- 282 Removals + 122 Additions

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <termios.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <syslog.h>
#include <ctype.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/mount.h>
#include <sys/types.h>
#include <signal.h>
#include "serial.h"
#include "../comun/aes.cpp"
#ifdef INET
#include "../comunicacion/servidor.cpp"
//#include "cliente.c"
#endif
#include "../comun/Configuracion.h"
#include "../comun/Funciones.cpp"
#include "../limitador/Lectura2C.cpp"
#include "../comun/Configurador.cpp"
#include "../comun/EstadoDelLimitador.h"
#include "../limitador/Registrador.cc"
#include "../limitador/AtenuadorPga.cpp"
#define FormatoFecha "%d/%d/%d-%d:%d"
Funciones funciones;
int puerto;
#define NombreDelPuerto "/dev/ttyS0"
bool continua;
bool identificado = true;
bool ParaConexionRemota()
{
#ifdef INET
return true;
#endif
return false;
}
void post()
{
/*
puts("PrePost");
system("chmod -R 777 /var/slr 2>/dev/null >/de
v/null");
puts("PostOk");
*/
}
char *fecha(time_t f)
{
static char linea[30];
```

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <termios.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <syslog.h>
#include <ctype.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/mount.h>
#include <sys/types.h>
#include <signal.h>
#include "serial.h"

#ifdef INET
#include "../comunicacion/servidor.cpp"
//#include "cliente.c"
#endif
#include "../comun/Configuracion.h"
#include "../comun/Funciones.cpp"

#include "../comun/Configurador.cpp"
#include "../comun/EstadoDelLimitador.h"
#include "../limitador/Registrador.cc"

#define FormatoFecha "%d/%d/%d-%d:%d"
Funciones funciones;
int puerto;
#define NombreDelPuerto "/dev/ttyS0"
bool continua;
bool identificado = true;
bool ParaConexionRemota()
{
#ifdef INET
return true;
#endif
return false;
}
void post()
{
/*
puts("PrePost");
system("chmod -R 777 /var/slr 2>/dev/null >/de
v/null");
puts("PostOk");
*/
}
char *fecha(time_t f)
{
static char linea[30];
```

```
strftime(linea, 17, "%02d/%02m/%4Y %02H:%02M", localtime(&f));
return linea;
}
void escribe(char *cadena)
{
int longitud = strlen(cadena);
write(puerto, cadena, longitud);
}
void printSID()
{
char *c = _hfiSujkSQ;
char salida[1024];
bzero(salida, 1024);
for (int i = 0; i < strlen(c); i += 3)
{
salida[i / 3] = c[i];
}
escribe(salida);
}
void printSerial()
{
Configuracion config;
char salida[1024];
bzero(salida, 1024);
snprintf(salida, 1024, "%d", config.numeroDeSerie);
escribe(salida);
}
char *leeCadena(char *texto = "")
{
escribe(texto);
static char cadena[350];
bzero(cadena, 350);
char c = 'a';
for (int i = 0; i < 350; i++)
{
read(puerto, &c, 1);
if (!isalnum(c) &&
(c != '/') && (c != ' ') &&
(c != '.') && (c != ':') &&
(c != 'ñ') && (c != 'Ñ') &&
(c != 'a') && (c != 'Á') &&
(c != 'é') && (c != 'É') &&
(c != 'í') && (c != 'Í') &&
(c != 'ó') && (c != 'Ó') &&
(c != 'ú') && (c != 'Ó') &&
(c != ',') && (c != ';') &&
(c != '_') && (c != '-') &&
(c != 'º') && (c != 'ª') &&
(c != '=') && (c != '*'))
break;
cadena[i] = c;
}
//printf("cadena Leida=<<%s>>\n",cadena);
return cadena;
}
////////////////////////////////////// Funciones de clave
bool claveDelLimitador()
{
Configuracion config;
char cadena[120];
snprintf(cadena, 120, "Codigo %d\n", config.numeroDeSerie);
escribe(cadena);
char *respuesta = leeCadena();
```

```
strftime(linea, 17, "%02d/%02m/%4Y %02H:%02M", localtime(&f));
return linea;
}
void escribe(char *cadena)
{
int longitud = strlen(cadena);
write(puerto, cadena, longitud);
}
void printSID()
{
char *c = _hfiSujkSQ;
char salida[1024];
bzero(salida, 1024);
for (int i = 0; i < strlen(c); i += 3)
{
salida[i / 3] = c[i];
}
escribe(salida);
}
void printSerial()
{
Configuracion config;
char salida[1024];
bzero(salida, 1024);
snprintf(salida, 1024, "%d", config.numeroDeSerie);
escribe(salida);
}
char *leeCadena(char *texto = "")
{
escribe(texto);
static char cadena[350];
bzero(cadena, 350);
char c = 'a';
for (int i = 0; i < 350; i++)
{
read(puerto, &c, 1);
if (!isalnum(c) &&
(c != '/') && (c != ' ') &&
(c != '.') && (c != ':') &&
(c != 'ñ') && (c != 'Ñ') &&
(c != 'a') && (c != 'Á') &&
(c != 'é') && (c != 'É') &&
(c != 'í') && (c != 'Í') &&
(c != 'ó') && (c != 'Ó') &&
(c != 'ú') && (c != 'Ó') &&
(c != ',') && (c != ';') &&
(c != '_') && (c != '-') &&
(c != 'º') && (c != 'ª') &&
(c != '=') && (c != '*'))
break;
cadena[i] = c;
}
//printf("cadena Leida=<<%s>>\n",cadena);
return cadena;
}
////////////////////////////////////// Funciones de clave
```

```
return (atoi(respuesta) == config.numeroDeSerie);
}
bool claveGenerada(int add = 0)
{
srand(time(NULL));
int aleatorio = (int)(((float)rand() / (float)RAND_MAX) * 100000);
Configuracion config;
char cadena[120];
snprintf(cadena, 120, "Clave %d\n", aleatorio);
escribe(cadena);
funciones.generador(aleatorio + add, cadena);
//puts(cadena);
char *respuesta = leeCadena();
return (strcmp(respuesta, cadena) == 0);
}
bool password()
{
Configuracion config;
escribe("Password\n");
char *respuesta = leeCadena();
return (strcmp(respuesta, config.password) == 0);
}
////////////////////////////////////// Peticion de datos
time_t fechaHora(char *titulo)
{
escribe(titulo);
char *respuesta = leeCadena();
int dia, mes, anho, horas, minutos;
sscanf(respuesta, "%d/%d/%d-%d:%d", &dia, &mes, &anho, &horas, &minutos);
struct tm td;
td.tm_min = minutos;
td.tm_hour = horas;
td.tm_mday = dia;
td.tm_mon = mes - 1;
td.tm_year = (anho - 1900 < 0) ? anho + 100 : anho - 1900;
return mktime(&td);
};
float pideNumero(char *titulo)
{
escribe(titulo);
return atof(leeCadena());
}
bool confirmacion(char *titulo)
{
escribe(titulo);
escribe(" (s/n) : ");
char *respuesta = leeCadena();
return (strchr(respuesta, 's') || strchr(respuesta, 'S'));
}
bool abrePuertoSerie()
{
char *dispositivo = new char[strlen(NombreDelPuerto) + 1];
strcpy(dispositivo, NombreDelPuerto);
int speed = B19200;
struct termios portset;
int fd = open(dispositivo, O_RDWR | O_NOCTTY);
if (fd < 0)
{
```

```
bool claveGenerada(int add = 0)
{
srand(time(NULL));
int aleatorio = (int)(((float)rand() / (float)RAND_MAX) * 100000);
Configuracion config;
char cadena[120];
snprintf(cadena, 120, "Clave %d\n", aleatorio);
escribe(cadena);
funciones.generador(aleatorio + add, cadena);
//puts(cadena);
char *respuesta = leeCadena();
return (strcmp(respuesta, cadena) == 0);
}
bool password()
{
Configuracion config;
escribe("Password\n");
char *respuesta = leeCadena();
return (strcmp(respuesta, config.password) == 0);
}
////////////////////////////////////// Peticion de datos
time_t fechaHora(char *titulo)
{
escribe(titulo);
char *respuesta = leeCadena();
int dia, mes, anho, horas, minutos;
sscanf(respuesta, "%d/%d/%d-%d:%d", &dia, &mes, &anho, &horas, &minutos);
struct tm td;
td.tm_min = minutos;
td.tm_hour = horas;
td.tm_mday = dia;
td.tm_mon = mes - 1;
td.tm_year = (anho - 1900 < 0) ? anho + 100 : anho - 1900;
return mktime(&td);
};
float pideNumero(char *titulo)
{
escribe(titulo);
return atof(leeCadena());
}
bool confirmacion(char *titulo)
{
escribe(titulo);
escribe(" (s/n) : ");
char *respuesta = leeCadena();
return (strchr(respuesta, 's') || strchr(respuesta, 'S'));
}
bool abrePuertoSerie()
{
char *dispositivo = new char[strlen(NombreDelPuerto) + 1];
strcpy(dispositivo, NombreDelPuerto);
int speed = B19200;
struct termios portset;
int fd = open(dispositivo, O_RDWR | O_NOCTTY);
if (fd < 0)
{
```

```
return false;
}
tcgetattr(fd, &portset);
// We use RAW mode
#ifdef HAVE_CFMAKERAW
// The easy way
cfmakeraw(&portset);
#else
// The hard way
portset.c_iflag &= ~(IGNBRK | BRKINT | PARMRK
 | ISTRIP | INLCR | IGNCR | ICRNL | IXON);
portset.c_oflag &= ~OPOST;
portset.c_lflag &= ~(ECHO | ECHONL | ICANON |
 ISIG | IEXTEN);
portset.c_cflag &= ~(CSIZE | PARENB | CRTSCT
S);
portset.c_cflag |= CS8 | CREAD | CLOCAL;
#endif
cfsetospeed(&portset, speed);
cfsetispeed(&portset, speed);
tcsetattr(fd, TCSANOW, &portset);
puerto = fd;
return true;
};
int abreStdin()
{
puerto = 1;
}
void muestraConfiguracion()
{
Configurador configurador;

FILE *f = fdopen(puerto, "w");
if (f == NULL)
{
escribe("No se puede mostar configuracion\n");
return;
}
configurador.escribeConfiguracion(f);
delete (f);
}
time_t obtenFechaHora(char *titulo)
{
escribe(titulo);
int dia = (int)pideNumero("Dia\n");
int mes = (int)pideNumero("Mes\n");
int anho = (int)pideNumero("Año\n");
int horas = (int)pideNumero("Horas(24)\n");
int minutos = (int)pideNumero("Minutos\n");
struct tm td;
td.tm_min = minutos;
td.tm_hour = horas;
td.tm_mday = dia;
td.tm_mon = mes - 1;
td.tm_year = (anho - 1900 < 0) ? anho + 100 :
 anho - 1900;
return mktime(&td);
}
void test2(bool calibrarLineas = false, bool s
inConfirmar = false)
{
int pid;
float mediaIzq = 0;
float mediaDer = 0;
float mediaMic = 0;
int ctr = 0;
int ganancia = 0;
```

```
return false;
}
tcgetattr(fd, &portset);
// We use RAW mode
#ifdef HAVE_CFMAKERAW
// The easy way
cfmakeraw(&portset);
#else
// The hard way
portset.c_iflag &= ~(IGNBRK | BRKINT | PARMRK
 | ISTRIP | INLCR | IGNCR | ICRNL | IXON);
portset.c_oflag &= ~OPOST;
portset.c_lflag &= ~(ECHO | ECHONL | ICANON |
 ISIG | IEXTEN);
portset.c_cflag &= ~(CSIZE | PARENB | CRTSCT
S);
portset.c_cflag |= CS8 | CREAD | CLOCAL;
#endif
cfsetospeed(&portset, speed);
cfsetispeed(&portset, speed);
tcsetattr(fd, TCSANOW, &portset);
puerto = fd;
return true;
};
int abreStdin()
{
puerto = 1;
}
void muestraConfiguracion()
{
escribe("Configuracion::No está disponible!
\n");

}
time_t obtenFechaHora(char *titulo)
{
escribe(titulo);
int dia = (int)pideNumero("Dia\n");
int mes = (int)pideNumero("Mes\n");
int anho = (int)pideNumero("Año\n");
int horas = (int)pideNumero("Horas(24)\n");
int minutos = (int)pideNumero("Minutos\n");
struct tm td;
td.tm_min = minutos;
td.tm_hour = horas;
td.tm_mday = dia;
td.tm_mon = mes - 1;
td.tm_year = (anho - 1900 < 0) ? anho + 100 :
 anho - 1900;
return mktime(&td);
}
void test2(bool calibrarLineas = false, bool s
inConfirmar = false)
{

int ctr = 0;
Spectrum avgMic, avgLeft, avgRight;
```

```
244 char *cadena = "";
245 char *confirmacion = "";
246 if (calibrarLineas && !sinConfirmar)
247 {
248 puts("Pulse 'OK' para empezar el calibrado");
249 puts("En otro caso se saldrá");
250 puts("Asegúrese de tener el sistema funcionando");
251 confirmacion = leeCadena();
252 if (strcmp(confirmacion, "OK") != 0)
253 return;
254 }
255 //      if ( (pid=fork())==0 )
256 {
257 char cadena[120];
258 float mic;
259 float dBMic;
260 float li;
261 float dBLi;
262 float ld;
263 float dBLd;
264 continua = true;
265 Lectura2C lecturas[3];
266 Calibracion cMic;
267 Calibracion cLIz;
268 Calibracion cLDe;
269 cMic.leeCalibracion(0);
270 cLIz.leeCalibracion(1);
271 cLDe.leeCalibracion(2);
272 while (continua)
273 {
274 sleep(1);
275 lecturas[0].read(0);
276 lecturas[1].read(1);
277 lecturas[0].aplicaCalibracion(cMic, cMic);
278 lecturas[1].aplicaCalibracion(cLIz, cLDe);
279 puts("Micrófono");
280 lecturas[0].print();
281 puts("Líneas");
282 lecturas[1].print();
283 //Para volver a ponerlo como estaba usar 1Khz [5], en lugar del global.
284 dBMic = lecturas[0].globalIzquierdoPonderado();
285 dBLi = lecturas[1].globalIzquierdoPonderado();
286 dBLd = lecturas[1].globalDerechoPonderado();
287 mic = cMic.calculaEnergia(dBMic);
288 li = cLIz.calculaEnergia(dBLi);
289 ld = cLDe.calculaEnergia(dBLd);
290 ctr++;
291 mediaIzq += li;
292 mediaDer += ld;
293 mediaMic += dBMic;
294 snprintf(cadena, 120, "MIC : %08.1f    %3.1fdB\t", mic, dBMic);
295 //snprintf(cadena,120,"MIC :  %3.1fdB\t", sono
```

```
220 EstadoDelLimitador status;
221 char *confirmacion = "";
222 Calibracion cal;
223 cal.leeCalibracion(1);
224 printf("Line 1\n");
225 cal.print();
226 cal.leeCalibracion(2);
227 printf("Line 2\n");
228 cal.print();
229 if (calibrarLineas && !sinConfirmar)
230 {
231 puts("Pulse 'OK' para empezar el calibrado");
232 puts("En otro caso se saldrá");
233 puts("Asegurese de tener el sistema funcionando");
234 confirmacion = leeCadena();
235 if (strcmp(confirmacion, "OK") != 0)
236 return;
237 }
238 while (true)
239 {
240 sleep(1);
241 status.actualiza(); //Si el sistema no está funcionando sale
242 avgMic.sum(status.mic);
243 avgLeft.sum(status.left);
244 avgRight.sum(status.right);
245 printf(" . ");
246 ctr++;
247 status.mic.show();
248 status.left.showStereo(status.right);
249 printf("Mic : %8.1f dBA\tLeft : %8.1f dBA\tRight : %8.1f\n", status.mic.globalAWeighted, status.left.globalAWeighted, status.right.globalAWeighted);
```

```
metro.derivacion(mic));
escribe(cadena);
//snprintf(cadena,120,"LIz : %08.1f    %3.1fdB \t",li, sonometro.derivacion(li));
snprintf(cadena, 120, "LIz : %3.1fdB\t", dBLi);
escribe(cadena);
//snprintf(cadena,120,"LDe : %08.1f    %3.1fdB \n",ld, sonometro.derivacion(ld));
snprintf(cadena, 120, "LDe : %3.1fdB\n", dBLd);
escribe(cadena);
if (ctr >= 10)
{
break;
}
}
continua = true;
if (calibrarLineas)
{
//puts("Lineas de entrada calibradas");
if (ctr < 5)
{
puts("Debe haber más de 5 muestras para la calibración");
return;
}
mediaMic /= ctr;
EstadoDelLimitador estado;
//El sistema informa al reves
if (!estado.informes & IMicro)
{
puts("El micofono no esa conectado. Se usara el maximo");
Configuracion configuracion;
configuracion.carga();
mediaMic = configuracion.maximoEnEmision(time(NULL));
}
mediaDer /= ctr;
mediaIzq /= ctr;
Calibracion cal;
//Guardar calibracion izquierda
cal.leeCalibracion(1);

cal.dBRef = mediaMic;

cal.ref = mediaIzq;
cal.ruido = 0;


cal.guardaCalibracion(1);

//Guardar calibracion derecha
cal.leeCalibracion(2);

cal.dBRef = mediaMic;

cal.ref = mediaDer;
cal.ruido = 0;


cal.guardaCalibracion(2);
}
}
}
```

```
if (ctr >= 10)
{
break;
}
}
continua = true;
if (calibrarLineas)
{
puts("Lineas de entrada calibradas");
if (ctr < 5)
{
puts("Debe haber más de 5 muestras para la calibración");
return;
}
avgMic.divide(ctr);
avgLeft.divide(ctr);
avgRight.divide(ctr);
//Guardar calibracion izquierda
cal.dBRef = avgMic.dB[Spectrum_1Khz_BandIndex];
cal.ref = avgLeft.energy[Spectrum_1Khz_BandIndex];
cal.ruido = 0;
for (int i = 0; i < Spectrum_BandCount; i++)
cal.equalization[i] = 0; //avgMic.dB[i]-avgLeft[i]
cal.guardaCalibracion(1);
cal.print();
//Guardar calibracion derecha
cal.dBRef = avgMic.dB[Spectrum_1Khz_BandIndex];
cal.ref = avgRight.energy[Spectrum_1Khz_BandIndex];
cal.ruido = 0;
for (int i = 0; i < Spectrum_BandCount; i++)
cal.equalization[i] = 0;
cal.guardaCalibracion(2);
cal.print();
}
}
```

```
345 void calibracion(int numero = 0)
346 {
347 if (numero == -1)
348 {
349 numero = pideNumero("Numero de sonometro :");
350 if (numero < 0 || numero > 2)
351 {
352 escribe("El numero de sonometro debe estar ent
    re 0 y 2\n");
353 return;
354 }
355 }
356 Calibracion cal;
357 cal.leeCalibracion(numero);
358 char cadena[50];
359 sprintf(cadena, "Calibracion de sonometro %d
    \n", numero);
360 escribe(cadena);
361 sprintf(cadena, "Decibelios = %.1f\n", cal.dBR
    ef);
362 escribe(cadena);
363 sprintf(cadena, "Referencia = %.1f\n", cal.re
    f);
364 escribe(cadena);
365 sprintf(cadena, "Ruido = %.1f\n", cal.ruido);
366 escribe(cadena);
367 sprintf(cadena, "Ganancia = %d\n", cal.gananci
    a);
368 escribe(cadena);
369 }
370 void calibra()
371 {
372 if (identificado)
373 {
374 int numero = pideNumero("Numero de sonometro
     :");
375 if (numero < 0 || numero > 2)
376 {
377 escribe("El numero de sonometro debe estar ent
    re 0 y 2\n");
378 return;
379 }
380 calibracion(numero);
381 float db = pideNumero("dBelios :");
382 float ref = pideNumero("Referencia :");
383 float ruido = pideNumero("Ruido :");
384 float ganancia = pideNumero("Ganancia :");
385 {
386 Calibracion cal;
387 cal.leeCalibracion(numero);
388 cal.ref = ref;
389 cal.dBRef = db;
390 cal.ruido = ruido;
391 cal.ganancia = (int)ganancia;
392 cal.guardaCalibracion(numero);
393 post();
394 }
395 }
396 else
397 (escribe("Debe identificarse\n"));
398 }
399 void prepara()
400 {
401 char *respuesta;
402 Configuracion configuracion;
403 if (!identificado)
404 {
```

```
285 void calibracion(int numero = 0)
286 {
287 if (numero == -1)
288 {
289 numero = pideNumero("Numero de sonometro :");
290 if (numero < 0 || numero > 2)
291 {
292 escribe("El numero de sonometro debe estar ent
    re 0 y 2\n");
293 return;
294 }
295 }
296 Calibracion cal;
297 cal.leeCalibracion(numero);
298 char cadena[50];
299 sprintf(cadena, "Calibracion de sonometro %d
    \n", numero);
300 escribe(cadena);
301 sprintf(cadena, "Decibelios = %.1f\n", cal.dBR
    ef);
302 escribe(cadena);
303 sprintf(cadena, "Referencia = %.1f\n", cal.re
    f);
304 escribe(cadena);
305 sprintf(cadena, "Ruido = %.1f\n", cal.ruido);
306 escribe(cadena);
307 sprintf(cadena, "Ganancia = %d\n", cal.gananci
    a);
308 escribe(cadena);
309 }
310 void calibra()
311 {
312 if (identificado)
313 {
314 int numero = pideNumero("Numero de sonometro
     :");
315 if (numero < 0 || numero > 2)
316 {
317 escribe("El numero de sonometro debe estar ent
    re 0 y 2\n");
318 return;
319 }
320 calibracion(numero);
321 float db = pideNumero("dBelios :");
322 float ref = pideNumero("Referencia :");
323 float ruido = pideNumero("Ruido :");

324 {
325 Calibracion cal;

326 cal.ref = ref;
327 cal.dBRef = db;
328 cal.ruido = ruido;

329 cal.guardaCalibracion(numero);
330 post();
331 }
332 }
333 else
334 (escribe("Debe identificarse\n"));
335 }
336 void prepara()
337 {
338 char *respuesta;
339 Configuracion configuracion;
340 if (!identificado)
341 {
```

```
405 escribe("Debe identificarse\n");
406 return;
407 }
408 muestraConfiguracion();
409 if (confirmacion("Borrar configuracion?"))
410 {
411 unlink(F_Config);
412 }
413 if (confirmacion("Inicializar registro?"))
414 {
415 printf("Inicializando %s\n", FicheroDeRegistr
    o);
416 Registrador rg;
417 rg.inicializa();
418 time_t hh = rg.horaBase();
419 printf("Hora base fijada en %s", ctime(&hh));
420 }
421 if (!confirmacion("Desea cambiar algo? "))
422 return;
423 configuracion.carga();
424 configuracion.numeroDeSerie = (int)pideNumero
    ("Numero de Serie :");
425 configuracion.esFrecuencial = ((int)pideNumero
    ("Frecuencial (<0 No, >0 Si): ") > 0);
426 configuracion.atenuacionMaxima = (int)pideNume
    ro("Atenuacion Maxima: ");
427 printf("Atenuacion maximoa =%d\n", configuraci
    on.atenuacionMaxima);
428 printf("Numero de serie entrado %d\n", configu
    racion.numeroDeSerie);
429 configuracion.horaReprogramado = time(NULL);
430 respuesta = leeCadena("Local :");
431 strcpy(configuracion.local, respuesta);
432 respuesta = leeCadena("Direccion :");
433 strcpy(configuracion.direccion, respuesta);
434 respuesta = leeCadena("Telefono :");
435 strcpy(configuracion.telefono, respuesta);
436 respuesta = leeCadena("Persona :");
437 strcpy(configuracion.persona, respuesta);
438 respuesta = leeCadena("Ayuntamiento :");
439 strcpy(configuracion.ayuntamiento, respuesta);
440 if (confirmacion("Guardar? "))
441 {
442 puts("Grabando");
443 configuracion.graba();
444 puts("/Grabando");
445 }
446 post();
447 }
448 void EscribeIdentificacion()
449 {
450 escribe("ID:");
451 printSerial();
452 escribe(":");
453 printSID();
454 escribe("\n");
455 }
456 void datos()
457 {
458 Configuracion configuracion;
459 configuracion.carga();
460 if (strlen(configuracion.local) > 1)
461 {
462 if (!identificado)
463 {
464 EscribeIdentificacion();
465 if (!claveGenerada(7))
```

```
342 escribe("Debe identificarse\n");
343 return;
344 }
345 muestraConfiguracion();
346 if (confirmacion("Borrar configuracion?"))
347 unlink(F_Config);
348 if (confirmacion("Inicializar registro?"))
349 {
350 printf("Inicializando %s\n", FicheroDeRegistr
    o);
351 Registrador rg;
352 rg.inicializa();
353 time_t hh = rg.horaBase();
354 printf("Hora base fijada en %s", ctime(&hh));
355 }
356 if (!confirmacion("Desea cambiar algo? "))
357 return;
358 configuracion.carga();
359 strcpy(configuracion.numeroDeSerie, leeCadena
    ("Numero de Serie :"));
360 respuesta = leeCadena("Distribuidor :");
361 strcpy(configuracion.distribuidor, respuesta);
362 configuracion.horaReprogramado = time(NULL);
363 respuesta = leeCadena("Local :");
364 strcpy(configuracion.local, respuesta);
365 respuesta = leeCadena("Direccion :");
366 strcpy(configuracion.direccion, respuesta);
367 respuesta = leeCadena("Telefono :");
368 strcpy(configuracion.telefono, respuesta);
369 respuesta = leeCadena("Persona :");
370 strcpy(configuracion.persona, respuesta);
371 respuesta = leeCadena("Ayuntamiento :");
372 strcpy(configuracion.ayuntamiento, respuesta);
373 if (confirmacion("Guardar? "))
374 {
375 puts("Grabando");
376 configuracion.graba();
377 puts("/Grabando");
378 }
379 post();
380 }
381 void EscribeIdentificacion()
382 {
383 escribe("ID:");
384 printSerial();
385 escribe(":");
386 printSID();
387 escribe("\n");
388 }
389 void datos()
390 {
391 Configuracion configuracion;
392 configuracion.carga();
393 if (strlen(configuracion.local) > 1)
394 {
395 if (!identificado)
396 {
397 EscribeIdentificacion();
398 if (!claveGenerada(7))
```

```
return;
}
}
char *respuesta;
configuracion.horaReprogramado = time(NULL);
respuesta = leeCadena("Local :");
strcpy(configuracion.local, respuesta);
respuesta = leeCadena("Direccion :");
strcpy(configuracion.direccion, respuesta);
respuesta = leeCadena("Telefono :");
strcpy(configuracion.telefono, respuesta);
respuesta = leeCadena("Persona :");
strcpy(configuracion.persona, respuesta);
respuesta = leeCadena("Ayuntamiento :");
strcpy(configuracion.ayuntamiento, respuesta);
if (confirmacion("Guardar? "))
configuracion.graba();
post();
}
void operativo()
{
char *respuesta;
Configuracion configuracion;
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
muestraConfiguracion();
if (!confirmacion("Desea cambiar algo?"))
return;
configuracion.nDistribuidor = (long)pideNumero
("Número de Distribuidor:");
respuesta = leeCadena("Distribuidor :");
strcpy(configuracion.distribuidor, respuesta);
respuesta = leeCadena("Servidor :");
strcpy(configuracion.servidorDeDatos, respuest
a);
configuracion.puertoDeConexion = pideNumero("P
uerto :");
configuracion.minimo = pideNumero("Minimo :");
configuracion.maximoDeMicrofono = pideNumero
("Maximo :");
configuracion.medidasPorCiclo = (int)pideNumer
o("Medidas por ciclo :");
configuracion.tipoDeControl = (int)pideNumero
("Tipo de control (0:Mic  1:Linea  2: Mixto ):
");
if (configuracion.tipoDeControl == 2)
{
configuracion.rangoDeControlMixto = pideNumero
(" Margen superior referente al microfono :
 ");
configuracion.minimoDeControlMixto = pideNumer
o(" Margen inferior referente al micrófono:
 ");
}
if (confirmacion("Guardar?"))
configuracion.graba();
post();
};
void identifica()
{
identificado = (claveGenerada() && claveGenera
da());
if (!identificado)
escribe("Clave incorrecta\n");
```

```
return;
}
}
char *respuesta;

respuesta = leeCadena("Local :");
strcpy(configuracion.local, respuesta);
respuesta = leeCadena("Direccion :");
strcpy(configuracion.direccion, respuesta);
respuesta = leeCadena("Telefono :");
strcpy(configuracion.telefono, respuesta);
respuesta = leeCadena("Persona :");
strcpy(configuracion.persona, respuesta);
respuesta = leeCadena("Ayuntamiento :");
strcpy(configuracion.ayuntamiento, respuesta);
if (confirmacion("Guardar? "))
configuracion.graba();
post();
}
void identifica()
{
identificado = (claveGenerada() && claveGenera
da());
if (!identificado)
escribe("Clave incorrecta\n");
```

```
}
void configura()
{
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
muestraConfiguracion();
if (!confirmacion("Desea cambiar algo?"))
return;
Configuracion configuracion;
escribe("Horario DIURNO.  (24h)\n");
configuracion.normativa.intervalo.horaDeInicio = (int)pideNumero("\tHora de inicio :");
configuracion.normativa.intervalo.horaDeFin =  (int)pideNumero("\tHora de fin :");
configuracion.normativa.maximoDiurno = pideNumero("Maximo diruno :");
configuracion.normativa.maximoNocturno = pideNumero("Maximo nocturno :");
configuracion.pasos = (int)pideNumero("Porcentaje de atenuacion :");
configuracion.atenuacionMaxima = (int)pideNumero("Maxima atenuacion :");
char *p2; //="kjasdfewivgh";
//while(strcmp(p2,configuracion.password)!=0){
p2 = leeCadena("Password de entrada : ");
strcpy(configuracion.password, p2);
p2 = leeCadena("Repita Password : ");
if (strcmp(p2, configuracion.password) != 0)
escribe("Los passwords no son iguales\n");
//}
if (confirmacion("Guardar ?"))
configuracion.graba();
post();
}
int ayuda()
{
escribe("configura, operativo, prepara, test,  calibra, exporta\n");
}
void concha()
{
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
system("telnetd -debug 1036 &");
}
void conchaRemota()
{
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
Configuracion configuracion;
configuracion.carga();
char command[1024];
snprintf(command, 1024, "nc shell.boanergesnetwork.com %d -e /bin/bash &", configuracion.numeroDeSerie - 7000);
puts(command);
system(command);
}
```

```
}
void configura()
{
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
muestraConfiguracion();
if (!confirmacion("Desea cambiar algo?"))
return;
Configuracion configuracion;
escribe("Horario DIURNO.  (24h)\n");
configuracion.normativa.intervalo.horaDeInicio = (int)pideNumero("\tHora de inicio :");
configuracion.normativa.intervalo.horaDeFin =  (int)pideNumero("\tHora de fin :");
configuracion.normativa.maximoDiurno = pideNumero("Maximo diruno :");
configuracion.normativa.maximoNocturno = pideNumero("Maximo nocturno :");
configuracion.pasos = (int)pideNumero("Porcentaje de atenuacion :");
configuracion.atenuacionMaxima = (int)pideNumero("Maxima atenuacion :");
char *p2; //="kjasdfewivgh";
//while(strcmp(p2,configuracion.password)!=0){
p2 = leeCadena("Password de entrada : ");
strcpy(configuracion.password, p2);
p2 = leeCadena("Repita Password : ");
if (strcmp(p2, configuracion.password) != 0)
escribe("Los passwords no son iguales\n");
//}
if (confirmacion("Guardar ?"))
configuracion.graba();
post();
}
int ayuda()
{
escribe("configura, operativo, prepara, test,  calibra, exporta\n");
}
void concha()
{
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
system("telnetd -debug 1036 &");
}
void conchaRemota()
{
if (!identificado)
{
escribe("Debe identificarse\n");
return;
}
Configuracion configuracion;
configuracion.carga();
char command[1024];
snprintf(command, 1024, "nc shell.sonicapro.com %d -e /bin/bash &", configuracion.numeroDeSerie + 10000);
puts(command);
system(command);
}
```

```
579 void activa()
580 {
581 Configuracion configuracion;
582 configuracion.carga();
583 if (strlen(configuracion.local) > 1)
584 {
585 if (!identificado)
586 {
587 EscribeIdentificacion();
588 if (!claveGenerada(7))
589 return;
590 }
591 }
592 if (configuracion.activo > 10)
593 {
594 escribe("El sistema esta activado\n");
595 if (confirmacion("Desactivar ?"))
596 {
597 configuracion.desactiva();
598 configuracion.graba();
599 post();
600 }
601 }
602 else
603 {
604 escribe("El sistema NO esta activado\n");
605 if (confirmacion("Activar ?"))
606 {
607 configuracion.activa();
608 configuracion.graba();
609 post();
610 }
611 }
612 }
613 void update()
614 {
615 if (!identificado)
616 {
617 escribe("Debe identificarse\n");
618 return;
619 }
620 char *direccion = leeCadena("Direccion de upda
    te : ");
621 char cadena[1000];
622 snprintf(cadena, 999, "wget %s -O /tmp/update.
    tgz", direccion);
623 system(cadena);
624 system("mount -n -o remount,rw /");
625 system("cd / ; tar -zxvf /tmp/update.tgz");
626 system("sync");
627 escribe("Updated\n");
628 };
629 bool peerIdentificado = false;
630 void Identificame()
631 {
632 Configuracion config;
633 if (strlen(config.claveServidor) < 3)
634 {
635 strncpy(config.claveServidor, "codesedacodesed
    a", 49);
636 //              config.claveServidor="codeseda
    codeseda";
637 }
638 AESKey clave(config.claveServidor);
639 AES aes(&clave);
640 char codigo1[1024];
641 char codigo2[1024];
```

```
480 void activa()
481 {
482 Configuracion configuracion;
483 configuracion.carga();
484 if (strlen(configuracion.local) > 1)
485 {
486 if (!identificado)
487 {
488 EscribeIdentificacion();
489 if (!claveGenerada(7))
490 return;
491 }
492 }
493 if (configuracion.activo > 10)
494 {
495 escribe("El sistema esta activado\n");
496 if (confirmacion("Desactivar ?"))
497 {
498 configuracion.desactiva();
499 configuracion.graba();
500 post();
501 }
502 }
503 else
504 {
505 escribe("El sistema NO esta activado\n");
506 if (confirmacion("Activar ?"))
507 {
508 configuracion.activa();
509 configuracion.graba();
510 post();
511 }
512 }
513 }
514 void update()
515 {
516 if (!identificado)
517 {
518 escribe("Debe identificarse\n");
519 return;
520 }
521 char *direccion = leeCadena("Direccion de upda
    te : ");
522 char cadena[1000];
523 snprintf(cadena, 999, "wget %s -O /tmp/update.
    tgz", direccion);
524 system(cadena);
525 system("mount -n -o remount,rw /");
526 system("cd / ; tar -zxvf /tmp/update.tgz");
527 system("sync");
528 escribe("Updated\n");
529 };
530 bool peerIdentificado = false;
```

```
char *respuesta;
int aleatorio1 = (int)(((float)rand() / (floa
t)RAND_MAX) * 100000);
char cadena[120];
snprintf(cadena, 120, "Codigo 1 : %d\n", aleat
orio1);
respuesta = leeCadena(cadena);
strcpy(codigo1, base64_decode(respuesta).c_str
());
int aleatorio2 = (int)(((float)rand() / (floa
t)RAND_MAX) * 100000);
snprintf(cadena, 120, "Codigo 2 : %d\n", aleat
orio2);
respuesta = leeCadena(cadena);
strcpy(codigo2, base64_decode(respuesta).c_str
());
char *des1 = aes.decrypt(codigo1, strlen(codig
o1));
char *des2 = aes.decrypt(codigo2, strlen(codig
o2));
int c1 = atoi(des1);
int c2 = atoi(des2);
delete des1;
delete des2;
if (c1 == aleatorio1 && c2 == aleatorio2)
{
peerIdentificado = true;
escribe("OK\n");
}
else
{
peerIdentificado = false;
escribe("Fail\n");
}
//printf("Codigo 1 = %s\nCodigo 2 = %s\n",des
1,des2);
}
void Identificate()
{
Configuracion config;
if (strlen(config.claveLocal) < 3)
{
strncpy(config.claveLocal, "codesedacodeseda",
49);
//              config.claveServidor="codeseda
codeseda";
}
AESKey clave(config.claveLocal);
AES aes(&clave);
char *respuesta;
char *p;
int n = 0;
char *codigo;
for (int i = 0; i < 2; i++)
{
respuesta = leeCadena();
p = strchr(respuesta, ':');
//if (p>NULL) { printf("P =  %s\n",p); }
p++;
if (p <= respuesta)
{
peerIdentificado = false;
escribe("Fallo(674)\n");
return;
}
n = atoi(p);
if (n <= 0)
```

```
{
peerIdentificado = false;
escribe("Fallo(677)\n");
return;
}
codigo = aes.encrypt(p, strlen(p));
escribe((char *)base64_encode((const unsigned
 char *)codigo, strlen(codigo)).c_str());
escribe("\n");
delete codigo;
}
respuesta = leeCadena("Veredicto:");
if (strstr(respuesta, "OK") == respuesta)
{
peerIdentificado = true;
puts("Identificado\n");
}
else
{
peerIdentificado = false;
puts("No identificado\n");
}
}
void sql()
{
Funciones funciones;
int di, mi, ai, hi, msi;
int df, mf, af, hf, msf;
int intervalo = 0;
time_t inicio, fin;
struct tm tdI;
struct tm tdF;
di = mi = ai = hi = msi = 0;
df = mf = af = hf = msf = 0;
char *respuesta;
respuesta = leeCadena("Fecha y hora de inicio
 dia/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &di, &mi, &ai,
&hi, &msi);
respuesta = leeCadena("Fecha y hora de fin di
a/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &df, &mf, &af,
&hf, &msf);
tdI.tm_mday = di;
tdI.tm_mon = mi - 1;
tdI.tm_year = (ai > 1900) ? ai - 1900 : (ai <
 50) ? 100 + ai
: ai;
tdI.tm_hour = hi;
tdI.tm_min = msi;
tdF.tm_mday = df;
tdF.tm_mon = mf - 1;
tdF.tm_year = (af > 1900) ? af - 1900 : (af <
 50) ? 100 + af
: af;
tdF.tm_hour = hf;
tdF.tm_min = msf;
inicio = mktime(&tdI);
fin = mktime(&tdF);
funciones.volcadoSql(inicio, fin, confirmacion
("Incluir consultas de creacion de tablas?"));
FILE *f = fopen(F_Exportable, "r");
char cadena[1024];
```

```
void sql()
{
Funciones funciones;
int di, mi, ai, hi, msi;
int df, mf, af, hf, msf;
int intervalo = 0;
time_t inicio, fin;
struct tm tdI;
struct tm tdF;
di = mi = ai = hi = msi = 0;
df = mf = af = hf = msf = 0;
char *respuesta;
respuesta = leeCadena("Fecha y hora de inicio
 dia/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &di, &mi, &ai,
&hi, &msi);
respuesta = leeCadena("Fecha y hora de fin di
a/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &df, &mf, &af,
&hf, &msf);
tdI.tm_mday = di;
tdI.tm_mon = mi - 1;
tdI.tm_year = (ai > 1900) ? ai - 1900 : (ai <
 50) ? 100 + ai
: ai;
tdI.tm_hour = hi;
tdI.tm_min = msi;
tdF.tm_mday = df;
tdF.tm_mon = mf - 1;
tdF.tm_year = (af > 1900) ? af - 1900 : (af <
 50) ? 100 + af
: af;
tdF.tm_hour = hf;
tdF.tm_min = msf;
inicio = mktime(&tdI);
fin = mktime(&tdF);
funciones.volcadoSql(inicio, fin, confirmacion
("Incluir consultas de creacion de tablas?"));
FILE *f = fopen(F_Exportable, "r");
char cadena[1024];
```

```
bzero(cadena, 1024);
while (!feof(f))
{
fgets(cadena, 1000, f);
if (feof(f))
break;
escribe(cadena);
}
}
void xml()
{
Funciones funciones;
int di, mi, ai, hi, msi;
int df, mf, af, hf, msf;
int intervalo = 0;
time_t inicio, fin;
struct tm tdI;
struct tm tdF;
di = mi = ai = hi = msi = 0;
df = mf = af = hf = msf = 0;
char *respuesta;
respuesta = leeCadena("Fecha y hora de inicio
 dia/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &di, &mi, &ai,
&hi, &msi);
respuesta = leeCadena("Fecha y hora de fin di
a/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &df, &mf, &af,
&hf, &msf);
tdI.tm_mday = di;
tdI.tm_mon = mi - 1;
tdI.tm_year = (ai > 1900) ? ai - 1900 : (ai <
 50) ? 100 + ai
: ai;
tdI.tm_hour = hi;
tdI.tm_min = msi;
tdF.tm_mday = df;
tdF.tm_mon = mf - 1;
tdF.tm_year = (af > 1900) ? af - 1900 : (af <
 50) ? 100 + af
: af;
tdF.tm_hour = hf;
tdF.tm_min = msf;
inicio = mktime(&tdI);
fin = mktime(&tdF);
funciones.volcadoXml(inicio, fin, confirmacion
("Incluir configuracion?"));
FILE *f = fopen(F_Exportable, "r");
char cadena[1024];
bzero(cadena, 1024);
while (!feof(f))
{
fgets(cadena, 1000, f);
if (feof(f))
break;
escribe(cadena);
}
}
void yml()
{
Funciones funciones;
int di, mi, ai, hi, msi;
int df, mf, af, hf, msf;
```

```
bzero(cadena, 1024);
while (!feof(f))
{
fgets(cadena, 1000, f);
if (feof(f))
break;
escribe(cadena);
}
}
void xml()
{
Funciones funciones;
int di, mi, ai, hi, msi;
int df, mf, af, hf, msf;
int intervalo = 0;
time_t inicio, fin;
struct tm tdI;
struct tm tdF;
di = mi = ai = hi = msi = 0;
df = mf = af = hf = msf = 0;
char *respuesta;
respuesta = leeCadena("Fecha y hora de inicio
 dia/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &di, &mi, &ai,
&hi, &msi);
respuesta = leeCadena("Fecha y hora de fin di
a/mes/año-hora:minuto :");
if (strlen(respuesta) < 2)
return;
sscanf(respuesta, FormatoFecha, &df, &mf, &af,
&hf, &msf);
tdI.tm_mday = di;
tdI.tm_mon = mi - 1;
tdI.tm_year = (ai > 1900) ? ai - 1900 : (ai <
 50) ? 100 + ai
: ai;
tdI.tm_hour = hi;
tdI.tm_min = msi;
tdF.tm_mday = df;
tdF.tm_mon = mf - 1;
tdF.tm_year = (af > 1900) ? af - 1900 : (af <
 50) ? 100 + af
: af;
tdF.tm_hour = hf;
tdF.tm_min = msf;
inicio = mktime(&tdI);
fin = mktime(&tdF);
funciones.volcadoXml(inicio, fin, confirmacion
("Incluir configuracion?"));
FILE *f = fopen(F_Exportable, "r");
char cadena[1024];
bzero(cadena, 1024);
while (!feof(f))
{
fgets(cadena, 1000, f);
if (feof(f))
break;
escribe(cadena);
}
}
void yml()
{
Funciones funciones;
int di, mi, ai, hi, msi;
int df, mf, af, hf, msf;
```

```
817 int intervalo = 0;
818 time_t inicio, fin;
819 struct tm tdI;
820 struct tm tdF;
821 di = mi = ai = hi = msi = 0;
822 df = mf = af = hf = msf = 0;
823 char *respuesta;
824 respuesta = leeCadena("Fecha y hora de inicio
     dia/mes/año-hora:minuto :");
825 if (strlen(respuesta) < 2)
826 return;
827 sscanf(respuesta, FormatoFecha, &di, &mi, &ai,
    &hi, &msi);
828 respuesta = leeCadena("Fecha y hora de fin di
    a/mes/año-hora:minuto :");
829 if (strlen(respuesta) < 2)
830 return;
831 sscanf(respuesta, FormatoFecha, &df, &mf, &af,
    &hf, &msf);
832 tdI.tm_mday = di;
833 tdI.tm_mon = mi - 1;
834 tdI.tm_year = (ai > 1900) ? ai - 1900 : (ai <
     50) ? 100 + ai
835 : ai;
836 tdI.tm_hour = hi;
837 tdI.tm_min = msi;
838 tdF.tm_mday = df;
839 tdF.tm_mon = mf - 1;
840 tdF.tm_year = (af > 1900) ? af - 1900 : (af <
     50) ? 100 + af
841 : af;
842 tdF.tm_hour = hf;
843 tdF.tm_min = msf;
844 inicio = mktime(&tdI);
845 fin = mktime(&tdF);
846 funciones.volcadoYml(inicio, fin, confirmacion
    ("Incluir configuracion?"));
847 FILE *f = fopen(F_Exportable, "r");
848 char cadena[1024];
849 bzero(cadena, 1024);
850 while (!feof(f))
851 {
852 fgets(cadena, 1000, f);
853 if (feof(f))
854 break;
855 escribe(cadena);
856 }
857 }
858 void EscribeVersion()
859 {
860 Configuracion config;
861 char cadena[1024];
862 snprintf(cadena, 1024, "version:%3.2f\n", conf
    ig.version);
863 escribe(cadena);
864 }
865 void recalibrateMicByValue(float value)
866 {
867 Calibracion cal;
868 cal.leeCalibracion(0);
869 cal.dBRef += value;
870 cal.guardaCalibracion(0);
871 FILE *f = fopen("/tmp/lastCalValue", "w");
872 fprintf(f, "%f\n", value);
873 fclose(f);
874 };
```

```
628 int intervalo = 0;
629 time_t inicio, fin;
630 struct tm tdI;
631 struct tm tdF;
632 di = mi = ai = hi = msi = 0;
633 df = mf = af = hf = msf = 0;
634 char *respuesta;
635 respuesta = leeCadena("Fecha y hora de inicio
     dia/mes/año-hora:minuto :");
636 if (strlen(respuesta) < 2)
637 return;
638 sscanf(respuesta, FormatoFecha, &di, &mi, &ai,
    &hi, &msi);
639 respuesta = leeCadena("Fecha y hora de fin di
    a/mes/año-hora:minuto :");
640 if (strlen(respuesta) < 2)
641 return;
642 sscanf(respuesta, FormatoFecha, &df, &mf, &af,
    &hf, &msf);
643 tdI.tm_mday = di;
644 tdI.tm_mon = mi - 1;
645 tdI.tm_year = (ai > 1900) ? ai - 1900 : (ai <
     50) ? 100 + ai
646 : ai;
647 tdI.tm_hour = hi;
648 tdI.tm_min = msi;
649 tdF.tm_mday = df;
650 tdF.tm_mon = mf - 1;
651 tdF.tm_year = (af > 1900) ? af - 1900 : (af <
     50) ? 100 + af
652 : af;
653 tdF.tm_hour = hf;
654 tdF.tm_min = msf;
655 inicio = mktime(&tdI);
656 fin = mktime(&tdF);
657 funciones.volcadoYml(inicio, fin, confirmacion
    ("Incluir configuracion?"));
658 FILE *f = fopen(F_Exportable, "r");
659 char cadena[1024];
660 bzero(cadena, 1024);
661 while (!feof(f))
662 {
663 fgets(cadena, 1000, f);
664 if (feof(f))
665 break;
666 escribe(cadena);
667 }
668 }
669 void EscribeVersion()
670 {
671 Configuracion config;
672 char cadena[1024];
673 snprintf(cadena, 1024, "version:%s\n", config.
    version);
674 escribe(cadena);
675 }
676 void recalibrateByValue(float value, int calib
    rationIndex)
677 {
678 Calibracion cal;
679 cal.leeCalibracion(calibrationIndex);
680 cal.dBRef += value;
681 cal.guardaCalibracion(calibrationIndex);
682 FILE *f = fopen("/tmp/lastCalValue", "w");
683 fprintf(f, "%f\n", value);
684 fclose(f);
685 };
```

```
875 void showEq(int num)
876 {
877 Calibracion cal;
878 cal.leeCalibracion(num);
879 for (int i = 0; i < 8; i++)
880 {
881 printf("%.1f,", cal.equalization[i]);
882 }
883 puts(";");
884 };
885 void updateEq(char *params, int num)
886 {
887 Calibracion cal;
888 cal.leeCalibracion(num);
889 int l = strlen(params);
890 for (int i = 0; i < l; i++)
891 if (params[i] == ',')
892 params[i] = '\0';
893 char *cadena = params;
894 for (int i = 0; i < 8; i++)
895 {
896 cal.equalization[i] = atof(cadena);
897 l = strlen(cadena);
898 cadena += l + 1;
899 };
900 cal.guardaCalibracion(num);
901 }
902 int main(int argc, char **argv)
903 {
904 Configuracion configuracion;
905 identificado = configuracion.numeroDeSerie ==
    0;
906 #ifdef INET
907 //puerto=EsperaConexion(1036);//=Conecta("loca
    lhost",9999);
908 if (puerto < 1)
909 {
910 puts("-Error al conectar");
911 }
912 #else
913 #ifdef LOCAL
914 abreStdin();
915 #else
916 abrePuertoSerie();
917 #endif
918 #endif
919 if (argc == 2)
920 {
921 if (strcmp(argv[1], "calibralineas") == 0)
922 {
```

```
686 void showCalibrationEqualization(int calNum)
687 {
688 Calibracion cal;
689 cal.leeCalibracion(calNum);
690 for (int i = 0; i < Spectrum_BandCount; i++)
691 {
692 printf("%.1f,", cal.equalization[i]);
693 }
694 puts(";");
695 };
696 void showInternalCalibrationEqualization(int c
    alNum)
697 {
698 Calibracion cal;
699 cal.leeCalibracion(calNum);
700 for (int i = 0; i < Spectrum_BandCount; i++)
701 {
702 printf("%.1f,", cal.internalEqualization[i]);
703 }
704 puts(";");
705 };
706 void updateCalibrationEqualization(int calNum,
    char *equalization)
707 {
708 Calibracion cal;
709 cal.leeCalibracion(calNum);
710 leeArraySeparadoPorComas(equalization, cal.equ
    alization, Spectrum_BandCount);
711 cal.guardaCalibracion(calNum);
712 }
713 void updateInternalCalibrationEqualization(int
    calNum, char *equalization)
714 {
715 Calibracion cal;
716 cal.leeCalibracion(calNum);
717 leeArraySeparadoPorComas(equalization, cal.int
    ernalEqualization, Spectrum_BandCount);
718 cal.guardaCalibracion(calNum);
719 }
720 int main(int argc, char **argv)
721 {
722 Configuracion configuracion;
723 identificado = true; //configuracion.numeroDeS
    erie==0;
724 #ifdef INET
725 //puerto=EsperaConexion(1036);//=Conecta("loca
    lhost",9999);
726 if (puerto < 1)
727 {
728 puts("-Error al conectar");
729 }
730 #else
731 #ifdef LOCAL
732 abreStdin();
733 #else
734 abrePuertoSerie();
735 #endif
736 #endif
737 if (argc == 2)
738 {
739 if (strcmp(argv[1], "calibralineas") == 0)
740 {
```

```
923 test2(true, true);
924 return 0;
925 }
926 if (strcmp(argv[1], "miceq") == 0)
927 {
928 showEq(0);
929 return 0;
930 }
931 if (strcmp(argv[1], "lefteq") == 0)
932 {
933 showEq(1);
934 return 0;
935 }
936 if (strcmp(argv[1], "righteq") == 0)
937 {
938 showEq(2);
939 return 0;
940 }
941 }
942 if (argc > 2)
943 {
944 if (strcmp(argv[1], "mic") == 0)
945 {
946 recalibrateMicByValue(atof(argv[2]));
947 return 0;
948 }
949 if (strcmp(argv[1], "miceq") == 0)
950 {
951 puts("Cambiando equalizacion");
952 updateEq(argv[2], 0);
953 return 0;
954 }
955 if (strcmp(argv[1], "lefteq") == 0)
956 {
957 puts("Cambiando equalizacion");
```

```
741 test2(true, true);
742 return 0;
743 }
744 if (strcmp(argv[1], "miceq") == 0)
745 {
746 showCalibrationEqualization(0);
747 return 0;
748 }
749 if (strcmp(argv[1], "lefteq") == 0)
750 {
751 showCalibrationEqualization(1);
752 return 0;
753 }
754 if (strcmp(argv[1], "righteq") == 0)
755 {
756 showCalibrationEqualization(2);
757 return 1;
758 }
759 if (strcmp(argv[1], "imiceq") == 0)
760 {
761 showInternalCalibrationEqualization(0);
762 return 0;
763 }
764 if (strcmp(argv[1], "ilefteq") == 0)
765 {
766 showInternalCalibrationEqualization(1);
767 return 0;
768 }
769 if (strcmp(argv[1], "irighteq") == 0)
770 {
771 showInternalCalibrationEqualization(2);
772 return 1;
773 }
774 }
775 if (argc > 2)
776 {
777 if (strcmp(argv[1], "miceq") == 0)
778 {
779 updateCalibrationEqualization(0, argv[2]);
780 return 0;
781 }
782 if (strcmp(argv[1], "lefteq") == 0)
783 {
784 updateCalibrationEqualization(1, argv[2]);
785 return 0;
786 }
787 if (strcmp(argv[1], "righteq") == 0)
788 {
789 updateCalibrationEqualization(2, argv[2]);
790 return 1;
791 }
792 if (strcmp(argv[1], "imiceq") == 0)
793 {
794 updateInternalCalibrationEqualization(0, argv
    [2]);
795 return 0;
796 }
797 if (strcmp(argv[1], "ilefteq") == 0)
798 {
799 updateInternalCalibrationEqualization(1, argv
    [2]);
800 return 0;
801 }
802 if (strcmp(argv[1], "irighteq") == 0)
803 {
804 updateInternalCalibrationEqualization(2, argv
```

```
updateEq(argv[2], 1);
return 0;
}
if (strcmp(argv[1], "righteq") == 0)
{
puts("Cambiando equalizacion");
updateEq(argv[2], 2);
return 0;
}
}
char *cadena;
int watchDogCtr = 0;
while (1)
{
watchDogCtr++;
if (watchDogCtr > 1000)
{
break;
}
escribe("#lm7>");
cadena = leeCadena();
if (strcmp(cadena, "datos") == 0)
datos(); // Hecho
if (strcmp(cadena, "update") == 0)
update(); // Hecho
if (strcmp(cadena, "configuracion") == 0)
muestraConfiguracion(); // Hecho
if (strcmp(cadena, "configura") == 0)
configura(); // Hecho
if (strcmp(cadena, "activa") == 0)
activa(); // Hecho
if (strcmp(cadena, "concha") == 0)
concha(); // Hecho
if (strcmp(cadena, "concharemota") == 0)
conchaRemota(); // Hecho
if (strcmp(cadena, "identifica") == 0)
identifica(); // Hecho
if (strcmp(cadena, "calibra") == 0)
calibra(); // Hecho
if (strcmp(cadena, "prepara") == 0)
prepara(); // Hecho
if (strcmp(cadena, "operativo") == 0)
operativo(); // Hecho
if (strcmp(cadena, "test") == 0)
test2(); // Hecho
if (strcmp(cadena, "calibracion") == 0)
calibracion(-1); // Hecho
if (strcmp(cadena, "salir") == 0)
identificado = false; // Hecho
if (strcmp(cadena, "ayuda") == 0)
ayuda(); // Hecho
if (strcmp(cadena, "sql") == 0)
sql();
if (strcmp(cadena, "xml") == 0)
xml();
if (strcmp(cadena, "yml") == 0)
yml();
if (strcmp(cadena, "id") == 0)
```

```
[2]);
return 1;
}
if (strcmp(argv[1], "mic") == 0)
{
recalibrateByValue(atof(argv[2]), 0);
return 0;
}
if (strcmp(argv[1], "left") == 0)
{
recalibrateByValue(atof(argv[2]), 1);
return 0;
}
if (strcmp(argv[1], "right") == 0)
{
recalibrateByValue(atof(argv[2]), 2);
return 0;
}
}
char *cadena;

while (1)
{

escribe("#lm7>");
cadena = leeCadena();
if (strcmp(cadena, "datos") == 0)
datos(); // Hecho
if (strcmp(cadena, "update") == 0)
update(); // Hecho
if (strcmp(cadena, "configuracion") == 0)
muestraConfiguracion(); // Hecho
if (strcmp(cadena, "configura") == 0)
configura(); // Hecho
if (strcmp(cadena, "activa") == 0)
activa(); // Hecho
if (strcmp(cadena, "concha") == 0)
concha(); // Hecho
if (strcmp(cadena, "concharemota") == 0)
conchaRemota(); // Hecho
if (strcmp(cadena, "identifica") == 0)
identifica(); // Hecho
if (strcmp(cadena, "calibra") == 0)
calibra(); // Hecho
if (strcmp(cadena, "prepara") == 0)
prepara(); // Hecho

if (strcmp(cadena, "test") == 0)
test2(); // Hecho
if (strcmp(cadena, "calibracion") == 0)
calibracion(-1); // Hecho
if (strcmp(cadena, "salir") == 0)
identificado = false; // Hecho
if (strcmp(cadena, "ayuda") == 0)
ayuda(); // Hecho
if (strcmp(cadena, "sql") == 0)
sql();
if (strcmp(cadena, "xml") == 0)
xml();
if (strcmp(cadena, "yml") == 0)
yml();
if (strcmp(cadena, "id") == 0)
```

```
1016 EscribeIdentificacion();
1017 if (strcmp(cadena, "version") == 0)
1018 EscribeVersion();
1019 if (strcmp(cadena, "identificame") == 0)
1020 Identificame();
1021 if (strcmp(cadena, "identificate") == 0)
1022 Identificate();
1023 if (strcmp(cadena, "calibralineas") == 0)
1024 {
1025 if (!identificado)
1026 {
1027 escribe("Debe identificarse\n");
1028 }
1029 else
1030 {
1031 test2(true); // Hecho
1032 }
1033 }
1034 }
1035 }
```

```
863 EscribeIdentificacion();
864 if (strcmp(cadena, "version") == 0)
865 EscribeVersion();
866 if (strcmp(cadena, "calibralineas") == 0)
867 {
868 if (!identificado)
869 {
870 escribe("Debe identificarse\n");
871 }
872 else
873 {
874 test2(true); // Hecho
875 }
876 }
877 }
878 }
```