

void lectura_configuracio(void);
void guarda_mensaje( char *mensaje, char *fichero);
void guarda_medida( char *mensaje, char *fichero);
int  lee_calibrado( char *fichero, float *param_a, float *param_b);

/********************** Subrutinas externas necesarias **************/

/******************** Variables comunes del programa ****************/

extern datos_fia      dat;
extern valores_fia    valor;

/*************************************************************************/

/*************************************************************************
 Subrutina lectura de la configuraci¢ inicial del fitxer de dades fia4.dat */

void lectura_configuracio(void)
{
FILE *obert;
int si,i;
char linea[150];

if ( access("c:\\fiacfa\\fia4.dat",0) == 0)
  {
  obert=fopen("c:\\fiacfa\\fia4.dat","rt");

  /*En primer lugar se lee la l¡nea y despu,s se traduce a n£meros*/

  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.t_estabilit);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.inici_lectura_base);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.fi_lectura_base);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%d", &dat.inj_entre_calibrats);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.inici_lectura_max1);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.inici_lectura_max2);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.t_carrega_loop);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.t_injeccio);
  for (i=0;i<4;i++)
    {
    fgets(linea,sizeof(linea),obert);
      sscanf(linea,"%I", &dat.delta_t[i]);
```



```
    }
  for (i=0;i<3;i++)
    {
    fgets(linea,sizeof(linea),obert);
      sscanf(linea,"%I", &dat.delta_pic1[i]);
    }
  for (i=0;i<3;i++)
    {
    fgets(linea,sizeof(linea),obert);
      sscanf(linea,"%I", &dat.delta_pic2[i]);
    }
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.inici_lectura_base_calibratno3);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.fi_lectura_base_calibratno3);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%f", &dat.diferencia_entre_pics);
  fgets(linea,sizeof(linea),obert);
     sscanf(linea,"%I", &dat.t_canvi_punt_mostreig);
  fscanf(obert,"%i",&dat.t_lectura_amoni);
  si=fscanf(obert,"%*[^0123456789]");

  if (si==EOF)          /*Si hay menos datos de los necesarios*/
    {
    printf("\nError:  Faltan dades en el fitxer de configuraci¢ fia4.dat \n");
    fclose(obert);
    exit(1);
    }
  fclose(obert);
  }
else
  {
  printf("\nError:  El fitxer de configuraci¢ fia4.dat no existeix\n");
  exit (1);
  }
return;
}

/*************************************************************************/


void guarda_mensaje( char *mensaje, char *fichero)
{
   FILE *stream;
   time_t ltime;
   char la_frase[160];

   time (&ltime);

   sprintf(la_frase,"%s    %s\n",ctime( &ltime ), mensaje);
   la_frase[24]=32;

   stream = fopen( fichero, "a+");
   fprintf(stream,"%s", la_frase);
   fclose(stream);

return;
}

/*************************************************************************/

void guarda_medida( char *mensaje, char *fichero)
{
   FILE *stream;
   struct  tm *t;
   time_t ltime;

   time (&ltime);
   t = localtime( &ltime );

   stream = fopen( fichero, "a+");
   fprintf(stream, "%s", mensaje);
   fprintf(stream, "%2d %02d %02d  ", t->tm_hour, t->tm_min, t->tm_sec);
   fprintf(stream,"%s", ctime( &ltime ));

   fclose(stream);
```

```
return;
}

/*****************************************************************************/

int  lee_calibrado( char *fichero, float *param_a, float *param_b)
{
FILE *obert;
char linea1[150], linea2[150];

if ( access(fichero,0) != 0) return -1;

obert=fopen(fichero,"rt");
memset(linea1,0,sizeof(linea1));

while(fgets(linea1,sizeof(linea1),obert) != NULL) /* Lee linea mientras hay */
  {
  memcpy(linea2,linea1,sizeof(linea2));     /* Guarda la fltima linea */
  memset(linea1,0,sizeof(linea1));
  }

fclose(obert);

sscanf(linea2,"%f%f", param_a, param_b);    /* Convierte linea a par metros */

return 0;
}

/*****************************************************************************/
#ifdef TEST
void main(void)
{
float a=0.0F,b=0.0F;
int retorno;

retorno=lee_calibrado( "c:\\fiacfa\\datos\\calibnh4.his", &a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);
retorno=lee_calibrado( "c:\\fiacfa\\datos\\calilno2.his", &a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);
retorno=lee_calibrado( "c:\\fiacfa\\datos\\cali2no2.his", &a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);
retorno=lee_calibrado( "c:\\fiacfa\\datos\\calibno3.his", &a, &b);
printf("a: %f\tb: %f\t retorno: %i\n",a,b,retorno);


}
#endif
/*****************************************************************************/
```



```
/*****************************************************************************

   GRAFICOS.C -       FUNCIONES PARA LA PRESENTACION DE DATOS Y GRAFICAS
                   POR PANTALLA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 6/5/97

*****************************************************************************/

/********************** Librerias  a incluir ************************/

#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <graph.h>
#include <math.h>
#include <time.h>
#include "nombres.h"

/************************ Definiciones ********************************/
/*#define TEST*/
#ifdef TEST
#define extern
#endif

#define PATHFONTS "c:\\fiacfa\\fonts\\*.fon"

/********************** Subrutinas del fichero **********************/

void modo_grafico(void);
void graf_base(void);
void graf_led( int led, int on_off );
void graf_hora(void);
void graf_sombra_grafica(void);
void graf_ayuda_general(void);
void graf_borra_ayuda(void);
void graf_display(void);
void graf_valores_display(void);
void graf_valor_absor(void);
void graf_valor_mv(void);
void salir_al_dos(void);
void graf_ejes_tiempo(int escala);
void graf_eje_mv_absor(void);
void graf_eje_vertical
     (short x1, short y1, short y2, float min, float max, int div, short col);
void graf_eje_horizontal
     (short x1, short y1, short x2, int min, int max, int div, short col);
void graf_tanques(void);
void graf_tanque(int tanque);
void graf_presentacion(void);
int  boton ( int on_off, char *nombre, short x1, short y1,
          short x2, short y2, short font);
int graf_texto
   (char *nombre, short x1, short y1, short x2, short y2, int font, int fondo);
void actualizar_matrices_graficas(void);
void seleccionar_grafico(int orden);
void grafica_variable(int variable);
void barra_botones(void);

/****************** Variables comunes del programa ******************/

extern valores_fia  valor;
extern datos_fia    dat;

matriz_grafica mat_gra[1800]; /*Datos cada 4 segundos, 2 horas en total*/

/*****************************************************************************/
/*  Pasa del modo texto al gr fico.                                          */

void modo_grafico(void)
{
if( !_setvideomode( _MAXRESMODE ) )     exit( 1 );

_registerfonts( PATHFONTS ); /*Fuentes gr ficas*/
}
```

```
/*************************************************************************/
/* Funci¢n que llama a las funciones necesarias para dibujar la pantalla. */

void graf_base(void)
{

graf_sombra_grafica();
/*graf_ayuda_graficas();*/
graf_ayuda_general();
graf_display();
graf_hora();
barra_botones();
graf_texto("Temps  (minuts)", 130, 455, 230, 473, 1, 0);
graf_eje_mv_absor();
graf_eje_horizontal (80,434,560, (short)((dat.inicio_graf-4)*30),
      (short)((dat.fin_graf-4)*30), 6, 30);
graf_valor_absor();
graf_valor_mv();

}

/*************************************************************************/
/* Reloj digital, con fecha y hora.                                      */

void graf_hora(void)
{
time_t ltime;
short x=445,
      y=460;
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[30];
int segundos,minutos;

_setgtextvector( 1, 0 );
_setfont( fuente );

time (&ltime);
sprintf(buffer,"%s", ctime( &ltime ));

_setcolor(242);
_rectangle( _GFILLINTERIOR, 441, 459 ,587, 471);
_setcolor(28);
_rectangle( _GBORDER, 441, 459 ,587, 471);

_setcolor(29);
_moveto(x,y);
_outgtext(buffer);

_rectangle( _GBORDER, 350, 459 ,430, 471);
segundos=(int)(time(NULL)-dat.t_cicle);
minutos=segundos/60;
segundos=segundos-minutos*60;
sprintf( buffer, "T cicle:  %2i : %2i", minutos,segundos);
graf_texto( buffer, 351, 460, 429, 470, 0, 1);

}

/*************************************************************************/
/* Escribe la pantalla de ayuda general del programa.                    */

void graf_ayuda_general(void)
{
int i;
short x,y;
char fuente[20]="t'Tms Rmn'h8w5b";
char buffer[15][28]={/************************/
                     "F1  Carga mostra FIA ",
                     "F2  Enc,n bomba FIA  ",
                     "F3  Apaga bomba FIA ",
                     "F4  Inyecci¢ mostra FIA ",
                     "F5  Inic. v…lvula 6 vies ",
                     "F6  Apaga v…l. sup. FIA ",
                     "F7  Enc,n v…l. sup. FIA ",
                     "F8  Enc,n v…l. inf. FIA ",
                     "F9  Apaga bomba CFA ",
                     "AltF2  Enc,n bomba CFA ",
```



```
                     "AltF3  Apaga bomba CFA ",
                     "AltF6  Apaga v…lv.sup.CFA",
                     "AltF7  Enc,n v…lv.sup.CFA",
                     "AltF8  Enc,n v…lv.inf.CFA",
                     "AltF9  Apaga v…lv.inf.CFA "};
                     /*,
                     "F10  Sortida del programa",
                     "ALT-D  Sortida al DOS",
                     "ESC  Cr,ditos programa"};       */

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(242);/*antes 19*/
_rectangle( _GFILLINTERIOR, 0, 0, 595, 38);
_setcolor(29);
_rectangle( _GBORDER, 0, 0, 595, 38);

for (i=0;i<15;i++)
    {
    x=(short)(5+i*120-(int)(i/5)*600);
    y=(short)(2+12*(int)(i/5));
    _moveto(x,y);
    _outgtext(buffer[i]);
    }
}

/*************************************************************************/
/* Dibuja un rect ngulo negro para borrar el cuadro de ayuda, y as¡ poder
   escribir un mensaje en esa zona.                                      */

void graf_borra_ayuda(void)
{
_setcolor(0);
_rectangle( _GFILLINTERIOR, 0, 0, 595, 38);
_setcolor(29);
_rectangle( _GBORDER, 0, 0, 595, 38);
}

/*************************************************************************/
/* Dibuja el rect ngulo, las rayas interiores y los t¡tulos del display
   donde se muestran los valores.                                        */

void graf_display(void)
{
int i, j;
short x, y;

char fuente[20]="t'Tms Rmn'h8w5b";
char variables[6][12]={/**********/
                       "  N-NO3-   ",
                       "  N-NO2-   ",
                       "  N-NH4+   ",
                       "    mV     ",
                       "  Absorb   ",
                       "           "};
char tanques[3][12]=  {/**********/
                       "   Mesura ",
                       "   Temps ",
                       "   Punt   "};

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(171);
_rectangle( _GFILLINTERIOR, 0, 41, 595, 97);
_setcolor(29);
_rectangle( _GBORDER, 0, 41, 595, 97);

_setcolor(10);
_rectangle( _GBORDER, 15, 56, 76, 82);
_moveto(20,59);
_outgtext("ULTIMS");
_moveto(20,69);
_outgtext("VALORS");

_setcolor(30);
```

```
_rectangle( _GBORDER, 500, 56, 576, 82);
_moveto(535,56);
_lineto(535,82);
_moveto(535,69);
_lineto(500,69);
_moveto(505,58);
_outgtext("Absor ");
_moveto(505,70);
_outgtext("mV  ");

for (i=1;i<4;i++)
    {
    _setcolor(30);
    x=(short)(103+i*70);
    y=(short)(45);
    _rectangle( _GBORDER, x, y, (short)(x+70), (short)(y+12));
    _setcolor(10);
    _moveto((short)(x+10),y);
    _outgtext(variables[i-1]);
    }
for (j=1;j<4;j++)
    {
    _setcolor(30);
    x=(short)(103);
    y=(short)(45+j*12);
    _rectangle( _GBORDER, x, y, (short)(x+70), (short)(y+12));
    _setcolor(10);
    _moveto((short)(x+10),y);
    _outgtext(tanques[j-1]);
    }

_setcolor(30);
_rectangle( _GBORDER, 415, 56, 476, 82);
_moveto(415,69);
_lineto(476,69);
_moveto(417,57);
_outgtext("Punt mostreig");


graf_valores_display();
}

/*************************************************************************/
/* Escribe los valores de las variables medidas del sistema en el display.*/

void graf_valores_display(void)
{
int i, j;
short x, y;
char fuente[20]="t'Tms Rmn'h8w5b";
char buffer[10];
struct tm *temps;

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(0);
_rectangle( _GFILLINTERIOR, 173, 57, 383, 93);

_setcolor(242);
for (j=1;j<4;j++)
{
  for (i=1;i<4;i++)
    {
    x=(short)(103+i*70);
    y=(short)(45+j*12);
    _rectangle( _GBORDER, x, y, (short)(x+70), (short)(y+12));
    }
}

_setcolor(30);
_rectangle( _GBORDER, 173, 57, 383, 93);

      sprintf(buffer,"%4.2f",valor.nitrats);
     _moveto(200, 58); _outgtext(buffer);
     temps = localtime(&valor.t_nitrats);
     strftime( buffer, 20, "%H:%M:%S", temps);
```



```
     _moveto(190, 70); _outgtext(buffer);
      sprintf(buffer,"%li",valor.punt_nitrats);
     _moveto(205, 82); _outgtext(buffer);

      sprintf(buffer,"%4.2f",valor.nitrits);
     _moveto(270, 58); _outgtext(buffer);
     temps = localtime(&valor.t_nitrits);
     strftime( buffer, 20, "%H:%M:%S", temps);
     _moveto(260, 70); _outgtext(buffer);
      sprintf(buffer,"%li",valor.punt_nitrits);
     _moveto(275, 82); _outgtext(buffer);

      sprintf(buffer,"%4.2f",valor.amoni);
     _moveto(340, 58); _outgtext(buffer);
     temps = localtime(&valor.t_amoni);
     strftime( buffer, 20, "%H:%M:%S", temps);
     _moveto(330, 70); _outgtext(buffer);
      sprintf(buffer,"%li",valor.punt_amoni);
     _moveto(345, 82); _outgtext(buffer);

sprintf(buffer,"%li",dat.punt_mostreig);
graf_texto( buffer, 416, 70, 475, 81, 0, 1);

}
/*************************************************************************/
/* Escribe los valores de las variables medidas del sistema en el display.*/

void graf_valor_absor(void)
{
char medida[10];

sprintf (medida, "%2.3f", valor.absor);
graf_texto( medida, 536, 57, 575, 68, 0, 1);
}

/*************************************************************************/
/* Escribe los valores de las variables medidas del sistema en el display.*/

void graf_valor_mv(void)
{
char medida[10];

sprintf (medida, "%2.3f", valor.mv);
graf_texto( medida, 536, 70, 575, 81, 0, 1);
}

/*************************************************************************/
/* Dibuja la sombra de la gr fica, la pantalla base.                     */

void graf_sombra_grafica(void)
{
_setcolor(23);              /*sombra*/
_rectangle( _GFILLINTERIOR, 5, 105, 595, 479);
_setcolor(171);
_rectangle( _GFILLINTERIOR, 0, 100, 590, 474);
_setcolor(29);
_rectangle( _GBORDER, 0, 100 , 590, 474);
_setcolor(0);
_rectangle( _GFILLINTERIOR, 80, 115 , 560, 434);
_setcolor(29);
_rectangle( _GBORDER, 80, 115 , 560, 434);
_moveto(560,275);
_lineto(80,275);
}


/*************************************************************************/
/* Funci¢n para salir al dos. Cambio de modo gr fico a texto, ejecuta un
   command y despu,s redibuja la pantalla.            */

void salir_al_dos(void)
{
_setvideomode( _DEFAULTMODE );
printf("\nEscribe 'exit' para volver al programa\n");
system("COMMAND.COM");
if( !_setvideomode( _MAXRESMODE ) )     exit( 1 );
```

```
graf_base();

}

/*************************************************************************/
/* Dibuja el eje horizontal del tiempo, en tres escalas diferentes.      */

void graf_ejes_tiempo(int escala)
{
char fuente[20]="t'Tms Rmn'h12w50b";

_setgtextvector( 1, 0 );
_setfont( fuente );
_setcolor (30);

switch(escala)
  {
  case 1:                          /*Escala de una hora*/
      graf_eje_horizontal( 80,     434,  560, 0, 60,12, 30);
     _moveto(150,460);
     _outgtext("Tiempo  (minutos)");
  break;
  case 2:                          /*Escala de un d¡a*/
      graf_eje_horizontal( 80,     434,  560, 0, 24,24, 30);
     _moveto(150,460);
     _outgtext("Tiempo  (horas)");
  break;
  case 3:                          /*Escala de una semana*/
      graf_eje_horizontal( 80,     434,  560, 0, 7,7, 30);
     _moveto(150,460);
     _outgtext("Tiempo  (d¡as)");
  break;
  }

/*
_rectangle( _GBORDER, 80, 115 , 560, 434);
*/
}

/*************************************************************************/
/* Dibuja el eje vertical de mv y absorbancia.                           */

void graf_eje_mv_absor(void)
{
graf_eje_vertical(80,120,265,-0.1F,1.0F,11,92);
graf_eje_vertical(80,285,430,-250.0F,50.0F,6,100);

_setcolor(92);
_setgtextvector( 0, 1);
_moveto(20,220);
_outgtext("Absorb…ncia");
_setcolor(100);
_moveto(20,400);
_outgtext("Potencial (mV)");
}

/*************************************************************************/
void graf_eje_vertical
     (short x1, short y1, short y2, float min, float max, int div, short col)
{
int i=0;
short y;
float incremento=0;
float inc_med=0.0F;
char valor[5];
char fuente[20]="t'Tms Rmn'h12w50b";

_setcolor(col);
_setgtextvector( 1, 0 );
_setfont( fuente );
incremento=(float)((float)(y2-y1)/div);
inc_med= ( (max-min) /div);

_moveto(x1,y1);
_lineto(x1,y2);

for (i=0;i<=div;i++)
```



```
   {
   y=(short)(y2-incremento*i);
   _moveto(x1,y);
   _lineto((short)(x1-5),y);
   if (max<5.0F) sprintf(valor,"%2.1f",inc_med*i+min);
   else sprintf(valor,"%2.0f",inc_med*i+min);
   _moveto((short)(x1-30),(short)(y-5));
   _outgtext(valor);
   }
}

/*************************************************************************/

void graf_eje_horizontal
     (short x1, short y1, short x2, int min, int max, int div, short col)
{
int i=0;
short x;
float incremento=0;
int inc_med=0;
char valor[5];
char fuente[20]="t'Tms Rmn'h12w50b";

_setcolor(col);
_setgtextvector( 1, 0 );
_setfont( fuente );
incremento=(float)((float)(x2-x1)/div);
inc_med= (int) ( (max-min) /div);

_moveto(x1,y1);
_lineto(x2,y1);

for (i=0;i<=div;i++)
   {
   x=(short)(x1+incremento*i);
   _moveto(x,y1);
   _lineto(x,(short)(y1+5));
   sprintf(valor,"%2i",min+inc_med*i);
   _moveto((short)(x-7),(short)(y1+10));
   _outgtext(valor);
   }
}

/*************************************************************************/
/* Dibuja el nombre del tanque al que corresponde la gr fica m£ltiple.   */

void graf_tanque(int tanque)
{
char fuente[20]="t'Tms Rmn'h12w50b";
char buffer[30];

_setgtextvector( 1, 0 );
_setfont( fuente );

_setcolor(0);
_rectangle( _GFILLINTERIOR, 300, 459 ,370, 471);
_setcolor(30);
_rectangle( _GBORDER, 300, 459 ,370, 471);

_moveto(310 ,460);
sprintf(buffer,"Tanque %1i",tanque);
_setcolor(51);
_outgtext(buffer);
}

/*************************************************************************/
/* Funci¢n con el copyright, presentaci¢n del programa.                  */

void graf_presentacion(void)
{
char fuente1[20]="t'Tms Rmn'h60w50b";
char fuente2[20]="t'Tms Rmn'h16w50b";
char fuente3[20]="t'Tms Rmn'h12w50b";

_setgtextvector( 1, 0 );

_setcolor(104);
```

```
_rectangle( _GFILLINTERIOR, 81, 116 , 559, 433);

_moveto(250 ,150);
_setcolor(79);
_setfont( fuente1 );
_outgtext("FIACFA");
_moveto(180 ,200);
_setfont( fuente2 );
_outgtext("Programa pel control d'analitzadors");
_moveto(220 ,225);
_outgtext("de Nitrits, Nitrats i Amoni");
_setcolor(77);
_moveto(100 ,362);
_setfont( fuente3 );
_outgtext("Copyright: Juan Baeza - David Gabriel 1997");
_moveto(100 ,380);
_outgtext("Departament d'Enginyeria Qu¡mica");
_moveto(100 ,398);
_outgtext("Universitat Aut*noma de Barcelona");

/* 80, 115 , 560, 434*/
}

/*****************************************************************************/

int boton
 ( int on_off, char *nombre, short x1, short y1, short x2, short y2, short font)
{
short x11=(short)(x1+1),
      x12=(short)(x1+2),
      x21=(short)(x2-1),
      x22=(short)(x2-2),
      y11=(short)(y1+1),
      y12=(short)(y1+2),
      y21=(short)(y2-1),
      y22=(short)(y2-2);

short color1,color2;
short xt,yt,largo;
char fuente[3][20]={"t'Terminal'h12w50b",
                    "t'Tms Rmn'h10w50b",
                    "t'Tms Rmn'h8w5b"};

_setgtextvector( 1, 0 );
_setfont( fuente[font] );

largo=_getgtextextent(nombre);
if (largo>(x2-x1-4)) return -2;

switch(on_off)
{
  case 0:
    color1 = 15;
    color2 = 8;
  break;
  case 1:
    color1 =  8;
    color2 =  7;
  break;
  default:
  return  -1;
}

_setcolor(7);
_rectangle( _GFILLINTERIOR, x1, y1, x2, y2);
_setcolor(0);
_rectangle( _GBORDER, x1, y1, x2, y2);
_setcolor(color1);
_moveto(x11,y21);
_lineto(x11,y11);
_lineto(x21,y11);
_moveto(x12,y22);
_lineto(x12,y12);
_lineto(x22,y12);
_setcolor(color2);
_moveto(x21,y11);
_lineto(x21,y21);
```



```
_lineto(x11,y21);
_moveto(x22,y12);
_lineto(x22,y22);
_lineto(x12,y22);

_setcolor(0);
switch(on_off)
{
  case 0:
    xt=(short)((x1+x2-largo)/2);
    yt=(short)((y1+y2-12)/2);
  break;
  case 1:
    xt=(short)((x1+x2-largo)/2+1);
    yt=(short)((y1+y2-12)/2+1);
  break;
  default:
    xt=(short)(x1+2);
    yt=(short)(y1+2);
  break;
}
_moveto ( xt, yt);
_outgtext( nombre );
}
/*****************************************************************************/

int graf_texto
   (char *nombre, short x1, short y1, short x2, short y2, int font, int fondo)
{
short largo;
char fuente[2][20]={"t'Tms Rmn'h8w5b","t'Tms Rmn'h12w50b"};
short xt,yt;

_setgtextvector( 1, 0 );
_setfont( fuente[font] );

largo=_getgtextextent(nombre);
if (largo>(x2-x1-2)) return -2;

if (fondo==1)
    {
    _setcolor(0);
    _rectangle( _GFILLINTERIOR, x1, y1, x2, y2);
    }
xt=(short)((x1+x2-largo)/2);
yt=(short)((y1+y2-9)/2);
_setcolor(30);
_moveto ( xt, yt);
_outgtext( nombre );

return 0;
}

/*****************************************************************************/
/* Actualiza làs matrices utilizadas para dibujar las gr ficas.
   mat_graf; datos de 2 horas, cada 4 segundos      */

void actualizar_matrices_graficas(void)
{
int i;

for (i=0;i<1799;i++)
    {
    mat_gra[i]=mat_gra[i+1];
    }
 /*Hacemos sitio para el £ltimo valor*/

  mat_gra[1799].absor=valor.absor;
  mat_gra[1799].mv=valor.mv;
  mat_gra[1799].t_absor=valor.t_absor;
  mat_gra[1799].t_mv=valor.t_mv;
 /*Asignamos el £ltimo valor*/
}

/*****************************************************************************/
void seleccionar_grafico(int orden)
{
```

```
#define N_SEC  4
switch(orden){
        case 1: /*AMUNT*/
            dat.inicio_graf--;
            if (dat.inicio_graf<0)
                {
                dat.inicio_graf=0;
                dat.fin_graf++;
                if (dat.fin_graf>N_SEC) dat.fin_graf=N_SEC;
                }
        break;
        case 2: /*ABAIX*/
            dat.inicio_graf++;
            if ((dat.fin_graf-dat.inicio_graf)<1)
                dat.inicio_graf=dat.fin_graf-1;
        break;
        case 3: /*DRETA*/
            dat.fin_graf++;
            if (dat.fin_graf>N_SEC)
                {
                dat.fin_graf=N_SEC;
                break;
                }
            dat.inicio_graf++;
            if (dat.inicio_graf>(N_SEC-1)) dat.inicio_graf=(N_SEC-1);
        break;
        case 4:   /*ESQUERRA*/
            dat.inicio_graf--;
            if (dat.inicio_graf<0)
                {
                dat.inicio_graf=0;
                break;
                }
            dat.fin_graf--;
            if (dat.fin_graf<1) dat.fin_graf=1;
        break;
        }

_setcolor(171);
_rectangle( _GFILLINTERIOR, 70, 442 ,570, 454);
graf_eje_horizontal (80,434,560, (short)((dat.inicio_graf-N_SEC)*(120/N_SEC)),
        (short)((dat.fin_graf-N_SEC)*(120/N_SEC)), 6, 30);
grafica_variable(0);
grafica_variable(1);

}

/*************************************************************************/
/* Grafica las l¡neas correspondientes de mV y absorbacia
   en la escala de tiempo proporcionada.       */

void grafica_variable(int variable)
{
int i;
short x,y;
float x0[2]={80.0F,80.0F},
      y0[2]={265.0F,430.0F};
float  alto=145.0F;
float sets[2][3]={/*m¡nimo, m ximo, rango*/
                  {-0.10F,1.0F,1.1F},              /*Absorbancia*/
                  {-250.0F,50.0F,300.0F}};  /*mv*/
double increm_t, increm_tt;
time_t t_final,t_inicial;
int inicio=0,
    fin=1799;

_setcolor(30);
_moveto(560,275);
_lineto(80,275);

t_inicial = dat.t_actual-((4-dat.inicio_graf)*30*60);
t_final =  t_inicial+((dat.fin_graf-dat.inicio_graf)*30*60);
increm_tt = (double)(t_final-t_inicial);

switch(variable){
  case 0:               /*Absorbancia*/
    for (i=0;i<1800;i++)
```



```
        if (mat_gra[i].t_absor>t_inicial)
        {
            inicio=i;
            break;
        }
    for (i=0;i<1800;i++)
        if (mat_gra[i].t_absor>t_final)
        {
            fin=i;
            break;
        }
    _setcolor(0);
    _rectangle( _GFILLINTERIOR, 81, 116, 559, 274);
    _setcolor(92);
    x=(short)(x0[0]+1);
    y=(short)(y0[0]-alto/sets[0][2]*(mat_gra[inicio].absor-sets[0][0]));
    if (y<120) y=120;
    if (y>265) y=265;
    _moveto(x,y);
    for (i=inicio+1;i<fin;i++)
        {
        increm_t=(double) difftime(t_final,mat_gra[i].t_absor);
        x=(short)(559.0F-increm_t/increm_tt*478.0F);
        if (x<81 | x>559)       continue;
        y=(short)(y0[0]-alto/sets[0][2]*(mat_gra[i].absor-sets[0][0]));
        if (y<120) y=120;
        if (y>265) y=265;
          _lineto(x,y);
        }
    break;
  case 1:                       /* mV */
    for (i=0;i<1800;i++)
        if (mat_gra[i].t_mv>t_inicial)
        {
            inicio=i;
            break;
        }
    for (i=0;i<1800;i++)
        if (mat_gra[i].t_mv>t_final)
        {
            fin=i;
            break;
        }
    _setcolor(0);
    _rectangle( _GFILLINTERIOR, 81, 276, 559, 433);
    _setcolor(100);
    x=(short)(x0[1]+1);
    y=(short)(y0[1]-alto/sets[1][2]*(mat_gra[inicio].mv-sets[1][0]));
    if (y<285) y=285;
    if (y>430) y=430;
    _moveto(x,y);
    for (i=inicio+1;i<fin;i++)
        {
        increm_t=(double) difftime(t_final,mat_gra[i].t_mv);
        x=(short)(559-increm_t/increm_tt*478.0F);
        if (x<81 | x>559)       continue;
        y=(short)(y0[1]-alto/sets[1][2]*(mat_gra[i].mv-sets[1][0]));
        if (y<285) y=285;
        if (y>430) y=430;
        _lineto(x,y);
        }
    break;
    }
}

/*************************************************************************/

void inventa_valores(void)
{
time_t t1;
int i;

t1=time(NULL);

for (i=0;i<1800;i++)
    {
     mat_gra[i].t_mv=(long)(t1-4.0*(1800.0-i));
```

```
      mat_gra[i].t_absor=(long)(t1-4.0*(1800.0-i));
      mat_gra[i].mv=(float)(10.0-300.0*i/1800.0*sin((double)i/40.0));
      mat_gra[i].absor=(float)(0.5F+0.5*sin((double)i/50.0));
/*
matg[1][i].ph=(float)(7.0 + 4.0 * sin((double)i/30.0));
matg[1][i].temp=(float)(30.0+10.0 * cos((double)i/30.0));
matg[1][i].tiempo=(long int) (time(NULL)+((long int)i*60*5*8));*/
    }
}
/*************************************************************************/

void barra_botones(void)
{
_setcolor(171);
_rectangle( _GFILLINTERIOR, 600, 0 ,639, 479);
_setcolor(29);
_rectangle( _GBORDER, 600, 0 ,639, 479);
boton( 0, "Barra", 603, 3, 636, 20, 1);
boton( 0, "Ar", 610, 30, 629, 50, 2);
boton( 0, "Ab", 610, 60, 629, 80, 2);
boton( 0, "De", 610, 90, 629, 110, 2);
boton( 0, "Iz", 610, 120, 629, 140, 2);

}
/*************************************************************************/
/* C¢digo necesario si desean hacerse pruebas.
    Si quiere utilizarse debe definirse TEST al inicio del fichero      */

#ifdef TEST

void main()
{
modo_grafico();
graf_display();
graf_base();
graf_presentacion();
graf_ejes_tiempo(1);
graf_valor_absor();
graf_valor_mv();
getch();
getch();
_setvideomode( _DEFAULTMODE );
}
#endif
```



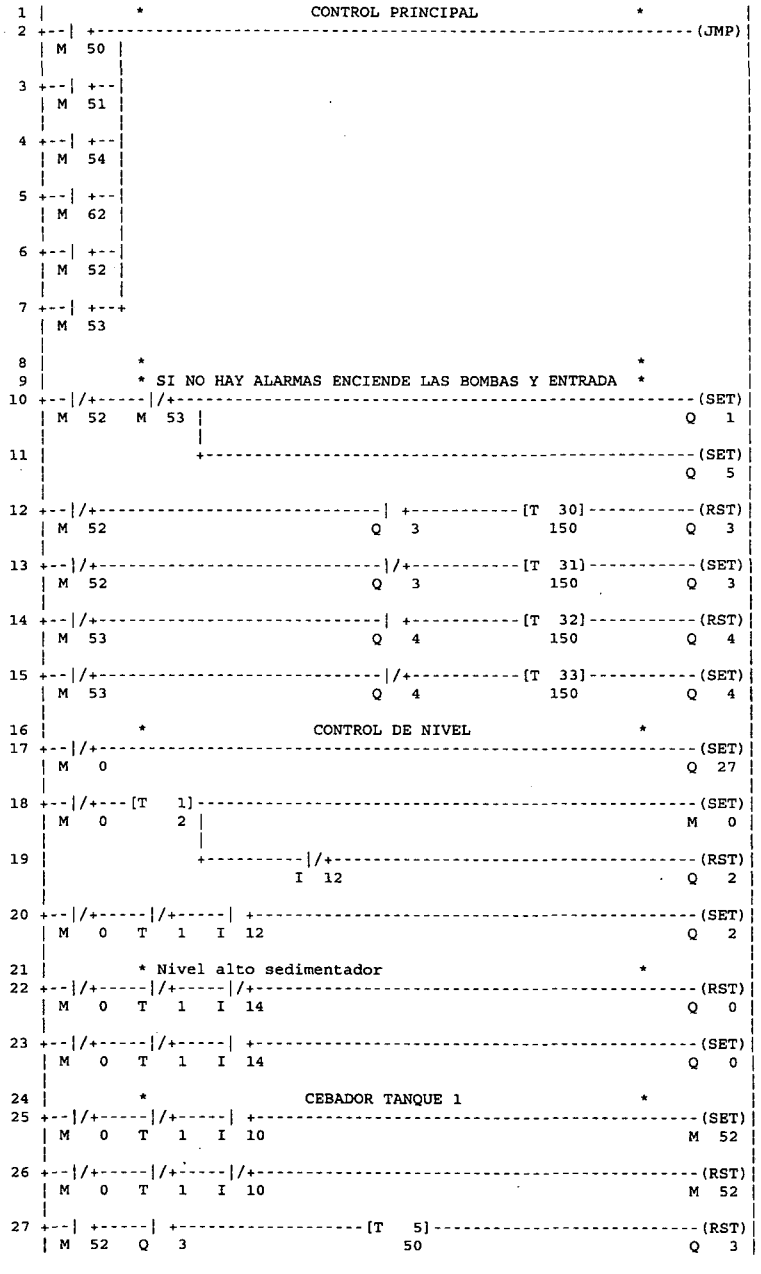
```
/*************************************************************************

   PCL711.C -   FUNCIONES PARA UTILIZAR LA PLACA DE SALIDAS  pcl711

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 28/8/97

*************************************************************************/

/******************** Librerias  a incluir ***************************/

#include <stdio.h>
#include <conio.h>

/********************** Definiciones *********************************/

#define BASE 0x220                      /*direcci¢ base de la placa pcl711*/

/********************** Subrutinas del fichero ***********************/

int pcl711 (int salida, int on_off);
void actuacion_externa_pcl711(void);
void presa_mostra(int la_mostra);
void apaga_pcl711(void);

/*************************************************************************/
/* Funci¢n para pedir por pantalla en modo texto los par metros para
   ejecutar la funci¢n pcl711, que var¡a el estado de una salida. */

void actuacion_externa_pcl711(void)
{
int salida,on_off;

printf ("N€mero de salida: ");
fscanf (stdin, "%d", &salida);
printf ("Encendido/Apagado: ");
fscanf (stdin, "%d", &on_off);

pcl711 (salida, on_off);
}

/*************************************************************************/
/* Control de la placa pcl711.
   Funci¢n para variar el estado de una salida de la placa pcl711.       */


int pcl711 (int salida, int on_off)
{
unsigned char high_byte,low_byte;
static unsigned int consigna=0;

if (on_off!=0 && on_off!=1) return -1;
if (salida<0 || salida>15) return -2;
  /*Si los par metros no son correctos, devuelve un c¢digo de error*/


if (on_off==1) consigna=consigna | (1<<salida);
if (on_off==0) consigna=consigna & ~(1<<salida);

    low_byte=(char) (consigna & 255);
    high_byte=(char) (consigna>>8);

    _outpw(BASE+13,low_byte);
    _outpw(BASE+14,high_byte);

return 1;
}
/*************************************************************************/

void apaga_pcl711(void)
{
int i=0;
for (i=1;i<16;i++) pcl711(i,0);
}

/*************************************************************************/
```

```
void presa_mostra(int la_mostra)
{
apaga_pcl711();
/*
gotoxy(78,25);
printf("%li",la_mostra);
*/
switch(la_mostra)
   {
   case 0:
        pcl711(8,1);   /* Circuit mostreig Reactor Anaerobi */
        pcl711(7,1);
   break;
   case 1:              /* Circuit mostreig Reactor 1 */
        pcl711(1,1);   /* electrovalvula 1*/
        pcl711(2,1);   /* electrovalvula 2*/
        pcl711(5,1);   /* electrovalvula 10*/
   break;
   case 2:              /* Circuit mostreig Reactor 2 */
        pcl711(1,1);    /* electrovalvula 1*/
        pcl711(3,1);    /* electrovalvula 3*/
        pcl711(5,1);    /* electrovalvula 10*/
   break;
   case 3:              /* Circuit mostreig Reactor 3 */
        pcl711(6,1);    /* electrovalvula 11*/
   break;
   case 4:              /* Circuit mostreig Sortida sistema */
        pcl711(4,1);    /* electrovalvula 7*/
   break;
   case 5:
   break;              /* Mostra manual FIA */
   }
}

/*************************************************************************/

#ifdef TEST

 int main( int argc, char *argv[])
 {
int salida,on_off;
long int i;
/*
if (argc==1)
    {
    printf("\nUtiliza:  pcl711 salida on_off \n");
    return -1;
    }

salida=atoi(argv[1]);
on_off=atoi(argv[2]);

if (pcl711 (salida, on_off)==1) printf("\nSalida OK\n");
else printf("\nSalida NO OK\n");
*/

for (salida=0;salida<16;salida++)
    {
    pcl711(salida,1);
    for (i=0;i<300000;i++);
    }
for (salida=0;salida<16;salida++)
    {
    pcl711(salida,0);
    for (i=0;i<300000;i++);
    }

return 0;
}
#endif
/*************************************************************************/
```

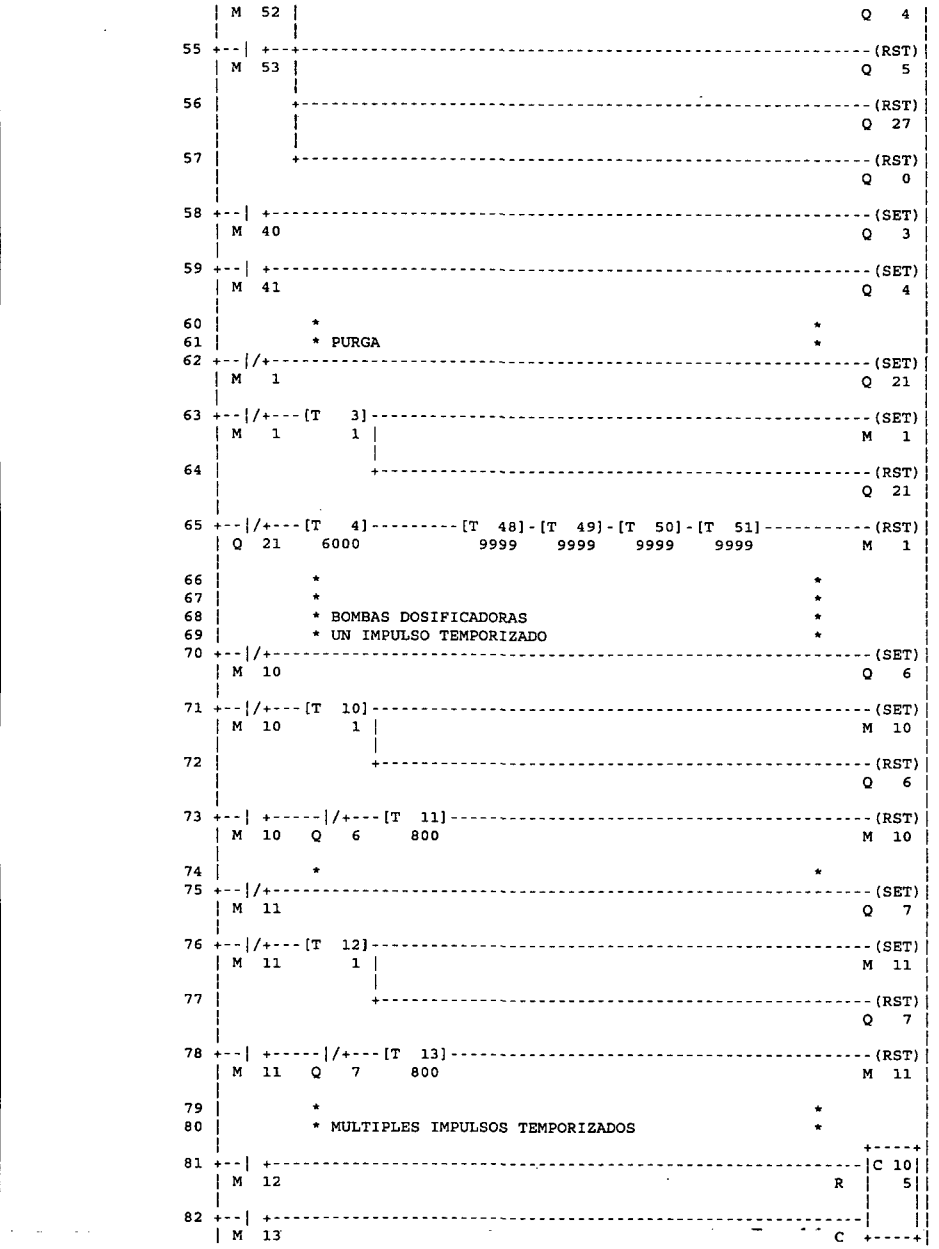


```
/*************************************************************

   FITXER DE CONFIGURACIO FIA4.DAT

*************************************************************/

10       temps estabilitzacio inicial sistema (seg)
30      temps inici calcul linea base en operacio normal (seg)
50      temps fi calcul linea base en operacio normal (seg)
6000     numero d'injeccions entre calibrats nitrats (nou hores)
85      temps inici lectura deteccio maxim nitrits(seg)
220     temps inici lectura deteccio maxim nitrats+nitrits(seg)
40      temps retorn a posicio 'load' (seg) (carrega loop)
600     temps retorn a posicio 'inject'(seg)
38      increment de temps entre 50 ppm i 30 ppm nitrats (seg)
65      increment de temps entre 50 ppm i 15 ppm (seg)
100     increment de temps entre 50 ppm i  5 ppm (seg)
130     increment de temps entre 50 ppm i  1 ppm (seg) (antes 330)
17      increment de temps pic entre 3 ppm i 2 ppm nitrits (seg)
28      increment de temps pic entre 3 ppm i 1 ppm nitrits (seg)
39      increment de temps pic entre 3 ppm i 0.5 ppm nitrits (seg)
32      increment de temps pic entre 7 ppm i 5 ppm nitrits (seg)
54      increment de temps pic entre 7 ppm i 3 ppm nitrits (seg)
87      increment de temps pic entre 7 ppm i 1 ppm nitrits (seg)
40      temps inici calcul linea base en calibracio nitrats (seg)
60      temps fi calcul linea base en calibracio nitrats (seg)
0.010   error maxim acceptat entre dos pics consecutius en la calibracio (mUA)
460     temps pel canvi del punt de mostreig
480      temps per la lectura dels mv pel calcul de l'amoni
```

3 PROGRAMA DEL PLC


CONTROL PRINCIPAL
M 50
JMP
M 51
M 54
M 62
M 52
M 53
SI NO HAY ALARMAS ENCIENDE LAS BOMBAS Y ENTRADA
M 52
M 53
SET
Q 1
Q 5
RST
T 30
150
Q 3
T 31
T 32
T 33
CONTROL DE NIVEL
M 0
Q 27
T 1
300
M 2
I 12
Q 2
Nivel alto sedimentador
I 14
CEBADOR TANQUE 1
M 10
M 52
T 5
50

T 6
50
CEBADOR TANQUE 2
M 11
M 53
T 7
T 8
ALARMA DE POSIBLE ESCAPE
M 13
M 54
TEMPORIZADOR DEL CONTROL DE NIVEL
T 2
CONTADORES - ENCENDIDO
C 0
C 1
T 65
CONTADORES - APAGADO
M 3
ALARMAS CONTADORES
M 50
M 51


M 52
Q 4
55
M 53
(RST)
Q 5
56
(RST)
Q 27
57
(RST)
Q 0
58
M 40
(SET)
Q 3
59
M 41
(SET)
Q 4
60
61
* PURGA
62
M 1
(SET)
Q 21
63
M 1
T 3
1
(SET)
M 1
64
(RST)
Q 21
65
Q 21
T 4
6000
T 48
9999
T 49
9999
T 50
9999
T 51
9999
(RST)
M 1
66
67
68
* BOMBAS DOSIFICADORAS
69
* UN IMPULSO TEMPORIZADO
70
M 10
(SET)
Q 6
71
M 10
T 10
1
(SET)
M 10
72
(RST)
Q 6
73
M 10
Q 6
800
T 11
(RST)
M 10
74
75
M 11
(SET)
Q 7
76
M 11
T 12
1
(SET)
M 11
77
(RST)
Q 7
78
M 11
Q 7
800
T 13
(RST)
M 11
79
80
* MULTIPLES IMPULSOS TEMPORIZADOS
81
M 12
R
C 10
5
82
M 13
C

83
C 10
3000
T 14
(
)
M 12
84
85
C 10
M 13
(SET)
Q 6
86
(SET)
Q 7
87
C 10
M 13
1
T 15
(SET)
M 13
88
(RST)
Q 6
89
(RST)
Q 7
90
M 13
C 10
Q 6
7
T 16
(RST)
M 13
91
92
(JEND)
93
94
* APAGADO EXTERNO
95
M 61
(RST)
Q 1
96
(RST)
Q 2
97
(RST)
Q 3
98
(RST)
Q 4
99
(RST)
Q 0
100
(RST)
Q 27
101
(SET)
M 62
102
T 21
10
(RST)
M 61
103
104
* RESET DE ALARMAS, REINICIO
105
M 60
(SET)
M 2
106
(SET)
M 3
107
(RST)
M 50
108
(RST)
M 51
109
(RST)

4 SERVIDOR DE DATOS

```
/*****************************************************************************

   MISERVER.C - FUNCIONES PARA LA COMUNICACION CON PCS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 20/10/98

*****************************************************************************/

/********************* Librerias  a incluir ***************************/


#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/uio.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <errno.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/ipc.h>  /*Memoria compartida*/
#include <sys/shm.h>  /*Memoria compartida*/
#include <time.h>     /* Tiempo */

/************************ Definiciones ******************************/

/*#define DEPURA */
#define Port (ushort)  32323 /* numero del puerto del socket */
#define TAMANY  512          /* tamaño del buffer de recepción de datos */
#define TAMANY_PETIT 32

#define SEMKEY1 ((key_t) 4223353L)  /*Semáforo, definiciones en semaforo.c*/
#define SEMKEY2 ((key_t) 4223354L)  /*Semáforo, definiciones en semaforo.c*/
#define SEMKEY3 ((key_t) 4223355L)  /*Semáforo, definiciones en semaforo.c*/

#define SHMKEY1 ((key_t) 1995)           /* Segmentos de memoria compartida */
#define SHMKEY2 ((key_t) 1996)
#define SHMKEY3 ((key_t) 1997)

#define TAMANY_SM1 1000                  /* Tamaño memoria compartida 1 */
#define TAMANY_SM2 1000                  /* Tamaño memoria compartida 2 */
#define TAMANY_SM3 1000                  /* Tamaño memoria compartida 3 */
#define TAMANY_M   100           /* Tamaño del mensaje cliente */

#define PERMS 0644               /* Permisos de acceso */
#define err_sys(A) printf (A "\n")

#define MIS_ADDRS      (u_long)0x00000000  /* Direcciones permitidas */

/************************ Subrutinas del fichero ************************/

void mi_servidor(int clientSocket);
int nuevos_datos_cliente(void);
int mensaje_cliente(void);
int comprueba_consignas_gsi(void);
void medidas_al_buffer(void);
void mensaje_al_buffer(void);
void medidas_texto(void);
void otras_medidas_texto(void);
void medidas_analisis(void);
int  datos_cliente_varios(void);
void ayuda_servidor(void);


int createService(ushort port);
int disconnectClient(int clientSocket);
void  error_bind(int errno);

/******************** Variables comunes del programa ********************/

struct tercera{       /*Estructura per parametres referents als controladors*/
       char nom[3][10];                 /*Nom del controlador*/
       int controlador[3];              /*Numero del controlador */
```



```
       double ph[3];                    /*Valor mesurat de la variable*/
       double temperatura[3];           /*Valor mesurat de la variable*/
       double potencial[3];             /*Valor mesurat de la variable*/
       double oxigeno[3];               /*Valor mesurat de la variable*/
       double mA[3];                    /*Valor mesurat de la variable*/
       double velocidad[3];             /*Valor mesurat de la variable*/
       double caudales[3];
       double set_point_oxigeno[3];
       double set_point_agitacion[3];
       char   entradas_plc[16];
       char   salidas_plc[24];
       char   estados_varios[20];
       int    tiempos[10];
       time_t tiempo;                           /*Tiempo de las variables*/
}medidas;

struct cuarta{    /* Estructura para las consignas */
       char    cambio_oxigeno[3],
               cambio_agitacion[3],
               cambio_caudales[3],
               cambio_tiempos[10];
       double  valor_oxigeno[3],
               valor_agitacion[3],
               valor_caudales[3];
       int     tiempos[10];
       char    cambio_variable[20];
       double valor_variable[20];
}consignas;

struct quinta{     /* Estructura para los analizadores */
       double nitratos;
       time_t t_nitratos;
       double nitritos;
       time_t t_nitritos;
       double amonio;
       time_t t_amonio;
}analisis[5];

struct varios{
       double variable[20];
       time_t t_variable[20];
};

char  buffer[TAMANY];
char  buffer_p[TAMANY_PETIT];

int semid1,
    semid2,
    semid3;
int shmid1,
    shmid2,
    shmid3;
char *direccion1,
     *direccion2,
     *direccion3;


/*****************************************************************************/
/*****************************************************************************/

main()
{
/*struct sockaddr_in clientSockAddr;*/
struct sockaddr clientSockAddr;
int serverSocket;
int clientSocket;
int addrLen;
int j;
int pid;


/*****************************************************************************/
/*  Segmentos de memoria compartida */

if ( (shmid1=shmget(SHMKEY1, TAMANY_SM1, PERMS|IPC_CREAT)) <0)
    err_sys("Miserver: No puedo obtener la memoria compartida 1\n");
if ( (direccion1=(char *)shmat(shmid1, 0, 0)) == (char *)-1)
```

```
    err_sys("Miserver: No puedo unirme a la memoria compartida 1\n");

if ( (shmid2=shmget(SHMKEY2, TAMANY_SM2, PERMS|IPC_CREAT)) <0)
    err_sys("Miserver: No puedo obtener la memoria compartida 2\n");
if ( (direccion2=(char *)shmat(shmid2, 0, 0)) == (char *)-1)
    err_sys("Miserver: No puedo unirme a la memoria compartida 2\n");

if ( (shmid3=shmget(SHMKEY3, TAMANY_SM3, PERMS|IPC_CREAT)) <0)
    err_sys("Miserver: No puedo obtener la memoria compartida 3\n");
if ( (direccion3=(char *)shmat(shmid3, 0, 0)) == (char *)-1)
    err_sys("Miserver: No puedo unirme a la memoria compartida 3\n");

/*********************************************************************/
/* Semáforos para acceder a la memoria compartida */

semid1=sem_create(SEMKEY1, 1);
semid2=sem_create(SEMKEY2, 1);
semid3=sem_create(SEMKEY3, 1);

#ifdef DEPURA
printf("He abierto el semáforo 1,  %lu\n", semid1);
printf("He abierto el semáforo 2,  %lu\n", semid2);
printf("He abierto el semáforo 3,  %lu\n", semid3);
#endif

/*********************************************************************/
/* Inicialización de la memoria compartida */

sem_wait(semid1); /* Espera el semáforo 1 */
memset( direccion1, 0, TAMANY_SM1);  /* Borra memoria compartida 1*/
sem_signal(semid1); /* Deja el semáforo 1 */

sem_wait(semid2); /* Espera el semáforo 2 */
memset( direccion2, 0, TAMANY_SM2); /* Borra memoria compartida 2*/
sem_signal(semid2); /* Deja el semáforo 2 */

sem_wait(semid3); /* Espera el semáforo 3 */
memset( direccion3, 0, TAMANY_SM3); /* Borra memoria compartida 3*/
sem_signal(semid3); /* Deja el semáforo 3 */

/*********************************************************************/


signal (SIGCHLD, SIG_IGN);    /* Ignorar muertes de hijos */

 j=0;

/*********************************************************************/
/*      Se crea un socket y se inicializa el servicio  TCP en el puerto
   seleccionado                                                  */

       serverSocket = createService(Port);
       if (serverSocket==-1) goto el_final;

       addrLen = sizeof(clientSockAddr);

/*********************************************************************/
/*      Se espera hasta que un cliente pida servicio                 */

for (;;)
{
j++;

#ifdef DEPURA
printf("\nServidor: esperando llegada cliente. #%i\n",j);
#endif

       clientSocket=accept(serverSocket, &clientSockAddr, &addrLen);
       if (clientSocket==-1) goto el_final;

#ifdef DEPURA
printf("Servidor: el cliente ha llegado. \n");
#endif

pid=fork();

#ifdef DEPURA
```



```
printf("Proceso: %i\n",pid);
#endif

if (pid==-1) exit(1); /* Error al hacer el proceso hijo */

if (pid==0)           /* Proceso hijo */
      {

#ifdef DEPURA
      printf("\nServidor en marcha ... \n");
#endif


      if(close(serverSocket)==-1)
            printf("ERROR close server\n");

      mi_servidor(clientSocket);

#ifdef DEPURA
printf("\n... fin de servidor.\n");
#endif
       exit(0);

      } /*Fin proceso hijo*/

if (close(clientSocket) == -1)              /* Tratamiento del padre */
      printf("ERROR en close client\n");
}

if (shmdt(direccion1) < 0)
    err_sys("Miserver: No puedo soltar la memoria compartida 1\n");
if (shmctl(shmid1, IPC_RMID, (struct shmid_ds *)0) <0)
    err_sys("Miserver: No puedo eliminar la memoria compartida 1\n");

if (shmdt(direccion2) < 0)
    err_sys("Miserver: No puedo soltar la memoria compartida 2\n");
if (shmctl(shmid2, IPC_RMID, (struct shmid_ds *)0) <0)
    err_sys("Miserver: No puedo eliminar la memoria compartida 2\n");

if (shmdt(direccion3) < 0)
    err_sys("Miserver: No puedo soltar la memoria compartida 3\n");
if (shmctl(shmid3, IPC_RMID, (struct shmid_ds *)0) <0)
    err_sys("Miserver: No puedo eliminar la memoria compartida 3\n");


sem_rm(semid1);
printf("He machacado el semáforo 1\n");
sem_rm(semid2);
printf("He machacado el semáforo 2\n");
sem_rm(semid3);
printf("He machacado el semáforo 3\n");

el_final:

printf ("Eso es todo amigos\n");
}

/*********************************************************************/
/* Intercambio de información con el PC */

void mi_servidor(int clientSocket)
{
char caracter[1];
int fin = 0;


do   /* Mientras existan peticiones ...*/
{
caracter[0]=0;
/* Recibimos petición */
recv(clientSocket, caracter, sizeof(caracter), 0);

#ifdef DEPURA
printf ("Petición: %c\n", caracter[0]);
#endif

/* Actuamos en función de la petición*/
```

```
switch(caracter[0])
   {
   case 'S':  /* Establecer nuevos datos */
        recv(clientSocket, buffer, sizeof(buffer), 0);
        if (nuevos_datos_cliente() == 1) /* Comprobación de datos */
               {
               caracter[0]='O';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
        else           /* Si no son correctos devuelve N */
               {
               caracter[0]='N';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
   break;
   case 'M':
        recv(clientSocket, buffer, sizeof(buffer), 0);
        if (mensaje_cliente() == 1) /* Comprobación de datos */
               {
               caracter[0]='O';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
        else           /* Si no son correctos devuelve N */
               {
               caracter[0]='N';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
   break;
   case 'R':
        medidas_al_buffer();
        send(clientSocket, buffer, sizeof(buffer), 0);
   break;
   case 'C':
        if ( comprueba_consignas_gsi()== 1)
          {
          caracter[0]='S';
          send(clientSocket, caracter , sizeof(caracter), 0);

          send(clientSocket, buffer, sizeof(buffer), 0);
          recv(clientSocket, caracter, sizeof(caracter), 0);
/* Si se ha conseguido la conexión, borra las consignas de la estructura */
          if (caracter[0]=='O')
               {
               sem_wait(semid2);     /* Espera el semáforo 2 */
               memset( direccion2, 0, sizeof(consignas));
               sem_signal(semid2);   /* Deja el semáforo 2 */
               }
          }
        else
          {
          caracter[0]='N';
          send(clientSocket, caracter , sizeof(caracter), 0);
          }
   break;
   case 'W': /* Warning, mensaje de la planta */
        mensaje_al_buffer();
        send(clientSocket, buffer, sizeof(buffer), 0);
   break;
   case 'F':  /* Fin de la conexion */
        disconnectClient(clientSocket);
        fin = 1;
   break;
   case 'E': /* Tratamiento especial de datos */
        medidas_texto();
        send(clientSocket, buffer, sizeof(buffer), 0);
   break;
   case 'O': /* Tratamiento especial de datos */
        otras_medidas_texto();
        send(clientSocket, buffer, sizeof(buffer), 0);
   break;
   case 'A': /* Datos recibidos de los analizadores */
        recv(clientSocket, buffer, sizeof(buffer), 0);
        if (nuevos_datos_analizadores() == 1) /* Comprobación de datos */
               {
               caracter[0]='O';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
```



```
        else           /* Si no son correctos devuelve N */
               {
               caracter[0]='N';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
   break;
   case 'N': /* Valores de los analizadores para enviar*/
        medidas_analisis();
        send(clientSocket, buffer, sizeof(buffer), 0);
   break;
   case 'V':
        recv(clientSocket, buffer_p, sizeof(buffer_p), 0);
        if (datos_cliente_varios() == 1) /* Comprobación de datos */
               {
               caracter[0]='O';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
        else           /* Si no son correctos devuelve N */
               {
               caracter[0]='N';
               send(clientSocket, caracter , sizeof(caracter), 0);
               }
   break;
   case 'H': /* Ayuda del servidor */
   case'?':
   case 'h':
        ayuda_servidor();
        send(clientSocket, buffer, sizeof(buffer), 0);
   break;
   default:   /* Si se trata de una orden no contemplada, se desconecta */
#ifdef DEPURA
        printf ("Por defecto ... \n");
#endif
        disconnectClient(clientSocket);
        fin = 1;
   break;
        }
} while (fin==0);

}


/*************************************************************************/
/* Toma el buffer mandado por el cliente y da valores a la estructura
   de las medidas */

int nuevos_datos_cliente(void)
{
struct segunda{
        char    nom[3][10],
                controlador[3][10],
                ph[3][10],
                temperatura[3][10],
                potencial[3][10],
                oxigeno[3][10],
                mA[3][10],
                velocidad[3][10],
                caudales[3][10],
                set_point_oxigeno[3][10],
                set_point_agitacion[3][10],
                entradas_plc[16],
                salidas_plc[24],
                estados_varios[20],
                tiempos[10][10];
}datos;

int i=0;
int  chequeo1=0L,
     chequeo2=0L;
time_t tiempo;

for (i=0;i<(TAMANY-12);i++) chequeo1+=buffer[i];

chequeo2= atoi(&buffer[500]);

if (chequeo1==chequeo2)
     {
```

```
int sem_create( key_t key, int initval)
{
register int  id, semval;

union semun
{
int             val;
struct semid_ds  *buf;
ushort          *array;
} semctl_arg;

   if (key == IPC_PRIVATE) return (-1);
   else if (key == (key_t) -1) return (-1);

again:
   if ( (id = semget(key, 3, PERMS | IPC_CREAT)) < 0)
      return (-1);
   if (semop(id, &op_lock[0], 2) < 0)
      {
      if (errno == EINVAL) goto again;
      err_sys("can't lock");
      }

   if ( (semval = semctl(id, 1, GETVAL, 0)) < 0)
      err_sys("can't GETVAL");
   if (semval == 0)
      {
      semctl_arg.val = initval;
      if (semctl(id, 0, SETVAL, semctl_arg) < 0)
         err_sys("can SETVAL[0]");

      semctl_arg.val = BIGCOUNT;
      if (semctl(id, 1, SETVAL, semctl_arg) < 0)
         err_sys("can SETVAL[1]");
      }
    if (semop(id, &op_endcreate[0], 2) < 0)
         err_sys("can't end create");

    return(id);
}

/*******************************************************/
/* Abre un semáforo que ya existe.
   Devuelve la identidad del semáforo.   */

int sem_open (key_t key)
{
register int  id;

   if (key == IPC_PRIVATE) return (-1);
   else if (key == (key_t) -1) return (-1);

   if ( (id = semget(key, 3, 0)) < 0)
     return (-1);

   if (semop(id, &op_open[0], 1) < 0)
     err_sys("can't open");

   return (id);
}

/*******************************************************/
/* Elimina un semáforo.
   Debe ser llamada por el servidor. */

void sem_rm(int id)
{

  if (semctl(id, 0 ,IPC_RMID, 0) < 0)
      err_sys("can't IPC_RMID");
}

/*******************************************************/
/* Cierra un semáforo. */

void sem_close(int id)
```



```
{
register int  semval;

  if (semop(id, &op_close[0], 3) < 0)
       err_sys("can't semop");

  if (semval=semctl(id, 1, GETVAL, 0) < 0)
       err_sys("can't GETVAL");
  if (semval > BIGCOUNT)
       err_sys("sem[1] > BIGCOUNT");
  else if (semval == BIGCOUNT)
       sem_rm(id);
  else
       if (semop(id, &op_unlock[0], 1) < 0)
          err_sys("can't unlock");
}

/*******************************************************/
/* Espera hasta que el semáforo es mayor que cero.
   Después le resta 1 y vuelve */

void sem_wait(int id)
{
  sem_op(id,-1);
}

/*******************************************************/
/* Incrementa un semáforo en 1 */

void sem_signal(int id)
{
  sem_op(id,1);
}

/*******************************************************/
/* Operación general sobre un semáforo.  */

void sem_op(int id, int value)
{
  if ( (op_op[0].sem_op = value) == 0)
       err_sys("can't have value == 0");

  if (semop(id, &op_op[0], 1) < 0)
       err_sys("sem_op error");
}

/*******************************************************/
```

```
/*************************************************************************

   MIGSI.C - PROGRAMA PARA ACCEDER A LA MEMORIA COMPARTIDA Y MOSTRAR LOS DATOS
             IN-LINE POR PANTALLA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 26/10/98

*************************************************************************/

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <time.h>     /* Tiempo */

#define SEMKEY1 ((key_t) 4223353L)  /*Semáforo, definiciones en semaforo.c*/
#define SEMKEY2 ((key_t) 4223354L)  /*Semáforo, definiciones en semaforo.c*/

#define SHMKEY1 ((key_t) 1995)          /* Segmentos de memoria compartida */
#define SHMKEY2 ((key_t) 1996)

#define TAMANY_SM1 1000                 /* Tamaño memoria compartida 1 */
#define TAMANY_SM2 1000                 /* Tamaño memoria compartida 2 */
#define TAMANY_M   100            /* Tamaño del mensaje cliente */


#define PERMS 0644                /* Permisos de acceso */
#define err_sys(A) printf (A)


struct tercera{       /*Estructura per parametres referents als controladors*/
       char nom[3][10];                    /*Nom del controlador*/
       int controlador[3];                /*Numero del controlador */
       double ph[3];                       /*Valor mesurat de la variable*/
       double temperatura[3];              /*Valor mesurat de la variable*/
       double potencial[3];                /*Valor mesurat de la variable*/
       double oxigeno[3];                  /*Valor mesurat de la variable*/
       double mA[3];                       /*Valor mesurat de la variable*/
       double velocidad[3];                /*Valor mesurat de la variable*/
       double caudales[3];
       double set_point_oxigeno[3];
       double set_point_agitacion[3];
       char   entradas_plc[16];
       char   salidas_plc[24];
       char   estados_varios[20];
       int    tiempos[10];
       time_t tiempo;                            /*Tiempo de las variables*/
}medidas;

struct cuarta{    /* Estructura para las consignas */
       char    cambio_oxigeno[3],
               cambio_agitacion[3],
               cambio_caudales[3],
               cambio_tiempos[10];
       double  valor_oxigeno[3],
               valor_agitacion[3],
               valor_caudales[3];
       int     tiempos[10];
       char    cambio_variable[20];
       double  valor_variable[20];
}consignas;


int semid1,
    semid2;
int shmid1,
    shmid2;
char *direccion1,
     *direccion2;


void main(void)
{
int i;
struct tm *mihora;
```



```
time_t hora_actual;
char mensaje_cliente[TAMANY_M];

/*************************************************************************/
/*  Segmentos de memoria compartida */

if ( (shmid1=shmget(SHMKEY1, TAMANY_SM1, PERMS )) <0)
    err_sys("Miserver: No puedo obtener la memoria compartida 1\n");
if ( (direccion1=(char *)shmat(shmid1, 0, 0)) == (char *)-1)
    err_sys("Miserver: No puedo unirme a la memoria compartida 1\n");

if ( (shmid2=shmget(SHMKEY2,  TAMANY_SM2, PERMS )) <0)
    err_sys("Migsi: No puedo obtener la memoria compartida 2\n");
if ( (direccion2=(char *)shmat(shmid2, 0, 0)) == (char *)-1)
    err_sys("Migsi: No puedo unirme a la memoria compartida 2\n");


if ((shmid1==-1) || (shmid2==-1) ) goto elfin;

semid1=sem_open(SEMKEY1);
semid2=sem_open(SEMKEY2);

if ( (semid1 == -1) || (semid2 == -1) ) goto elfin;


sem_wait(semid1);     /* Esperando el semáforo 1 */
memcpy(&medidas,direccion1,sizeof(medidas));
memcpy(mensaje_cliente, &direccion1[500], sizeof(mensaje_cliente));
sem_signal(semid1);   /* Deja el semáforo 1*/

sem_wait(semid2);     /* Esperando el semáforo 2 */
memcpy(&consignas,direccion2,sizeof(consignas));
sem_signal(semid2);   /* Deja el semáforo 2*/

printf("Medidas: \n\n");

printf(" \t%s\t%s\t%s\n", medidas.nom[0],medidas.nom[1],medidas.nom[2]);
printf("                                                                \t%i\t\t%i\t\t%i\n",
medidas.controlador[0],medidas.controlador[1],medidas.controlador[2]);
printf("pH: \t%f\t%f\t%f\n", medidas.ph[0],medidas.ph[1],medidas.ph[2]);
printf("T:                                                              \t%f\t%f\t%f\n",
medidas.temperatura[0],medidas.temperatura[1],medidas.temperatura[2]);
printf("mV: \t%f\t%f\t%f\n", medidas.potencial[0],medidas.potencial[1],medidas.potencial[2]);
printf("O2: \t%f\t%f\t%f\n", medidas.oxigeno[0],medidas.oxigeno[1],medidas.oxigeno[2]);
printf("mA: \t%f\t%f\t%f\n", medidas.mA[0],medidas.mA[1],medidas.mA[2]);
printf("V:  \t%f\t%f\t%f\n", medidas.velocidad[0],medidas.velocidad[1],medidas.velocidad[2]);
printf("SP          O2:          \t%f\t%f\t%f\n",          medidas.set_point_oxigeno[0],
medidas.set_point_oxigeno[1],medidas.set_point_oxigeno[2]);
printf("SP V:  \t%f\t%f\t%f\n", medidas.set_point_agitacion[0],
medidas.set_point_agitacion[1],medidas.set_point_agitacion[2]);

printf("\nEntradas PLC:    ");
for (i=0;i<16;i++) printf (" %c",medidas.entradas_plc[i]);
printf("\nSalidas  PLC:    ");
for (i=0;i<24;i++) printf (" %c",medidas.salidas_plc[i]);
printf("\nEstados varios:  ");
for (i=0;i<20;i++) printf (" %c",medidas.estados_varios[i]);
printf("\nTiempos:         ");
for (i=0;i<10;i++) printf (" %i",medidas.tiempos[i]);
printf("\ncaudales:                                                       \t%f\t%f\t%f\n",
medidas.caudales[0],medidas.caudales[1],medidas.caudales[2]);


printf("\n\nConsignas: \n");

printf("\nCambio O2: ");
for (i=0;i<3;i++) printf (" \t%c",consignas.cambio_oxigeno[i]);
printf("\nCambio Vel: ");
for (i=0;i<3;i++) printf (" \t%c",consignas.cambio_agitacion[i]);
printf("\nCambio Caud:  ");
for (i=0;i<3;i++) printf (" \t%c",consignas.cambio_caudales[i]);
printf("\nCambio tiempos: ");
for (i=0;i<10;i++) printf (" \t%c",consignas.cambio_tiempos[i]);
printf("\nCambio variables: ");
for (i=0;i<20;i++) printf (" \t%c",consignas.cambio_variable[i]);
```

```
printf("\nValor O2: ");
for (i=0;i<3;i++) printf (" \t%f",consignas.valor_oxigeno[i]);
printf("\nValor Vel: ");
for (i=0;i<3;i++) printf (" \t%f",consignas.valor_agitacion[i]);
printf("\nValor Caud: ");
for (i=0;i<3;i++) printf (" \t%f",consignas.valor_caudales[i]);
printf("\nValor tiempos: ");
for (i=0;i<10;i++) printf ("\t%i",consignas.tiempos[i]);
printf("\nOtras variables: ");
for (i=0;i<20;i++) printf ("  %f",consignas.valor_variable[i]);


mihora=localtime( &(medidas.tiempo) );
printf("\n\nHora medidas:  %2i %2i %2i\n",
   mihora->tm_hour, mihora->tm_min, mihora->tm_sec);

time( &hora_actual );
mihora=localtime( &hora_actual );
printf("Hora actual:   %2i %2i %2i\n",
   mihora->tm_hour, mihora->tm_min, mihora->tm_sec);

while(mensaje_cliente[i]!=10 && i<(TAMANY_M-2)) i++;
mensaje_cliente[i+1]=0;

printf( "Mensaje:  %s\n", mensaje_cliente);

elfin:
if (shmdt(direccion1) < 0)
    err_sys("Migsi: No puedo soltar la memoria compartida 1\n");
if (shmdt(direccion2) < 0)
    err_sys("Migsi: No puedo soltar la memoria compartida 2\n");

sem_close(semid1);
sem_close(semid2);

exit(0);
}
```



```
/*************************************************************************

   MIGSIA.C - PROGRAMA PARA ACCEDER A LA MEMORIA COMPARTIDA Y MOSTRAR LOS
              DATOS ANALITICOS POR PANTALLA

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 5/6/97

*************************************************************************/

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <time.h>     /* Tiempo */

#define SEMKEY3 ((key_t) 4223355L)  /*Semáforo, definiciones en semaforo.c*/

#define SHMKEY3 ((key_t) 1997)

#define TAMANY_SM3 1000                  /* Tamaño memoria compartida 3 */

#define PERMS 0644               /* Permisos de acceso */
#define err_sys(A) printf (A)


struct quinta{     /* Estructura para los analizadores */
       double nitratos;
       time_t t_nitratos;
       double nitritos;
       time_t t_nitritos;
       double amonio;
       time_t t_amonio;
}analisis[5];

int  semid3;
int  shmid3;
char *direccion3;


void main(void)
{
int i;
struct tm *mihora;
time_t hora_actual;

/*************************************************************************/
/*  Segmentos de memoria compartida */


if ( (shmid3=shmget(SHMKEY3, TAMANY_SM3, PERMS )) <0)
    err_sys("Migsi: No puedo obtener la memoria compartida 3\n");
if ( (direccion3=(char *)shmat(shmid3, 0, 0)) == (char *)-1)
    err_sys("Migsi: No puedo unirme a la memoria compartida 3\n");


if (shmid3==-1) goto elfin;

semid3=sem_open(SEMKEY3);

if (semid3 == -1) goto elfin;


sem_wait(semid3);     /* Esperando el semáforo 3 */
memcpy(analisis,direccion3,sizeof(analisis));
sem_signal(semid3);   /* Deja el semáforo 3*/

printf("\nÚltimos valores analizados: \n\n");
for (i=0;i<5;i++)
       {
       printf("Punto: %i\n",i);
       printf("Nitratos: %f\n",analisis[i].nitratos);
       mihora=localtime( &(analisis[i].t_nitratos) );
       printf("T nitratos:  %2i %2i %2i, %2i/%2i/%2i\n",
              mihora->tm_hour, mihora->tm_min, mihora->tm_sec,
              mihora->tm_mday, 1+mihora->tm_mon, mihora->tm_year);
```

```
        printf("Nitritos: %f\n",analisis[i].nitritos);
        mihora=localtime( &(analisis[i].t_nitritos) );
        printf("T nitritos:  %2i %2i %2i, %2i/%2i/%2i\n",
               mihora->tm_hour, mihora->tm_min, mihora->tm_sec,
               mihora->tm_mday, 1+mihora->tm_mon, mihora->tm_year);
        printf("Amonio:    %f\n",analisis[i].amonio);
        mihora=localtime( &(analisis[i].t_amonio) );
        printf("T amonio:     %2i %2i %2i, %2i/%2i/%2i\n",
               mihora->tm_hour, mihora->tm_min, mihora->tm_sec,
               mihora->tm_mday, 1+mihora->tm_mon, mihora->tm_year);
        }


elfin:
if (shmdt(direccion3) < 0)
    err_sys("Migsi: No puedo soltar la memoria compartida 3\n");

sem_close(semid3);

exit(0);
}
```



```
/*****************************************************************************

   PLANTATR2.C - PROGRAMA PARA ACCEDER A LA MEMORIA COMPARTIDA Y CREAR
                 AUTOMATICAMENTE UNA PAGINA WEB CON DATOS IN-LINE

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 19/12/97

*****************************************************************************/

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <time.h>     /* Tiempo */

#define SEMKEY1 ((key_t) 4223353L)  /*Semáforo, definiciones en semaforo.c*/
#define SEMKEY2 ((key_t) 4223354L)  /*Semáforo, definiciones en semaforo.c*/

#define SHMKEY1 ((key_t) 1995)           /* Segmentos de memoria compartida */
#define SHMKEY2 ((key_t) 1996)

#define TAMANY_SM1 1000                  /* Tamaño memoria compartida 1 */
#define TAMANY_SM2 1000                  /* Tamaño memoria compartida 2 */
#define TAMANY_M   100            /* Tamaño del mensaje cliente */


#define PERMS 0644                /* Permisos de acceso */
#define err_sys(A) printf (A)


struct tercera{       /*Estructura per parametres referents als controladors*/
       char nom[3][10];                    /*Nom del controlador*/
       int controlador[3];                /*Numero del controlador */
       double ph[3];                       /*Valor mesurat de la variable*/
       double temperatura[3];              /*Valor mesurat de la variable*/
       double potencial[3];                /*Valor mesurat de la variable*/
       double oxigeno[3];                  /*Valor mesurat de la variable*/
       double mA[3];                       /*Valor mesurat de la variable*/
       double velocidad[3];                /*Valor mesurat de la variable*/
       double caudales[3];
       double set_point_oxigeno[3];
       double set_point_agitacion[3];
       char   entradas_plc[16];
       char   salidas_plc[24];
       char   estados_varios[20];
       int    tiempos[10];
       time_t tiempo;                              /*Tiempo de las variables*/
}medidas;

struct cuarta{    /* Estructura para las consignas */
       char    cambio_oxigeno[3],
               cambio_agitacion[3],
              cambio_caudales[3],
              cambio_tiempos[10];
       double  valor_oxigeno[3],
               valor_agitacion[3],
               valor_caudales[3];
       int     tiempos[10];
}consignas;


int semid1,
    semid2;
int shmid1,
    shmid2;
char *direccion1,
     *direccion2;
FILE *fichero_in;
FILE *fichero_out;

char frase[30];
char buffer[1500];

void main(void)
{
```

```
int i;
struct tm *mihora;
time_t hora_actual;
char mensaje_cliente[TAMANY_M];

/*********************************************************************/
/*  Segmentos de memoria compartida */

if ( (shmid1=shmget(SHMKEY1, TAMANY_SM1, PERMS )) <0)
    err_sys("Miserver: No puedo obtener la memoria compartida 1\n");
if ( (direccion1=(char *)shmat(shmid1, 0, 0)) == (char *)-1)
    err_sys("Miserver: No puedo unirme a la memoria compartida 1\n");

if ( (shmid2=shmget(SHMKEY2,  TAMANY_SM2, PERMS )) <0)
    err_sys("Migsi: No puedo obtener la memoria compartida 2\n");
if ( (direccion2=(char *)shmat(shmid2, 0, 0)) == (char *)-1)
    err_sys("Migsi: No puedo unirme a la memoria compartida 2\n");


if ((shmid1==-1) || (shmid2==-1) ) goto elfin;

semid1=sem_open(SEMKEY1);
semid2=sem_open(SEMKEY2);

if ( (semid1 == -1) || (semid2 == -1) ) goto elfin;

do{/*****************************/

sem_wait(semid1);      /* Esperando el semáforo 1 */
memcpy(&medidas,direccion1,sizeof(medidas));
memcpy(mensaje_cliente, &direccion1[500], sizeof(mensaje_cliente));
sem_signal(semid1);    /* Deja el semáforo 1*/

sem_wait(semid2);      /* Esperando el semáforo 2 */
memcpy(&consignas,direccion2,sizeof(consignas));
sem_signal(semid2);    /* Deja el semáforo 2*/

memset (frase,0,sizeof(frase));
memset (buffer,0,sizeof(buffer));

fichero_out=fopen("/quantum/httpd/htdocs/depuradoras/plantatr2.html","wb");

printf("Generando html ... \n\n");

fprintf(fichero_out,"<HTML>
<META HTTP-EQUIV=REFRESH CONTENT=20; URL=http://eq3.uab.es/depuradoras/plantatr2.html>
<HEAD>
<TITLE>Datos en tiempo real planta piloto</TITLE>
</HEAD>
<BODY bgcolor=#ffffff LINK=#0000ff VLINK=#800080>

<FONT FACE=Arial COLOR=#ff0000><P ALIGN=CENTER>PLANTA PILOTO: DATOS ACTUALIZADOS EN TIEMPO
REAL</P>
</FONT><FONT FACE=Arial SIZE=2><P> </P></FONT>
<P  ALIGN=CENTER><CENTER><TABLE  BORDER  CELLSPACING=1  bgcolor=#d1e9ff  BORDERCOLOR=#000000
CELLPADDING=5 WIDTH=683>
<TR><TD WIDTH=22% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41>
<P> </TD>
<TD WIDTH=25% VALIGN=MIDDLE COLSPAN=5 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P ALIGN=CENTER>REACTOR 1</B></FONT></TD>
<TD WIDTH=28% VALIGN=MIDDLE COLSPAN=4 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P ALIGN=CENTER>REACTOR 2</B></FONT></TD>
<TD WIDTH=26% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P ALIGN=CENTER>REACTOR 3</B></FONT></TD>
</TR>
<TR><TD WIDTH=22% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P>pH</B></FONT></TD>
<TD WIDTH=25% VALIGN=MIDDLE COLSPAN=5 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.ph[0]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=28% VALIGN=MIDDLE COLSPAN=4 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.ph[1]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
```



```
<TD WIDTH=26% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.ph[2]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
</TR>
<TR><TD WIDTH=22% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P>Temperatura (C)</B></FONT></TD>
<TD WIDTH=25% VALIGN=MIDDLE COLSPAN=5 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.temperatura[0]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=28% VALIGN=MIDDLE COLSPAN=4 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.temperatura[1]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=26% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.temperatura[2]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
</TR>
<TR><TD WIDTH=22% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P>Ox&iacute;geno (ppm)</B></FONT></TD>
<TD WIDTH=25% VALIGN=MIDDLE COLSPAN=5 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.oxigeno[0]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=28% VALIGN=MIDDLE COLSPAN=4 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.oxigeno[1]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=26% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41><P>","%s");
sprintf(frase,"%4.1f",medidas.oxigeno[2]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
</TR>
<TR><TD WIDTH=22% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41>
<B><FONT FACE=Arial SIZE=2 COLOR=#0000ff><P>Redox (mV)</B></FONT></TD>
<TD WIDTH=25% VALIGN=MIDDLE COLSPAN=5 HEIGHT=41><P>","%s");
sprintf(frase,"%4f",medidas.potencial[0]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=28% VALIGN=MIDDLE COLSPAN=4 HEIGHT=41><P>","%s");
sprintf(frase,"%4f",medidas.potencial[1]);
fwrite(frase,4,1,fichero_out);

fprintf(fichero_out,"</P></TD>
<TD WIDTH=26% VALIGN=MIDDLE COLSPAN=2 HEIGHT=41><P>","%s");
sprintf(frase,"%4f",medidas.potencial[2]);
fwrite(frase,4,1,fichero_out);


fprintf(fichero_out,"</P></TD>
</TR>
<TR><TD WIDTH=42% VALIGN=TOP COLSPAN=6 HEIGHT=38>
<P ALIGN=CENTER>Hora de la adquisici&oacute;n de datos:</TD>
<TD WIDTH=58% VALIGN=MIDDLE COLSPAN=7 HEIGHT=38><P>","%s");
sprintf(frase, "%s", ctime(&medidas.tiempo));
fwrite(frase,27,1,fichero_out);

fprintf(fichero_out,"</P></TD>
</TR>
</TABLE>
</CENTER></P>

<FONT FACE=Arial SIZE=2><P> </P></FONT>
<TABLE CELLSPACING=0 BORDER=0 CELLPADDING=5 WIDTH=791>
<TR><TD WIDTH=50% VALIGN=TOP>
<P> </TD>
<TD WIDTH=50% VALIGN=TOP>
```

```
<P> </TD>
</TR>
</TABLE>

<FONT FACE=Arial SIZE=2><P> </P>
<P> </P>
</FONT><P><A HREF=http://eq3.uab.es/index.html>Volver al &iacute;ndice de Ingenier&iacute;a
qu&iacute;mica</A></P></BODY>
</HTML>","%s");

fclose(fichero_out);


sleep(30);

}while(1L);/***********end while****************/

elfin:
if (shmdt(direccion1) < 0)
    err_sys("Migsi: No puedo soltar la memoria compartida 1\n");
if (shmdt(direccion2) < 0)
    err_sys("Migsi: No puedo soltar la memoria compartida 2\n");

sem_close(semid1);
sem_close(semid2);

exit(0);
}
```

5 COMUNICACIÓN GSI-SERVIDOR DE DATOS

```
/*************************************************************************

   GSIJB.C - PROGRAMA PRINCIPAL PARA LA COMUNICACIÓN GSI-SERVIDOR DE DATOS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 31/10/98

*************************************************************************/


#include <stdio.h>
#include <string.h>
#include <time.h>     /* Tiempo */
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include "gsi.h"
#include <stdlib.h>

#define SEMKEY1 ((key_t) 4223353L)  /*Semáforo, definiciones en semaforo.c*/
#define SEMKEY2 ((key_t) 4223354L)  /*Semáforo, definiciones en semaforo.c*/
#define SEMKEY3 ((key_t) 4223355L)  /*Semáforo, definiciones en semaforo.c*/

#define SHMKEY1 ((key_t) 1995)          /* Segmentos de memoria compartida */
#define SHMKEY2 ((key_t) 1996)
#define SHMKEY3 ((key_t) 1997)

#define TAMANY_SM1 1000                 /* Tamaño memoria compartida 1 */
#define TAMANY_SM2 1000                 /* Tamaño memoria compartida 2 */
#define TAMANY_SM3 1000                 /* Tamaño memoria compartida 3 */
#define TAMANY_M   100          /* Tamaño del mensaje cliente */

#define PERMS 0644              /* Permisos de acceso */
#define err_sys(A) printf (A "\n")

#define TCPIP_PORT_NUMBER  22041
/*#define DEBUG*/

/*************************************************************************/

struct tercera{       /*Estructura per parametres referents als controladors*/
       char nom[3][10];                  /*Nom del controlador*/
       int controlador[3];              /*Numero del controlador */
       double ph[3];                     /*Valor mesurat de la variable*/
       double temperatura[3];            /*Valor mesurat de la variable*/
       double potencial[3];              /*Valor mesurat de la variable*/
       double oxigeno[3];                /*Valor mesurat de la variable*/
       double mA[3];                     /*Valor mesurat de la variable*/
       double velocidad[3];              /*Valor mesurat de la variable*/
       double caudales[3];
       double set_point_oxigeno[3];
       double set_point_velocidad[3];
       char   entradas_plc[16];
       char   salidas_plc[24];
       char   estados_varios[20];
       int    tiempos[10];
       time_t tiempo;                          /*Tiempo de las variables*/
}medidas;
;

struct cuarta{    /* Estructura para las consignas */
       char    cambio_oxigeno[3],
               cambio_agitacion[3],
              cambio_caudales[3],
              cambio_tiempos[10];
       double  valor_oxigeno[3],
               valor_agitacion[3],
               valor_caudales[3];
       int     tiempos[10];
       char    cambio_variable[20];
       double valor_variable[20];
}consignas;

struct quinta{     /* Estructura para los analizadores */
       double nitratos;
       time_t t_nitratos;
       double nitritos;
```



```
       time_t t_nitritos;
       double amonio;
       time_t t_amonio;
}analisis[5];

struct de_los_indices{
    icp_int m_ph[3];                /* Indices medida de ph */
    icp_int m_temperatura[3];       /* Indices medida temperatura */
    icp_int m_potencial[3];         /* Indices medida redox */
    icp_int m_oxigeno[3];           /* Indices medida oxigeno */
    icp_int m_mA[3];                /* Indices medida miliamperios */
    icp_int m_velocidad[3];         /* Indices medida velocidad */
    icp_int m_nitratos[5];          /* Indices medida nitratos */
    icp_int m_nitritos[5];          /* Indices medida nitritos */
    icp_int m_amonio[5];            /* Indices medida amonio */
    icp_int m_sp_oxigeno[3];        /* Indices medida consigna oxigeno */
    icp_int m_sp_velocidad[3];            /* Indices medida consigna agitacion */
    icp_int m_caudales[3];          /* Indices medida caudales */
    icp_int m_entradas_plc[16];           /* Indices medida entradas plc */
    icp_int m_salidas_plc[24];            /* Indices medida salidas plc */
    icp_int m_estados_varios[20];   /* Indices medida estados */
    icp_int m_tiempos[10];          /* Indices medida tiempos */
    icp_int m_variable[20];         /* Indices medida variables */
    icp_int c_oxigeno[3];           /* Indices consigna oxigeno */
    icp_int c_velocidad[3];         /* Indices consigna agitacion */
    icp_int c_caudales[3];          /* Indices consigna caudales */
    icp_int c_tiempos[10];          /* Indices consigna tiempos */
    icp_int c_variable[20];         /* Indices variables varias */
    icp_int m_plc[2];               /* Indices E/S PLC */
}indice;

/* 0,1,2 Reactores 1,2,3; 3 -> salida; 4 -> anaerobio */

struct varios{
       double variable[20];
       time_t t_variable[20];
}var;

int semid1,
    semid2,
    semid3;
int shmid1,
    shmid2,
    shmid3;
char *direccion1,
     *direccion2,
     *direccion3;


/*************************************************************************/
/* Devuelve el puerto a utilizar en la comunicación G2 <--> GSI  */

gsi_int gsi_get_tcp_port()
{
    return(TCPIP_PORT_NUMBER);
}

/*************************************************************************/
/* Inicializa el sistem externo, buffers, contadores, variables comunes... */

gsi_int gsi_initialize_context(remote_process_init_string, length)
    char   *remote_process_init_string;
    gsi_int  length;
{
if ( (shmid1=shmget(SHMKEY1, TAMANY_SM1, PERMS)) <0)
    err_sys("gsi_jb: No puedo obtener la memoria compartida 1\n");
if ( (direccion1=(char *)shmat(shmid1, 0, 0)) == (char *)-1)
    err_sys("gsi_jb: No puedo unirme a la memoria compartida 1\n");

if ( (shmid2=shmget(SHMKEY2, TAMANY_SM2, PERMS )) <0)
    err_sys("gsi_jb: No puedo obtener la memoria compartida 2\n");
if ( (direccion2=(char *)shmat(shmid2, 0, 0)) == (char *)-1)
    err_sys("gsi_jb: No puedo unirme a la memoria compartida 2\n");

if ( (shmid3=shmget(SHMKEY3, TAMANY_SM3, PERMS )) <0)
    err_sys("Migsi: No puedo obtener la memoria compartida 3\n");
if ( (direccion3=(char *)shmat(shmid3, 0, 0)) == (char *)-1)
```

```
    err_sys("Migsi: No puedo unirme a la memoria compartida 3\n");

if ((shmid1==-1) || (shmid2==-1) || (shmid3==-1)) return(GSI_FAIL);

semid1=sem_open(SEMKEY1);
semid2=sem_open(SEMKEY2);
semid3=sem_open(SEMKEY3);

if ( (semid1 == -1) || (semid2 == -1) || (semid3 == -1) ) return(GSI_FAIL);

memset(&indice,0,sizeof(indice));
memset(&consignas,0,sizeof(consignas));

#ifdef DEBUG
    printf ("gsi_initialize_context in context %d\n",current_context);
#endif


    return (GSI_SUCCESS);
}

/*************************************************************************/
/* GSI llama a la función cuando se define o actualiza un objeto, antes de
   ser utlizado por primera vez. Se registra el índice de la variable.   */

gsi_int gsi_def_obj(obj_def_ptr, unused)
    obj_def_type  *obj_def_ptr, *unused;
{
int i=0;
char parametro[]="REACTOR0";

/*printf("Entrando en gsi_def_obj\n");*/
if ((obj_def_ptr->param_1.p_type!=3) || (obj_def_ptr->param_2.p_type!=3)
    || (obj_def_ptr->param_3.p_type!=3)) return (GSI_FAIL);

/*printf("tipo param1 %i\n", obj_def_ptr->param_1.p_type);*/
/*printf("Param1: %s\n",gsi_str_of(obj_def_ptr->param_1));*/

#ifdef DEBUG
printf("%s\n", obj_def_ptr->param_1.p_value.symbol_p);
printf("%s\n", obj_def_ptr->param_2.p_value.symbol_p);
printf("%s\n", obj_def_ptr->param_3.p_value.symbol_p);
#endif


/* Reactores 1-3 */
for (i=0;i<3;i++)
{
parametro[7]=49+i;

if(!strcmp(obj_def_ptr->param_1.p_value.symbol_p,parametro))
   {
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"MEDIDA"))
      {
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"PH"))
        {
        indice.m_ph[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"T"))
        {
        indice.m_temperatura[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"MV"))
        {
        indice.m_potencial[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"O2"))
        {
        indice.m_oxigeno[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"MA"))
        {
        indice.m_mA[i]=obj_def_ptr->obj_index;
```



```
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"VELOCIDAD"))
        {
        indice.m_velocidad[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"NITRATOS"))
        {
        indice.m_nitratos[i+1]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"NITRITOS"))
        {
        indice.m_nitritos[i+1]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"AMONIO"))
        {
        indice.m_amonio[i+1]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"SPOXIGENO"))
        {
        indice.m_sp_oxigeno[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"SPVELOCIDAD"))
        {
        indice.m_sp_velocidad[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"CONSIGNA"))
      {
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"O2"))
        {
        indice.c_oxigeno[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"AGITACION"))
        {
        indice.c_velocidad[i]=obj_def_ptr->obj_index;
        goto bien;
        }
      }
   }
}

/* Salida del sedimentador*/
if(!strcmp(obj_def_ptr->param_1.p_value.symbol_p,"SALIDA"))
   {
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"MEDIDA"))
      {
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"NITRATOS"))
        {
        indice.m_nitratos[4]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"NITRITOS"))
        {
        indice.m_nitritos[4]=obj_def_ptr->obj_index;
        goto bien;
        }
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"AMONIO"))
        {
        indice.m_amonio[4]=obj_def_ptr->obj_index;
        goto bien;
        }
      }
   }
/* Reactor anóxico */
if(!strcmp(obj_def_ptr->param_1.p_value.symbol_p,"REACTOR0"))
   {
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"MEDIDA"))
      {
      if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"NITRATOS"))
```

```
        {
        indice.m_nitratos[0]=obj_def_ptr->obj_index;
        goto bien;
        }
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"NITRITOS"))
        {
        indice.m_nitritos[0]=obj_def_ptr->obj_index;
        goto bien;
        }
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"AMONIO"))
        {
        indice.m_amonio[0]=obj_def_ptr->obj_index;
        goto bien;
        }
      }
    }

/* PLC*/

memset(parametro,0,sizeof(parametro));

if(!strcmp(obj_def_ptr->param_1.p_value.symbol_p,"PLC"))
   {
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"ENTRADA"))
      {
       parametro[0]='E';
       for (i=0;i<10;i++)
       {
       parametro[1]=48+i;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_entradas_plc[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
       parametro[1]='1';
       for (i=10;i<16;i++)
       {
       parametro[2]=48+i-10;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_entradas_plc[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"SALIDA"))
      {
       parametro[0]='S';
       for (i=0;i<10;i++)
       {
       parametro[1]=48+i;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_salidas_plc[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
       parametro[1]='1';
       for (i=10;i<20;i++)
       {
       parametro[2]=48+i-10;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_salidas_plc[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
       parametro[1]='2';
       for (i=20;i<24;i++)
       {
       parametro[2]=48+i-20;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_salidas_plc[i]=obj_def_ptr->obj_index;
          goto bien;
          }
```



```
       }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"MEDIDA"))
      {
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"ENTRADAS"))
          {
          indice.m_plc[0]=obj_def_ptr->obj_index;
          goto bien;
          }
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,"SALIDAS"))
          {
          indice.m_plc[1]=obj_def_ptr->obj_index;
          goto bien;
          }

      }
    }

/* VARIOS PLANTA*/

memset(parametro,0,sizeof(parametro));

if(!strcmp(obj_def_ptr->param_1.p_value.symbol_p,"PLANTA"))
   {
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"MEDIDA_CAUDAL"))
      {
       parametro[0]='C';
       for (i=0;i<3;i++)
       {
       parametro[1]=48+i;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_caudales[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"SP_CAUDAL"))
      {
       parametro[0]='C';
       for (i=0;i<3;i++)
       {
       parametro[1]=48+i;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.c_caudales[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"ESTADOS_VARIOS"))
      {
       parametro[0]='E';
       for (i=0;i<10;i++)
       {
       parametro[1]=48+i;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_estados_varios[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
       parametro[1]='1';
       for (i=10;i<20;i++)
       {
       parametro[2]=48+i-10;
       if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
          {
          indice.m_estados_varios[i]=obj_def_ptr->obj_index;
          goto bien;
          }
       }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"MEDIDA_TIEMPOS"))
      {
       parametro[0]='T';
       for (i=0;i<10;i++)
```

```
        {
        parametro[1]=48+i;
        if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
           {
           indice.m_tiempos[i]=obj_def_ptr->obj_index;
           goto bien;
           }
        }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"SP_TIEMPOS"))
      {
        parametro[0]='T';
        for (i=0;i<10;i++)
        {
        parametro[1]=48+i;
        if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
           {
           indice.c_tiempos[i]=obj_def_ptr->obj_index;
           goto bien;
           }
        }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"VARIABLE"))
      {
        parametro[0]='V';
        for (i=0;i<10;i++)
        {
        parametro[1]=48+i;
        if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
           {
           indice.m_variable[i]=obj_def_ptr->obj_index;
           goto bien;
           }
        }
        parametro[1]='1';
        for (i=10;i<20;i++)
        {
        parametro[2]=48+i-10;
        if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
           {
           indice.m_variable[i]=obj_def_ptr->obj_index;
           goto bien;
           }
        }
      }
   if(!strcmp(obj_def_ptr->param_2.p_value.symbol_p,"SP_VARIABLE"))
      {
        parametro[0]='V';
        for (i=0;i<10;i++)
        {
        parametro[1]=48+i;
        if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
           {
           indice.c_variable[i]=obj_def_ptr->obj_index;
           goto bien;
           }
        }
        parametro[1]='1';
        for (i=10;i<20;i++)
        {
        parametro[2]=48+i-10;
        if(!strcmp(obj_def_ptr->param_3.p_value.symbol_p,parametro))
           {
           indice.c_variable[i]=obj_def_ptr->obj_index;
           goto bien;
           }
        }
      }
    }


#ifdef DEBUG
    printf ("gsi_def_obj FAIL in context %d\n",current_context);
#endif
return(GSI_FAIL);
```



```
bien:

#ifdef DEBUG
printf ("gsi_def_obj in context %d\n",current_context);
printf (" %s, a %s registered with index %d\n",name_of(obj_def_ptr),
                        class_of(obj_def_ptr), obj_def_ptr->obj_index);
#endif

return (GSI_SUCCESS);
}

/*************************************************************************/
/* GSI proporciona una lista de variables de las que se requieren valores.
   La funciona toma los valores del sistema externo y los devuelve de la
   forma en que GSI los entiende */

gsi_int gsi_get_data(obj_list, count)
    obj_type *obj_list;
    gsi_int   count;
{
TIME_ARG_TYPE argumentos[25];
struct tm *mihora1, *mihora2;
int i,j;
time_t mihora;
long int  chequeo=0;

memset(argumentos,0,sizeof(argumentos));

sem_wait(semid1);     /* Esperando el semáforo 1 */
memcpy(&medidas,direccion1,sizeof(medidas));
for (i=0;i<200;i++) chequeo+=direccion1[i];
memcpy(&var, &direccion1[700], sizeof(var));
sem_signal(semid1);   /* Deja el semáforo 1*/

if (chequeo==0) return(GSI_FAIL);

chequeo=0;
sem_wait(semid3);     /* Esperando el semáforo 3 */
memcpy(analisis,direccion3,sizeof(analisis));
for (i=0;i<200;i++) chequeo+=direccion3[i];
sem_signal(semid3);   /* Deja el semáforo 3*/


for (i=0;i<count;i++)
    {
       mihora=medidas.tiempo;

       mihora1=localtime( &mihora );
       argumentos[i].obj_index=obj_list[i].obj_index;
       argumentos[i].value.p_type=FLOAT64_TAG;
       argumentos[i].date.year=(icp_int)(1900L+mihora1->tm_year);
       argumentos[i].date.month=(icp_int)(1L+mihora1->tm_mon);
       argumentos[i].date.day=(icp_int)(mihora1->tm_mday);
       argumentos[i].date.hour=(icp_int)(mihora1->tm_hour);
       argumentos[i].date.minute=(icp_int)(mihora1->tm_min);
       argumentos[i].date.second=(icp_int)(mihora1->tm_sec);

    for (j=0;j<3;j++)
       {
       if (obj_list[i].obj_index==indice.m_ph[j])
           {
           argumentos[i].value.p_value.float_p=medidas.ph[j];
           goto bien;
           }
       if (obj_list[i].obj_index==indice.m_temperatura[j])
           {
           argumentos[i].value.p_value.float_p=medidas.temperatura[j];
           goto bien;
           }
       if (obj_list[i].obj_index==indice.m_potencial[j])
           {
           argumentos[i].value.p_value.float_p=medidas.potencial[j];
           goto bien;
           }
       if (obj_list[i].obj_index==indice.m_oxigeno[j])
           {
           argumentos[i].value.p_value.float_p=medidas.oxigeno[j];
```

```
        goto bien;
        }
    if (obj_list[i].obj_index==indice.m_mA[j])
        {
        argumentos[i].value.p_value.float_p=medidas.mA[j];
        goto bien;
        }
    if (obj_list[i].obj_index==indice.m_velocidad[j])
        {
        argumentos[i].value.p_value.float_p=medidas.velocidad[j];
        goto bien;
        }
    if (obj_list[i].obj_index==indice.m_sp_oxigeno[j])
        {
        argumentos[i].value.p_value.float_p=medidas.set_point_oxigeno[j];
        goto bien;
        }
    if (obj_list[i].obj_index==indice.m_sp_velocidad[j])
        {
        argumentos[i].value.p_value.float_p=medidas.set_point_velocidad[j];
        goto bien;
        }
    if (obj_list[i].obj_index==indice.m_caudales[j])
        {
        argumentos[i].value.p_value.float_p=medidas.caudales[j];
        goto bien;
        }
    }

for (j=0;j<16;j++)
    {
    if (obj_list[i].obj_index==indice.m_entradas_plc[j])
        {
        argumentos[i].value.p_type=INTEGER_TAG;
        if (medidas.entradas_plc[j]==49)
            argumentos[i].value.p_value.integer_p=1;
        else argumentos[i].value.p_value.integer_p=0;
        goto bien;
        }
    }
for (j=0;j<24;j++)
    {
    if (obj_list[i].obj_index==indice.m_salidas_plc[j])
        {
        argumentos[i].value.p_type=INTEGER_TAG;
        if (medidas.salidas_plc[j]==49)
            argumentos[i].value.p_value.integer_p=1;
        else argumentos[i].value.p_value.integer_p=0;
        goto bien;
        }
    }

if (obj_list[i].obj_index==indice.m_plc[0])
        {
        int k;
        int valor=0;
        argumentos[i].value.p_type=INTEGER_TAG;
        for (k=0;k<16;k++)
            {
            if (medidas.entradas_plc[k]==49)
                    valor=valor | (1L << k);
            }
        argumentos[i].value.p_value.integer_p=valor;
        goto bien;
        }

if (obj_list[i].obj_index==indice.m_plc[1])
        {
        int k;
        int valor=0;
        argumentos[i].value.p_type=INTEGER_TAG;
        for (k=0;k<24;k++)
            {
            if (medidas.salidas_plc[k]==49)
                    valor=valor | (1L << k);
            }
        argumentos[i].value.p_value.integer_p=valor;
```



```
        goto bien;
        }

    for (j=0;j<20;j++)
        {
        if (obj_list[i].obj_index==indice.m_estados_varios[j])
            {
            argumentos[i].value.p_type=INTEGER_TAG;
            argumentos[i].value.p_value.integer_p=
                medidas.estados_varios[j]-48;
            goto bien;
            }
        }

    for (j=0;j<10;j++)
        {
        if (obj_list[i].obj_index==indice.m_tiempos[j])
            {
            argumentos[i].value.p_type=INTEGER_TAG;
            argumentos[i].value.p_value.integer_p=medidas.tiempos[j];
            goto bien;
            }
        }

    for (j=0;j<20;j++)
        {
        if (obj_list[i].obj_index==indice.m_variable[j])
            {
            if (var.t_variable[j]==0) /* No existe medida */
                {
                argumentos[i].error_cd=10;
                argumentos[i].value.p_type=NULL_TAG;
                goto bien;
                }
            argumentos[i].value.p_value.float_p=var.variable[j];
            mihora2=localtime( &(var.t_variable[j]) );
            argumentos[i].value.p_type=FLOAT64_TAG;
            argumentos[i].date.year=(icp_int)(1900L+mihora2->tm_year);
            argumentos[i].date.month=(icp_int)(1L+mihora2->tm_mon);
            argumentos[i].date.day=(icp_int)(mihora2->tm_mday);
            argumentos[i].date.hour=(icp_int)(mihora2->tm_hour);
            argumentos[i].date.minute=(icp_int)(mihora2->tm_min);
            argumentos[i].date.second=(icp_int)(mihora2->tm_sec);
            goto bien;
            }
        }


    for (j=0;j<5;j++)
        {
        if (obj_list[i].obj_index==indice.m_nitratos[j])
            {
            if (analisis[j].t_nitratos==0) /* No existe medida */
                {
                argumentos[i].error_cd=10;
                argumentos[i].value.p_type=NULL_TAG;
                goto bien;
                }
            argumentos[i].value.p_value.float_p=analisis[j].nitratos;
            mihora2=localtime( &(analisis[j].t_nitratos) );
            argumentos[i].value.p_type=FLOAT64_TAG;
            argumentos[i].date.year=(icp_int)(1900L+mihora2->tm_year);
            argumentos[i].date.month=(icp_int)(1L+mihora2->tm_mon);
            argumentos[i].date.day=(icp_int)(mihora2->tm_mday);
            argumentos[i].date.hour=(icp_int)(mihora2->tm_hour);
            argumentos[i].date.minute=(icp_int)(mihora2->tm_min);
            argumentos[i].date.second=(icp_int)(mihora2->tm_sec);
            goto bien;
            }
        if (obj_list[i].obj_index==indice.m_nitritos[j])
            {
            if (analisis[j].t_nitritos==0) /* No existe medida */
                {
                argumentos[i].error_cd=10;
                argumentos[i].value.p_type=NULL_TAG;
                goto bien;
                }
```

```
            argumentos[i].value.p_value.float_p=analisis[j].nitritos;
            mihora2=localtime( &(analisis[j].t_nitritos) );
            argumentos[i].value.p_type=FLOAT64_TAG;
            argumentos[i].date.year=(icp_int)(1900L+mihora2->tm_year);
            argumentos[i].date.month=(icp_int)(1L+mihora2->tm_mon);
            argumentos[i].date.day=(icp_int)(mihora2->tm_mday);
            argumentos[i].date.hour=(icp_int)(mihora2->tm_hour);
            argumentos[i].date.minute=(icp_int)(mihora2->tm_min);
            argumentos[i].date.second=(icp_int)(mihora2->tm_sec);
            goto bien;
            }
        if (obj_list[i].obj_index==indice.m_amonio[j])
            {
            if (analisis[j].t_amonio==0) /* No existe medida */
                {
                argumentos[i].error_cd=10;
                argumentos[i].value.p_type=NULL_TAG;
                goto bien;
                }
            argumentos[i].value.p_value.float_p=analisis[j].amonio;
            mihora2=localtime( &(analisis[j].t_amonio) );
            argumentos[i].value.p_type=FLOAT64_TAG;
            argumentos[i].date.year=(icp_int)(1900L+mihora2->tm_year);
            argumentos[i].date.month=(icp_int)(1L+mihora2->tm_mon);
            argumentos[i].date.day=(icp_int)(mihora2->tm_mday);
            argumentos[i].date.hour=(icp_int)(mihora2->tm_hour);
            argumentos[i].date.minute=(icp_int)(mihora2->tm_min);
            argumentos[i].date.second=(icp_int)(mihora2->tm_sec);
            goto bien;
            }
        }

#ifdef DEBUG
    printf ("gsi_get_data FAIL in context %d\n",current_context);
#endif
return(GSI_FAIL);


bien:

#ifdef DEBUG

#endif
    }


#ifdef DEBUG
 for (i=0;i<count;i++)
    printf("Obj: %i Hora: %2i %2i %2i  Valor: %f\n",
        argumentos[i].obj_index, argumentos[i].date.hour,
       argumentos[i].date.minute, argumentos[i].date.second,
        argumentos[i].value.p_value.float_p);

    printf ("gsi_get_data in context %d\n",current_context);
#endif


gsi_ret_tvals(argumentos,count);
return (GSI_SUCCESS);
}

/*************************************************************************/
/* GSI proporciona una lista de consignas. La función debe comunicarse con
   el sistema externo y establecer esos valores */

gsi_int gsi_set_data(obj_list, count)
    obj_type *obj_list;
    gsi_int   count;
{
int i,j;

sem_wait(semid2);     /* Esperando el semáforo 2 */
memcpy(&consignas,direccion2,sizeof(consignas));
sem_signal(semid2);   /* Deja el semáforo 2*/

for (i=0;i<count;i++)
    {
```



```
    for (j=0;j<3;j++)
        {

        if (obj_list[i].obj_index==indice.c_oxigeno[j])
            {
            consignas.cambio_oxigeno[j]='1';

            if (obj_list[i].value.p_type==FLOAT64_TAG)
                consignas.valor_oxigeno[j]=
                        obj_list[i].value.p_value.float_p;
            if (obj_list[i].value.p_type==INTEGER_TAG)
                consignas.valor_oxigeno[j]=
                        (double)obj_list[i].value.p_value.integer_p;
            goto bien;
            }
        if (obj_list[i].obj_index==indice.c_velocidad[j])
            {
            consignas.cambio_agitacion[j]='1';

            if (obj_list[i].value.p_type==FLOAT64_TAG)
                consignas.valor_agitacion[j]=
                        obj_list[i].value.p_value.float_p;
            if (obj_list[i].value.p_type==INTEGER_TAG)
                consignas.valor_agitacion[j]=
                        (double)obj_list[i].value.p_value.integer_p;
            goto bien;
            }
        if (obj_list[i].obj_index==indice.c_caudales[j])
            {
            consignas.cambio_caudales[j]='1';

            if (obj_list[i].value.p_type==FLOAT64_TAG)
                consignas.valor_caudales[j]=
                        obj_list[i].value.p_value.float_p;
            if (obj_list[i].value.p_type==INTEGER_TAG)
                consignas.valor_caudales[j]=
                        (double)obj_list[i].value.p_value.integer_p;
            goto bien;
            }
        }
    for (j=0;j<10;j++)
        {

        if (obj_list[i].obj_index==indice.c_tiempos[j])
            {
            consignas.cambio_tiempos[j]='1';

            if (obj_list[i].value.p_type==FLOAT64_TAG)
                consignas.tiempos[j]=(int)obj_list[i].value.p_value.float_p;
            if (obj_list[i].value.p_type==INTEGER_TAG)
                consignas.tiempos[j]=obj_list[i].value.p_value.integer_p;
            goto bien;
            }
        }
    for (j=0;j<20;j++)
        {

        if (obj_list[i].obj_index==indice.c_variable[j])
            {
            consignas.cambio_variable[j]='1';

            if (obj_list[i].value.p_type==FLOAT64_TAG)
                consignas.valor_variable[j]=obj_list[i].value.p_value.float_p;
            if (obj_list[i].value.p_type==INTEGER_TAG)
                consignas.valor_variable[j]=
                (double)obj_list[i].value.p_value.integer_p;
            goto bien;
            }
        }
bien:
    }


sem_wait(semid2);     /* Esperando el semáforo 2 */
memcpy(direccion2,&consignas,sizeof(consignas));
sem_signal(semid2);   /* Deja el semáforo 2*/
```

```
#ifdef DEBUG
    printf ("gsi_set_data in context %d\n",current_context);
#endif

    return (GSI_SUCCESS);
}

/*************************************************************************/
/* Función que ejecuta GSI para que el sistema externo deje de obtener datos
   para enviárselos a G2 */

gsi_int gsi_pause_context()
{
#ifdef DEBUG
    printf ("gsi_pause_context in context %d\n",current_context);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* Función que ejecuta GSI para que el sistema externo vuelva a obtener datos
   para enviárselos a G2 */

gsi_int gsi_resume_context()
{
#ifdef DEBUG
    printf ("gsi_resume_context in context %d\n",current_context);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* Función que GSI utiliza para que el sistema externo deje de adquirir datos
   y para realizar las acciones necesarias para finalizar este programa */

gsi_int gsi_shutdown_context()
{
if (shmdt(direccion1) < 0)
    err_sys("gsi_jb: No puedo soltar la memoria compartida 1\n");
if (shmdt(direccion2) < 0)
    err_sys("gsi_jb: No puedo soltar la memoria compartida 2\n");
if (shmdt(direccion3) < 0)
    err_sys("Migsi: No puedo soltar la memoria compartida 3\n");

sem_close(semid1);
sem_close(semid2);
sem_close(semid3);

#ifdef DEBUG
    printf ("gsi_shutdown_context in context %d\n",current_context);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* Función llamada anteriormente "gsi_accept_data()".
   Es utilizada por GSI para mandar a G2 datos no solicitados. */

gsi_int gsi_g2_poll()
{
char mensaje_cliente[TAMANY_M];
int i=0;

sem_wait(semid1);     /* Espera el semáforo 1 */
memcpy( mensaje_cliente, &direccion1[500], sizeof(mensaje_cliente));
sem_signal(semid1);   /* Deja el semáforo 1 */

if (mensaje_cliente[0]!=0)
      {
      while(mensaje_cliente[i]!=10 && i<(TAMANY_M-2)) i++;
      mensaje_cliente[i+1]=0;
      gsi_ret_msg(mensaje_cliente, sizeof(mensaje_cliente));
      sem_wait(semid1);     /* Espera el semáforo 1 */
      memset(&direccion1[500], 0, sizeof(mensaje_cliente));
      sem_signal(semid1);   /* Deja el semáforo 1 */
      }
```



```
#ifdef DEBUG
    printf ("\ngsi_g2_poll in context %d\n",current_context);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* GSI utiliza esta función para que el sistema externo deje de tomar datos
   de una variable, cuando cambia sus características o deja de existir */

gsi_int gsi_stop_data(obj_list, count)
    obj_type *obj_list;
    gsi_int   count;
{
#ifdef DEBUG
    printf ("gsi_stop_data in context %d\n",current_context);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* Función llamada anteriormente "gsi_rd_txt()".
   Cuando G2 manda un mensaje a GSI, esta función debe encargarse de
   redirigirlo al sistema externo */

gsi_int gsi_send_message(message, length, obj_index)
    char     *message;
    gsi_int   length;
    gsi_int   obj_index;
{
#ifdef DEBUG
    printf ("gsi_send_message in context %d\n",current_context);
    printf ("G2 says:%s\n",message);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* GSI llama 50 veces por segundo a esta función para realizar tareas si
   *DO_EXT_PROC es TRUE */

gsi_int allow_ext_proc()
{
#ifdef DEBUG
    printf ("allow_ext_proc in context %d\n",current_context);
#endif
    return (GSI_SUCCESS);
}

/*************************************************************************/
/* Función que registra las funciones RPC que G2 puede pedir a GSI */

gsi_int InitUsrRpcs()
{
    return(GSI_SUCCESS);
}

/*************************************************************************/
/* rpc definido para lanzar una alarma en la SUN */

gsi_int alarma(arguments, count, call_index)
   rpc_arg_type *arguments;
   gsi_int count, call_index;
{
#ifdef DEBUG
    printf ("rpc alarma in context %d\n",current_context);
#endif

system("amp /quantum/juan/mp3/Van* &");

    return(GSI_SUCCESS);
}
```

```
/*****************************************************************************
:
:  gsi_main.c  --  Sample main function to link with GSI toolkit.
:
:  This is a functional main which is used to link with skeleton.c and other
:  GSI example programs.  Feel free to use this main as is, or modified to
:  suit your needs.
:
:  Although this software has been extensively tested, Gensym cannot
:  guarantee error-free performance in all applications.  Accordingly,
:  use of the software is at the customer's sole risk.
:
:                                                                    29jul93
:.............................................................................*/

#include "gsi.h"  /* Note, icp.h is included by gsi.h. */

extern gsi_int alarma();

main(argc, argv)
    int   argc;
    char *argv[];
{
   gsi_int status;

/*
 *  Specify GSI run-time options, if any...
 *
 *  Unless specified otherwise, GSI uses D floats under VAX/VMS, and
 *  IEEE floats under Alpha/OpenVMS.  In both VMS systems, G floats may
 *  be specified as a run-time option.  In the code below the selection
 *  of floating point format is automated for VMS platforms via the use
 *  of the symbol CC$gfloat.  CC$gfloat is set on VAXen when this file
 *  has been compiled with the /G_FLOAT option, and is set by default on
 *  Alphas (we have chosen IEEE as the default in all GSI example code).
 *
 *  Although off by default, GSI_ONE_CYCLE is explicitly reset to
 *  demonstrate how options are turned off.
 */
#   ifdef vms
#       if CC$gfloat
            gsi_set_option(GSI_USE_GFLOATS);
#       endif
#   endif
    gsi_reset_option(GSI_ONE_CYCLE);

   /* gsi_reset_option(ICP_USE_RPCS);   /* Por ahora no hay RPCs */
  /*  gsi_set_option(ICP_RPC_ARRARGS);  /* Nuevo modo G2-GSI RPC */

/*
 *  Set the version control variables to the values defined in gsi.h
 *  Though optional, this is recommended to ensure version consistency
 *  between the GSI object libraries, and the GSI and ICP include
 *  files used to compile your application code.
 */
    gsi_include_file_major_version = GSI_INCLUDE_MAJ_VER_NUM;
    gsi_include_file_minor_version = GSI_INCLUDE_MIN_VER_NUM;
    icp_include_file_major_version = ICP_INCLUDE_MAJ_VER_NUM;
    icp_include_file_minor_version = ICP_INCLUDE_MIN_VER_NUM;

/* Registro de las RPC declaradas en gsi_jb */

    if ( gsi_rpc_declare_local( alarma, "ALARMA")!= GSI_SUCCESS)
        exit(1);

/*
 *  Initialize GSI and enter the event handler loop.
 */
    status = gsi_start(argc, argv);

/*
 *  Since the GSI_ONE_CYCLE option is not set, gsi_start() never returns.
 *  If GSI_ONE_CYCLE were set, gsi_start() would return after setup
 *  and one iteration of the GSI event loop.  When using the GSI_ONE_CYCLE
 *  option, subsequent calls to gsi_run_loop() are required after the initial
 *  call to gsi_start() in order to allow GSI to react to messages from
 *  G2 and to make calls to the user code funcitons.  An example implementation
```



```
 *  is provided below (commented out).  The function scan_and_react() is a
 *  hypothetical function that interacts with some other process and GSI.
 */
/*  do {
        status = gsi_run_loop();
        status = scan_and_react(status);
    } while (status == GSI_SUCCESS);
 */
    return 0;

} /* end main */
```

```
/* Miscellaneous macros
------------------------*/
#ifndef max
#define max(A, B) ((A) > (B) ? (A) : (B))
#endif

#ifndef min
#define min(A, B) ((A) > (B) ? (B) : (A))
#endif


/* Most of the following typedefs and defines support
the old naming conventions for the data structures defined
above.  Some provide aliases for multi-purpose structures
-------------------------------------------------------------*/
typedef obj_def_type        USER_ARG_TYPE;
typedef obj_def_type       *obj_def_ptr_type;
typedef obj_value_type      ICP_PARAM;
typedef obj_value_type      array_value_type;
typedef icp_value_union     ICP_VALUE;
typedef obj_type            TEMP_ARG_TYPE;
typedef obj_value_type      TEMP_PARAM;
typedef obj_value_type      attr_value_type;
typedef obj_value_type      rpc_arg_type;
typedef icp_value_union     TEMP_VALUE;
typedef timed_obj_type      TIME_ARG_TYPE;
typedef attr_type           VECTOR_TYPE;
typedef timed_attr_type     T_VECTOR_TYPE;
typedef icp_timestamp_type  TIMESTAMP;

typedef void               *user_def_type;
typedef void               *user_ptr_type;
typedef icp_int                  obj_index_type,
                           tag_type,
                           status_type,
                           interval_type;
```



```
extern void                gsi_set_attr();
extern void                gsi_set_obj_index();
extern void                gsi_set_status();
extern void                gsi_set_attr_name();
extern void                gsi_set_time();
extern void                gsi_set_attr_time();
extern void                gsi_set_is_timed();
extern void                gsi_set_type();
extern void                gsi_set_int();
extern void                gsi_set_log();
extern void                gsi_set_flt();
extern void                gsi_set_sym();
extern void                gsi_set_str();
extern boolean             gsi_context_received_data();

/* The following defines allow you to use the
old names for certain variables and functions.
-------------------------------------------------*/
#define CurrentContext              icp_current_context
#define current_context             icp_current_context
#define flush_point                 icp_flush_point
#define DO_EXT_PROC                 do_ext_proc
#define SetIcpMode(x)               (icp_options = icp_options | x)
#define ResetIcpMode(x)             (icp_options = icp_options & (-x))
#define TestIcpMode(x)              (icp_options & x)
#define ENABLE_EXT_PROC()           *do_ext_proc = TRUE
#define CORR_OBJ_TYPE               corr_obj_type
#define str_ptr_of(x)               gsi_str_of(x)
#define sym_ptr_of(x)               gsi_sym_of(x)
#define gsi_user_def_of(x,y)        gsi_user_ptr_of(x,y)
#define gsi_set_user_def(x,y,z)     gsi_set_user_ptr(x,y,z)

/* The following defines allow shorter (without 'gsi_' prefix) names
for access functions to be used.  Prefixes were added to protect
against name collisions when linking with third party products.
-----------------------------------------------------------------*/
#define type_of(x)          gsi_type_of(x)
#define name_of(x)          gsi_name_of(x)
```



```
/*****************************************************************************

File name:              gsi.h
Owner:                  Peter Fandel

Purpose:                This include file serves to define the various
                        structure types and contants required by GSI user
                        applications.

This file depends on:   icp.h

Dependent files:        All GSI modules and their end-user applications.

Legal Notice:           Copyright (C) 1993, Gensym Corp.,
                        125 CambridgePark Drive, Cambridge MA 02140
                        All rights reserved.

History:                18feb93  PAF  Created
                        01feb95  PAF  Simplified version numbering system.
                       23mar95  PAF  Added alias for icp_timestamp_type

*****************************************************************************/

#include "icp.h"

/* These declarations are used for optional version control.
-------------------------------------------------------------*/
#define GSI_INCLUDE_MAJ_VER_NUM        3L
#define GSI_INCLUDE_MIN_VER_NUM        2L

extern gsi_int gsi_include_file_major_version;
extern gsi_int gsi_include_file_minor_version;


/* Maximum size for a local function name being exported to G2.
-------------------------------------------------------------*/
#define MAX_LOCAL_FUNC_NAME    128L
```



```
#define LoadUsrRpc(x,y,z)          gsi_rpc_declare_local(x,y)
#define g2_start_rpc(x,y,z)        gsi_rpc_start(x,y,gsi_current_context)
#define g2_call_rpc(x,y,z)         gsi_rpc_call(x,y,gsi_current_context)
#define getgsiport()               gsi_get_tcp_port()
#define gsi_init_ds(x,y)           gsi_initialize_context(x,y)
#define gsi_pause_ds()             gsi_pause_context()
#define gsi_resume_ds()            gsi_resume_context()
#define gsi_shutdown()             gsi_shutdown_context()
#define gsi_accept_data()          gsi_g2_poll()
#define gsi_rd_txt(x,y,z)          gsi_send_message(x,y,z)
#define GSI_FAIL                   GSI_FAILURE
#define GSI_PORT_NUM               0L
```

```
typedef icp_int gspan_int;

/* These declarations are used for optional version control.
--------------------------------------------------------------*/
#define ICP_INCLUDE_MAJ_VER_NUM        3L
#define ICP_INCLUDE_MIN_VER_NUM        2L

extern icp_int icp_include_file_major_version;
extern icp_int icp_include_file_minor_version;

#define NULL_TAG              0L
#define INTEGER_TAG         0x01L
#define SYMBOL_TAG          0x03L
#define STRING_TAG          0x04L
#define LOGICAL_TAG           0x05L
#define FLOAT64_TAG           0x06L
#define ITEM_TAG              0x08L
#define MAX_TYPE_TAG          0x08L

#define NO_ERR                  0L
#define MAX_ATTR_NAME_SIZE    256L

#define ICP_TRUE             1000L
#define ICP_FALSE           -1000L

#define MAX_G2_INTEGER   536870911L
#define MIN_G2_INTEGER  -536870912L

#define VOID_INDEX            -1L

#define ICP_PORT_NUM        22041L

#define UNDEFINED_CONTEXT   0xFFFFFFFFL
#ifdef MSDOS
#    define MAX_CONTEXTS        10L
#else
#    define MAX_CONTEXTS        50L
#endif
```



```
    obj_value_type  param_5;
    obj_value_type  param_6;
    icp_int         interval;
    obj_value_type  value;
} obj_def_type;

typedef struct {
    obj_def_type *obj_ptr;
    union {
        icp_int  user_long;
        void    *user_ptr;
    } user_def;
    icp_int  tag;
} corr_obj_type;

typedef struct {
    icp_int        obj_index;
    icp_int        error_cd;
    icp_int        interval;
    obj_value_type value;
} obj_type;

typedef struct {
    icp_int year;
    icp_int month;
    icp_int day;
    icp_int hour;
    icp_int minute;
    icp_int second;
} icp_timestamp_type;

typedef struct at {
    char           attribute[MAX_ATTR_NAME_SIZE];
    obj_value_type value;
} attr_type;

typedef struct {
    char           attribute[MAX_ATTR_NAME_SIZE];
    obj_value_type value;
```



```
#define class_of(x)          gsi_class_of(x)
#define length_of(x)         gsi_length_of(x)
#define set_type(x,y)        gsi_set_type(x,y)
#define int_of(x)            gsi_int_of(x)
#define log_of(x)            gsi_log_of(x)
#define flt_of(x)            gsi_flt_of(x)
#define sym_of(x)            gsi_sym_of(x)
#define str_of(x)            gsi_str_of(x)
#define set_int(x,y)         gsi_set_int(x,y)
#define set_log(x,y)         gsi_set_log(x,y)
#define set_flt(x,y)         gsi_set_flt(x,y)
#define set_sym(x,y)         gsi_set_sym(x,y)
#define set_str(x,y)         gsi_set_str(x,y)
#define obj_index_of(x)      gsi_obj_index_of(x)
#define set_obj_index(x,y)   gsi_set_obj_index(x,y)
#define status_of(x)         gsi_status_of(x)
#define set_status(x,y)      gsi_set_status(x,y)
#define interval_of(x)       gsi_interval_of(x)
#define set_interval(x,y)    gsi_set_interval(x,y)
#define def_attr1_of(x)      gsi_def_attr1_of(x)
#define def_attr2_of(x)      gsi_def_attr2_of(x)
#define def_attr3_of(x)      gsi_def_attr3_of(x)
#define def_attr4_of(x)      gsi_def_attr4_of(x)
#define def_attr5_of(x)      gsi_def_attr5_of(x)
#define def_attr6_of(x)      gsi_def_attr6_of(x)
#define obj_of(x)            gsi_obj_of(x)
#define value_of(x)          gsi_value_of(x)
#define set_value(x,y)       gsi_set_value(x,y)
#define attr_of(x)           gsi_attr_of(x)
#define set_attr(x,y)        gsi_set_attr(x,y)
#define attr_name_of(x)      gsi_attr_name_of(x)
#define set_attr_name(x,y)   gsi_set_attr_name(x,y)
#define time_of(x)           gsi_time_of(x)
#define set_time(x,y)        gsi_set_time(x,y)
#define attr_time_of(x)      gsi_attr_time_of(x)
#define set_attr_time(x,y)   gsi_set_attr_time(x,y)
#define set_is_timed(x,y)    gsi_set_is_timed(x,y)
```



```
/*****************************************************************************

   SEMAFORO.C - FUNCIONES PARA LA UTILIZACION DE SEMAFOROS

   PROGRAMADOR:  Juan Antonio Baeza Labat
   ULTIMA REVISION: 5/6/97

*****************************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <errno.h>


/*#define SEMKEY ((key_t) 4223353L)*/
#define PERMS 0644
#define err_sys(A) printf (A "\n")
#define BIGCOUNT 10000

/*************************************************/
/* Declaraciones de las funciones */

int sem_create( key_t key, int initval);
void sem_rm(int id);
void sem_close(int id);
void sem_wait(int id);
void sem_signal(int id);
void sem_op(int id, int value);



/*************************************************/
int errno;

/* Definiciones de los conjuntos de instrucciones para
```

```
int sem_create( key_t key, int initval)
{
register int  id, semval;

union semun
{
int             val;
struct semid_ds *buf;
ushort          *array;
} semctl_arg;

   if (key == IPC_PRIVATE) return (-1);
   else if (key == (key_t) -1) return (-1);

again:
   if ( (id = semget(key, 3, PERMS | IPC_CREAT)) < 0)
      return (-1);
   if (semop(id, &op_lock[0], 2) < 0)
      {
      if (errno == EINVAL) goto again;
      err_sys("can't lock");
      }

   if ( (semval = semctl(id, 1, GETVAL, 0)) < 0)
      err_sys("can't GETVAL");
   if (semval == 0)
      {
      semctl_arg.val = initval;
      if (semctl(id, 0, SETVAL, semctl_arg) < 0)
         err_sys("can SETVAL[0]");

      semctl_arg.val = BIGCOUNT;
      if (semctl(id, 1, SETVAL, semctl_arg) < 0)
         err_sys("can SETVAL[1]");
      }
    if (semop(id, &op_endcreate[0], 2) < 0)
         err_sys("can't end create");

    return(id);
}

/*******************************************************/
/* Abre un semáforo que ya existe.
   Devuelve la identidad del semáforo.   */

int sem_open (key_t key)
{
register int  id;

   if (key == IPC_PRIVATE) return (-1);
   else if (key == (key_t) -1) return (-1);

   if ( (id = semget(key, 3, 0)) < 0)
     return (-1);

   if (semop(id, &op_open[0], 1) < 0)
     err_sys("can't open");

   return (id);
}

/*******************************************************/
/* Elimina un semáforo.
   Debe ser llamada por el servidor. */

void sem_rm(int id)
{

  if (semctl(id, 0 ,IPC_RMID, 0) < 0)
      err_sys("can't IPC_RMID");
}

/*******************************************************/
/* Cierra un semáforo. */

void sem_close(int id)
```



```
{
register int  semval;

  if (semop(id, &op_close[0], 3) < 0)
       err_sys("can't semop");

  if (semval=semctl(id, 1, GETVAL, 0) < 0)
       err_sys("can't GETVAL");
  if (semval > BIGCOUNT)
       err_sys("sem[1] > BIGCOUNT");
  else if (semval == BIGCOUNT)
       sem_rm(id);
  else
       if (semop(id, &op_unlock[0], 1) < 0)
          err_sys("can't unlock");
}

/*******************************************************/
/* Espera hasta que el semáforo es mayor que cero.
   Después le resta 1 y vuelve */

void sem_wait(int id)
{
  sem_op(id,-1);
}

/*******************************************************/
/* Incrementa un semáforo en 1 */

void sem_signal(int id)
{
  sem_op(id,1);
}

/*******************************************************/
/* Operación general sobre un semáforo.  */

void sem_op(int id, int value)
{
  if ( (op_op[0].sem_op = value) == 0)
       err_sys("can't have value == 0");

  if (semop(id, &op_op[0], 1) < 0)
       err_sys("sem_op error");
}

/*******************************************************/
```

6 SISTEMA EXPERTO

```
/************************************************************************/

      JERARQUÍA DE OBJETOS EN EL SISTEMA EXPERTO DESARROLLADO

/************************************************************************/


** Gensym G2 Knowledge Base Inspection Output
** From KB:     /usr/users/juan/super/kbtemporal.kb
** File:        /quantum/juan/bmp/hierarchy.qp
** Written at: 16 Jun 99  2:50:56 p.m.


** Command:

write to the file "/quantum/juan/bmp/hierarchy"
   the class hierarchy

** Results follow this line:

ITEM
|  KB-WORKSPACE -- 242 instances
|  SYSTEM-TABLE
|  |  DEBUGGING-PARAMETERS -- 9 instances
|  |  PRINTER-SETUP -- 9 instances
|  |  FONTS -- 9 instances
|  |  DRAWING-PARAMETERS -- 9 instances
|  |  TIMING-PARAMETERS -- 9 instances
|  |  LOGBOOK-PARAMETERS -- 9 instances
|  |  MESSAGE-BOARD-PARAMETERS -- 9 instances
|  |  LOG-FILE-PARAMETERS -- 9 instances
|  |  MENU-PARAMETERS -- 9 instances
|  |  LANGUAGE-PARAMETERS -- 9 instances
|  |  COLOR-PARAMETERS -- 9 instances
|  |  EDITOR-PARAMETERS -- 9 instances
|  |  MODULE-INFORMATION -- 9 instances
|  |  SAVING-PARAMETERS -- 9 instances
|  |  INFERENCE-ENGINE-PARAMETERS -- 9 instances
|  |  DATA-SERVER-PARAMETERS -- 9 instances
|  |  SIMULATION-PARAMETERS -- 9 instances
|  |  KB-RESTRICTIONS
|  |  |  KB-CONFIGURATION -- 9 instances
|  |  MISCELLANEOUS-PARAMETERS -- 9 instances
|  CONNECTION
|  |  NETWORK-WIRE
|  |  |  TCP-IP
|  |  DIGITAL
|  |  |  SERIAL
|  |  |  |  RS232
|  |  |  |  RS485
|  |  ELECTRICAL
|  |  |  PROBE-SIGNAL
|  |  |  ON-OFF -- 8 instances
|  |  |  |  220V -- 10 instances
|  |  |  |  24V -- 26 instances
|  |  |  4-20MA -- 6 instances
|  |  |  TRIPHASE
|  |  PNEUMATIC
|  |  |  3-15PSI
|  |  |  CONSTANT-PRESSURE
|  |  |  AERATION
|  |  HYDRAULIC
|  |  |  TAP-WATER
|  |  |  PROCESS-STREAM
|  |  |  |  CONNECTION-STREAM
|  |  |  |  INFLUENT
|  |  |  |  INTERNAL-RECYCLE
|  |  |  |  EXTERNAL-RECYCLE
|  |  |  |  FINAL-EFFLUENT
|  |  |  CONCENTRATE
|  |  |  |  N-CONCENTRATE
|  |  |  |  C-CONCENTRATE
|  |  MANRESA-CONNECTION
|  |  |  LINIA-AIGUA-LINE
|  |  |  |  ENT-PLANTA-LINE
|  |  |  |  ENT-BASSA-LINE
```



```
|  |  |  |  SORT-BASSA-LINE
|  |  |  |  FANGS-LINE
|  |  |  |  SORT-PLANTA-LINE
|  |  |  |  SORRA-LINE
|  |  LIN -- 17 instances
|  OBJECT
|  |  DEFAULT-JUNCTION -- 2 instances
|  |  |  JUNCTION-BLOCK-FOR-ENT-PLANTA-LINE
|  |  |  JUNCTION-BLOCK-FOR-ENT-BASSA-LINE
|  |  |  JUNCTION-BLOCK-FOR-SORT-BASSA-LINE
|  |  |  JUNCTION-BLOCK-FOR-FANGS-LINE
|  |  |  JUNCTION-BLOCK-FOR-SORT-PLANTA-LINE
|  |  |  JUNCTION-BLOCK-FOR-SORRA-LINE
|  |  |  JUNCTION-BLOCK-FOR-RS485 -- 4 instances
|  |  |  JUNCTION-BLOCK-FOR-SERIAL
|  |  |  JUNCTION-BLOCK-FOR-220V
|  |  |  JUNCTION-BLOCK-FOR-TCP-IP
|  |  |  JUNCTION-BLOCK-FOR-TAP-WATER
|  |  |  JUNCTION-BLOCK-FOR-CONCENTRATE
|  |  |  JUNCTION-BLOCK-FOR-PROCESS-STREAM -- 3 instances
|  |  |  JUNCTION-BLOCK-FOR-PNEUMATIC
|  |  |  JUNCTION-BLOCK-FOR-3-15PSI
|  |  |  JUNCTION-BLOCK-FOR-CONSTANT-PRESSURE -- 6 instances
|  |  |  JUNCTION-BLOCK-FOR-AERATION
|  |  |  JUNCTION-BLOCK-FOR-INFLUENT
|  |  |  JUNCTION-BLOCK-FOR-INTERNAL-RECYCLE
|  |  |  JUNCTION-BLOCK-FOR-EXTERNAL-RECYCLE
|  |  |  JUNCTION-BLOCK-FOR-CONNECTION-STREAM
|  |  |  JUNCTION-BLOCK-FOR-FINAL-EFFLUENT
|  |  |  JUNCTION-BLOCK-FOR-DIGITAL
|  |  |  JUNCTION-BLOCK-FOR-ELECTRICAL
|  |  |  JUNCTION-BLOCK-FOR-RS232
|  |  |  JUNCTION-BLOCK-FOR-ON-OFF -- 1 instance
|  |  |  JUNCTION-BLOCK-FOR-24V -- 1 instance
|  |  |  JUNCTION-BLOCK-FOR-4-20MA
|  |  |  JUNCTION-BLOCK-FOR-TRIPHASE
|  |  |  JUNCTION-BLOCK-FOR-PROBE-SIGNAL
|  |  |  JUNCTION-BLOCK-FOR-LIN -- 2 instances
|  |  CONNECTION-POST -- 9 instances
|  |  G2-EXTENSION
|  |  |  G2-WINDOW -- 1 instance
|  |  G2-LIST
|  |  |  ITEM-LIST
|  |  |  |  UIL-NO-DUPLICATE-ITEM-LIST
|  |  |  VALUE-LIST
|  |  |  |  SYMBOL-LIST
|  |  |  |  TEXT-LIST
|  |  |  |  TRUTH-VALUE-LIST
|  |  |  |  QUANTITY-LIST
|  |  |  |  |  FLOAT-LIST
|  |  |  |  |  INTEGER-LIST
|  |  G2-ARRAY
|  |  |  ITEM-ARRAY -- 5 instances
|  |  |  VALUE-ARRAY
|  |  |  |  SYMBOL-ARRAY -- 2 instances
|  |  |  |  TEXT-ARRAY -- 13 instances
|  |  |  |  TRUTH-VALUE-ARRAY
|  |  |  |  QUANTITY-ARRAY
|  |  |  |  |  FLOAT-ARRAY -- 32 instances
|  |  |  |  |  INTEGER-ARRAY -- 110 instances
|  |  VARIABLE-OR-PARAMETER
|  |  |  VARIABLE
|  |  |  |  G2-VARIABLE
|  |  |  |  |  LOGICAL-VARIABLE -- 20 instances
|  |  |  |  |  QUANTITATIVE-VARIABLE -- 163 instances
|  |  |  |  |  |  FLOAT-VARIABLE -- 239 instances
|  |  |  |  |  |  |  G2-G2 -- 11 instances
|  |  |  |  |  |  INTEGER-VARIABLE -- 86 instances
|  |  |  |  |  |  GSI-VARIABLE
|  |  |  |  |  |  |  GSI-SETPOINT -- 22 instances
|  |  |  |  |  |  |  GSI-MEASURE -- 58 instances
|  |  |  |  |  SYMBOLIC-VARIABLE -- 25 instances
|  |  |  |  |  TEXT-VARIABLE -- 4 instances
|  |  |  |  SENSOR
|  |  |  |  GFI-DATA-SERVICE
|  |  |  |  GSI-DATA-SERVICE
|  |  |  |  |  GSI-VARIABLE
```

```
| | | | | | | GSI-SETPOINT -- 22 instances
| | | | | | | GSI-MEASURE -- 58 instances
| | | | | G2-TO-G2-DATA-SERVICE
| | | | | | G2-G2 -- 11 instances
| | | | | G2-METER-DATA-SERVICE
| | | | PARAMETER
| | | | | LOGICAL-PARAMETER -- 17 instances
| | | | | QUANTITATIVE-PARAMETER -- 10 instances
| | | | | | INTEGER-PARAMETER -- 6 instances
| | | | | | FLOAT-PARAMETER -- 9 instances
| | | | | SYMBOLIC-PARAMETER -- 16 instances
| | | | | TEXT-PARAMETER
| | | GFI-OUTPUT-INTERFACE -- 4 instances
| | | GFI-INPUT-INTERFACE
| | | GSI-INTERFACE
| | | | GSI-BRIDGE -- 1 instance
| | | G2-TO-G2-DATA-INTERFACE -- 1 instance
| | | UIL-OBJECT
| | | | UIL-GROBJ
| | | | | UIL-BUTTON
| | | | | | UIL-ICON-BUTTON
| | | | | | | UIL-SELECTION-BUTTON
| | | | | | | UIL-WORKSPACE-BUTTON
| | | | | | | | UIL-NAVIGATION-BUTTON
| | | | | | | | | UIL-GOTO-WORKSPACE-BUTTON -- 141 instances
| | | | | | | | | UIL-GOTO-SUPERIOR-BUTTON -- 14 instances
| | | | | | | | | UIL-GOTO-NEXT-BUTTON
| | | | | | | | | UIL-GOTO-PREVIOUS-BUTTON
| | | | | | | | | UIL-HIDE-BUTTON -- 132 instances
| | | | | | | | | UIL-HELP-BUTTON
| | | | | | | | UIL-WORKSPACE-OPERATION-BUTTON
| | | | | | | | | UIL-PRINT-WORKSPACE-BUTTON
| | | | | UIL-DIALOG
| | | | | UIL-SELECTION-BOX
| | | | | | UIL-CHECK-BOX
| | | | | | UIL-RADIO-BOX
| | | | UIL-CONFIGURATION-CLASS
| | | | | UIL-NAVIGATION-BUTTON-CONFIGURATION-CLASS -- 2 instances
| | | | | UIL-TEXT-CONFIGURATION-CLASS
| | | GUIDE-ROOT
| | | | GUIDE-BUTTON-TEMPLATE
| | | | | GUIDE-NAVIGATION-BUTTON-TEMPLATE
| | | | | | GUIDE-GOTO-WORKSPACE-BUTTON-TEMPLATE
| | | | | | GUIDE-GOTO-SUPERIOR-BUTTON-TEMPLATE
| | | | | | GUIDE-GOTO-NEXT-BUTTON-TEMPLATE
| | | | | | GUIDE-GOTO-PREVIOUS-BUTTON-TEMPLATE
| | | | | | GUIDE-HIDE-BUTTON-TEMPLATE
| | | | | | GUIDE-HELP-BUTTON-TEMPLATE
| | | | | | GUIDE-PRINT-WORKSPACE-BUTTON-TEMPLATE
| | | | GUIDE-GARBAGE-PAIL
| | | STAR-ICON -- 1 instance
| | | ITEM-PROFILE-INFORMATION
| | | ACTIVITY-PROFILE-INFORMATION
| | | SYSTEM-PROFILE-INFORMATION
| | | OBJ-SUBWS-DEACTIV -- 4 instances
| | | INFORMADORS -- 73 instances
| | | PLANT -- 1 instance
| | | PROCESS_UNIT
| | | | WASTE -- 1 instance
| | | | REACTOR
| | | | | AEROBIC-REACTOR -- 2 instances
| | | | | ANOXIC-REACTOR -- 1 instance
| | | | ANAEROBIC-REACTOR -- 1 instance
| | | | VESSEL
| | | | | VESSEL-N -- 1 instance
| | | | | VESSEL-C -- 1 instance
| | | | SETTLER -- 1 instance
| | | | TAP -- 1 instance
| | | | JUNCTION-BOX -- 1 instance
| | | | AIR-SUPPLY -- 3 instances
| | | | PURGE -- 1 instance
| | | | JUNCTION-OUTLET -- 1 instance
| | | INSTRUMENTATION
| | | | PLC -- 1 instance
| | | | OLD-PLC
| | | | TRANSDUCER
| | | | | I-P_TRANSDUCER -- 4 instances
```



```
| | | | I-V_TRANSDUCER -- 3 instances
| | | | VALVE
| | | | | ELECTROVALVE-3-WAYS -- 1 instance
| | | | | ELECTROVALVE-ON_OFF -- 2 instances
| | | | | CONTROL-VALVE
| | | | | | CONTROL-VALVE-LIQUID -- 1 instance
| | | | | | CONTROL-VALVE-AIR -- 3 instances
| | | | MEASURING-DEVICE
| | | | | DETECTOR
| | | | | | LEVEL-DETECTOR -- 4 instances
| | | | | | LEAK-DETECTOR -- 2 instances
| | | | | PROBE
| | | | | | ORP-PROBE -- 3 instances
| | | | | | PH-PROBE -- 3 instances
| | | | | | T-PROBE -- 3 instances
| | | | | | DO-PROBE -- 3 instances
| | | | CONTROLLER
| | | | | PHROCON-18 -- 3 instances
| | | | FREQUENCY-CONVERTER -- 3 instances
| | | | STIRRER -- 3 instances
| | | | PUMP
| | | | | VARIABLE-DOSING-PUMP -- 2 instances
| | | | | PERISTALTIC-PUMP -- 2 instances
| | | | | FIXED-DOSING-PUMP -- 2 instances
| | | | | ELECTRIC-FLOJET
| | | | | AIR_FLOJET -- 2 instances
| | | | ANALYSER
| | | | | NO3-ANALYSER -- 1 instance
| | | | | NO2-ANALYSER -- 1 instance
| | | | | NH4-ANALYSER -- 1 instance
| | | | | PO4-ANALYSER
| | | SLUDGE -- 1 instance
| | | COMPUTER-INTERFACE
| | | | RS232-RS485_CONVERTER -- 2 instances
| | | COMPUTER
| | | | PROCESS-COMPUTER -- 1 instance
| | | | ANALYZER-COMPUTER -- 1 instance
| | | | G2-COMPUTER -- 1 instance
| | | MICROORGANISM
| | | | MICROFAUNA
| | | | | OTHER-MICROFAUNA
| | | | | | ACINERA-UNCINATA -- 1 instance
| | | | | | TRACHELOPHYLLUM -- 1 instance
| | | | | | DREPANOMONAS -- 1 instance
| | | | | | OTHER-OTHER-MICROFAUNA -- 1 instance
| | | | | PROTOZOA
| | | | | | CILIATES
| | | | | | | SWIMMING-CILIATES
| | | | | | | | OTHER-SWIMMING-CILIATES -- 1 instance
| | | | | | | | CINETOCHILUM-MARGARITACEUM -- 1 instance
| | | | | | | | COLPIDIUM -- 1 instance
| | | | | | | | CYCLIDIUM -- 1 instance
| | | | | | | | GLAUCOMA -- 1 instance
| | | | | | | | PARAMECIUM-CAUDATOM -- 1 instance
| | | | | | | | TETRAHYMENA -- 1 instance
| | | | | | | | URONEMA -- 1 instance
| | | | | | | | SATHROPHILUS -- 1 instance
| | | | | | | ATTACHED-CILIATES
| | | | | | | | SPIROSTOMUM -- 1 instance
| | | | | | | | VORTICELLA-CONVALLARIA -- 1 instance
| | | | | | | | VORTICELLA-AQUADULCIS -- 1 instance
| | | | | | | | VORTICELLA-MICROSTOMA -- 1 instance
| | | | | | | | CARCHESIUM -- 1 instance
| | | | | | | | ZOOTHAMNIUM -- 1 instance
| | | | | | | | OPERCULARIA -- 1 instance
| | | | | | | | EPISTYLIS -- 1 instance
| | | | | | | | VAGINICOLA -- 1 instance
| | | | | | | | STENTOR -- 1 instance
| | | | | | | | OTHER-ATTACHED-CILIATES -- 1 instance
| | | | | | | CRAWLING-CILIATES
| | | | | | | | OTHER-CRAWLING-CILIATES -- 1 instance
| | | | | | | | ASPIDISCA-CICADA -- 1 instance
| | | | | | | | ASPIDISCA-LYNCEUS -- 1 instance
| | | | | | | | EUPLOTES-AFFINIS -- 1 instance
| | | | | | | | EUPLOTES-PATELLA -- 1 instance
| | | | | | | | STYLONYCHIA -- 1 instance
| | | | | | | | TROCHILIA -- 1 instance
```