Universidad de Oviedo

Anexo del Trabajo Fin de Máster realizado por

MARCOS RODRÍGUEZ CASTAÑO

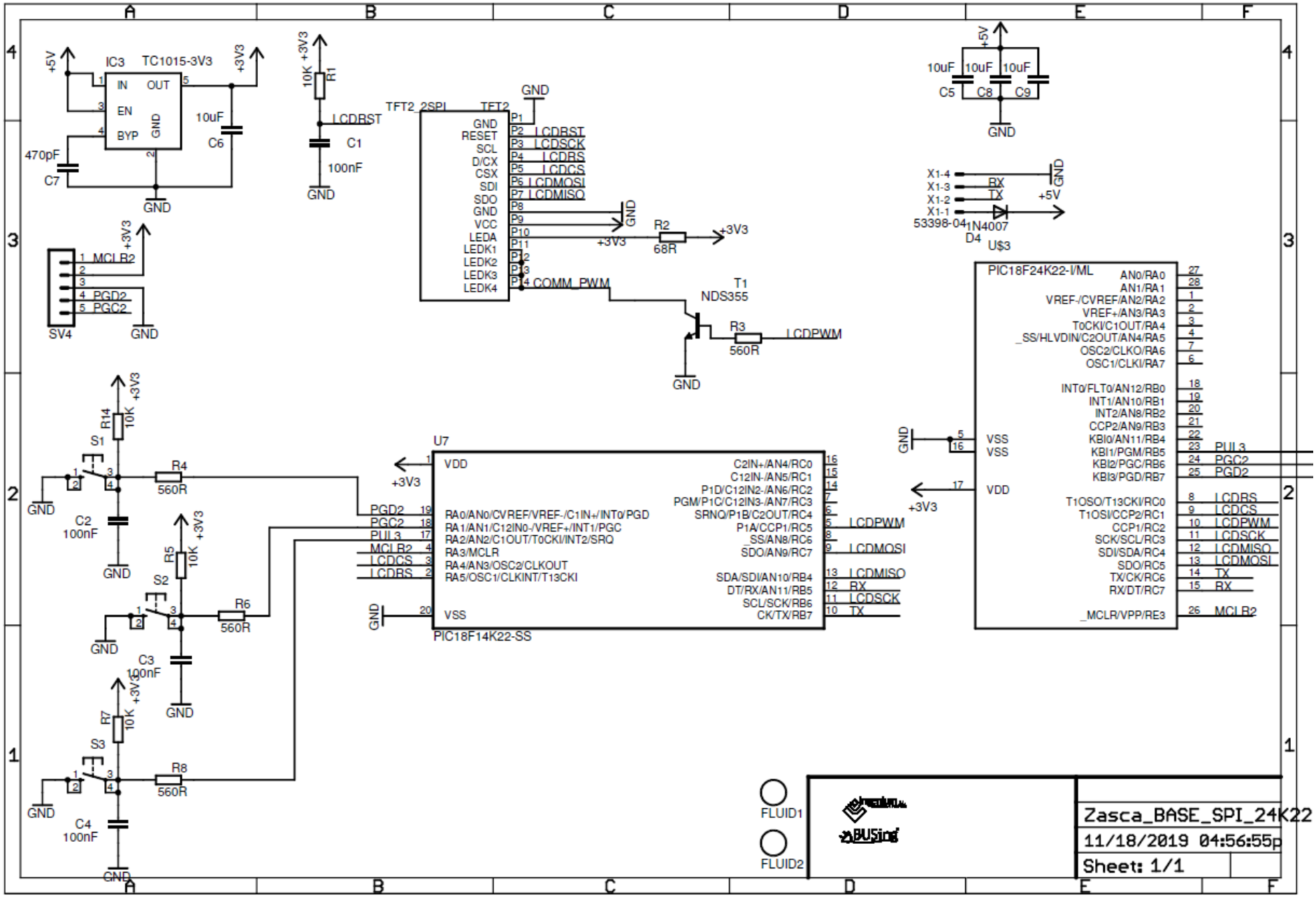
para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

Mejora y optimización de la pasarela DALI GW611100 de BES KNX (Ingenium)

JULIO 2020

Índice


IC3 TC1015-3V3
IN OUT
EN
BYP GND
+5V
+3V3
10uF C6
470pF C7
GND
10K +3V3 R1
LCDRST
C1 100nF
GND
TFT2 2SPI TFT2
GND RESET SCL D/CX CSX SDI SDO GND VCC LEDA LEDK1 LEDK2 LEDK3 LEDK4
P1 GND
P2 LCDRST
P3 LCDSCK
P4 LCDRS
P5 LCDCS
P6 LCDMOSI
P7 LCDMISO
P8
P9
P10
P11
P12
P13
P14 COMM_PWM
GND
+3V3
R2 68R
+3V3
T1 NDS355
R3 560R
LCDPWM
GND
10uF 10uF 10uF
C5 C8 C9
+5V
GND
X1-4
X1-3 RX
X1-2 TX
X1-1
53398-04
1N4007 D4
+5V
GND
U$3
SV4
1 MCLR2
2
3
4 PGD2
5 PGC2
+3V3
GND
R14 10K +3V3
S1
R4 560R
C2 100nF
GND
R5 10K +3V3
S2
R6 560R
C3 100nF
GND
R7 10K +3V3
S3
R8 560R
C4 100nF
GND
U7
VDD
+3V3
RA0/AN0/CVREF/VREF-/C1IN+/INT0/PGD
RA1/AN1/C12IN0-/VREF+/INT1/PGC
RA2/AN2/C1OUT/T0CKI/INT2/SRQ
RA3/MCLR
RA4/AN3/OSC2/CLKOUT
RA5/OSC1/CLKINT/T13CKI
VSS
GND
C2IN+/AN4/RC0
C12IN-/AN5/RC1
P1D/C12IN2-/AN6/RC2
PGM/P1C/C12IN3-/AN7/RC3
SRNQ/P1B/C2OUT/RC4
P1A/CCP1/RC5
SS/AN8/RC6
SDO/AN9/RC7
SDA/SDI/AN10/RB4
DT/RX/AN11/RB5
SCL/SCK/RB6
CK/TX/RB7
PGD2
PGC2
PUI3
MCLR2
LCDCS
LCDRS
LCDPWM
LCDMOSI
LCDMISO
RX
LCDSCK
TX
PIC18F14K22-SS
PIC18F24K22-I/ML
AN0/RA0
AN1/RA1
VREF-/CVREF/AN2/RA2
VREF+/AN3/RA3
T0CKI/C1OUT/RA4
_SS/HLVDIN/C2OUT/AN4/RA5
OSC2/CLKO/RA6
OSC1/CLKI/RA7
INT0/FLT0/AN12/RB0
INT1/AN10/RB1
INT2/AN8/RB2
CCP2/AN9/RB3
KBI0/AN11/RB4
KBI1/PGM/RB5
KBI2/PGC/RB6
KBI3/PGD/RB7
T1OSO/T13CKI/RC0
T1OSI/CCP2/RC1
CCP1/RC2
SCK/SCL/RC3
SDI/SDA/RC4
SDO/RC5
TX/CK/RC6
RX/DT/RC7
_MCLR/VPP/RE3
VSS
VSS
VDD
GND
+3V3
PUI3
PGC2
PGD2
LCDRS
LCDCS
LCDPWM
LCDSCK
LCDMISO
LCDMOSI
TX
RX
MCLR2
FLUID1
FLUID2
Zasca_BASE_SPI_24K22
11/18/2019 04:56:55p
Sheet: 1/1

decimal	hex	binary	sent twice	return frame	action	Rayzig usage and/or explanation
		YAAA AAA0 XXXX XXXX	N	N	direct power control (level = X..X)	Used for Level and ON actions & DALI command tab
0	00	YAAA AAA1 0000 0000	N	N	off	Used for OFF command
1	01	YAAA AAA1 0000 0001	N	N	up	DALI: config tab
2	02	YAAA AAA1 0000 0010	N	N	down	DALI: config tab
3	03	YAAA AAA1 0000 0011	N	N	step up	DALI: config tab
4	04	YAAA AAA1 0000 0100	N	N	step down	DALI: config tab
5	05	AAA AAA1 0000 0101	N	N	recall max level	DALI: config tab
6	06	YAAA AAA1 0000 0110	N	N	recall min level	DALI: config tab
7	07	YAAA AAA1 0000 0111	N	N	step down and off	DALI: config tab
8	08	YAAA AAA1 0000 1000	N	N	on and step up	DALI: config tab
9-15	09-0F	YAAA AAA1 0000 1XXX			reserved	
16-31	10-1F	YAAA AAA1 0001 XXXX	N	N	go to scene	DALI config tab and ON action mapped to scene
32	20	YAAA AAA1 0010 0000	Y	N	reset	DALI: config and list tabs -
33	21	YAAA AAA1 0010 0001	Y	N	store actual level in the DTR	
34-41	22-29	YAAA AAA1 0010 XXXX			reserved	
42	2A	YAAA AAA1 0010 1010	Y	N	store the DTR as max level	(Broadcast Level > DTR) then DTR to max
43	2B	YAAA AAA1 0010 1011	Y	N	store the DTR as min level	(Broadcast Level > DTR) then DTR. DTR to min
44	2C	YAAA AAA1 0010 1100	Y	N	store the DTR as sys fail level	(Broadcast Level > DTR) then DTR to sys fail
45	2D	YAAA AAA1 0010 1101	Y	N	store the DTR as power on level	(Broadcast Level > DTR) then DTR to power on
46	2E	YAAA AAA1 0010 1110	Y	N	store the DTR as fade time	(Broadcast Level > DTR) then DTR to fade time
47	2F	YAAA AAA1 0010 1111	Y	N	store the DTR as fade rate	(Broadcast Level > DTR) then DTR to fade rate
48-63	30-3F	YAAA AAA1 0011 XXXX			reserved	
64-79	40-4F	YAAA AAA1 0100 XXXX	Y	N	store the DTR as scene	(Broadcast Level > DTR) then DTR to scene XXXX
80-95	50-5F	YAAA AAA1 0101 XXXX	Y	N	remove from scene	Remove from scene XXXX
96-111	60-6F	YAAA AAA1 0110 XXXX	Y	N	add to group	Luminaire AAAAAA to group XXXX
112-127	70-7F	YAAA AAA1 0111 XXXX	Y	N	remove from group	Remove luminaire AAAAAA from group XXXX
128	80	YAAA AAA1 1000 0000	Y	N	store DTR as short address	DALI: config and list tabs -
129-143	81-8F	YAAA AAA1 1000 XXXX			reserved	
144	90	YAAA AAA1 1001 0000	N	Y	query status	DALI: config tab
145	91	YAAA AAA1 1001 0001	N	Y	query control gear	DALI: config tab
146	92	YAAA AAA1 1001 0010	N	Y	query lamp failure	DALI: config tab
147	93	YAAA AAA1 1001 0011	N	Y	query lamp power on	DALI: config tab
148	94	YAAA AAA1 1001 0100	N	Y	query limit error	DALI: config tab
149	95	YAAA AAA1 1001 0101	N	Y	query reset state	DALI: config tab
150	96	YAAA AAA1 1001 0110	N	Y	query missing short address	DALI: config tab
151	97	YAAA AAA1 1001 011	N	Y	query version number	DALI: config tab
152	98	YAAA AAA1 1001 1000	N	Y	query content DTR	
153	99	YAAA AAA1 1001 1001	N	Y	query device type	DALI: config tab
154	9A	YAAA AAA1 1001 1010	N	Y	query physical minimum level	DALI: config tab
155	9B	YAAA AAA1 1001 1011	N	Y	query power failure	DALI: config tab
156-159	9C-9F	YAAA AAA1 1001 11XX			reserved	
160	A0	YAAA AAA1 1010 0000	N	Y	query actual level	DALI: config tab
161	A1	YAAA AAA1 1010 0001	N	Y	query max level	DALI: config tab
162	A2	YAAA AAA1 1010 0010	N	Y	query min level	DALI: config tab
163	A3	YAAA AAA1 1010 0011	N	Y	query power on level	DALI: config tab
164	A4	YAAA AAA1 1010 0100	N	Y	query system failure level	DALI: config tab
165	A5	YAAA AAA1 1010 0101	N	Y	query fade time / fade rate	DALI: config tab
166-175	A6-AF	YAAA AAA1 1010 XXXX			reserved	
176-191	B0-BF	YAAA AAA1 1011 XXXX	N	Y	query scene level (scenes 0-15)	DALI config: command and list tabs
192	C0	YAAA AAA1 1100 0000	N	Y	query groups 0-7	DALI config: command and list tabs
193	C1	YAAA AAA1 1100 0001	N	Y	query groups 8-15	DALI config: command and list tabs
194	C2	YAAA AAA1 1100 0010	N	Y	query random address (H)	DALI config: command and list tabs
195	C3	YAAA AAA1 1100 0011	N	Y	query random address (M)	DALI config: command and list tabs
196	C4	YAAA AAA1 1100 0100	N	Y	query random address (L)	DALI config: command and list tabs
197-223	C5-DF	YAAA AAA1 110X XXXX			reserved	
224-255	E0-FF	YAAA AAA1 11XX XXXX			application extended commands	

hex code	binary	sent twice	return frame	action	Rayzig usage and/or explanation
A1	1010 0001 0000 0000	N	N	terminate	Terminates the 30 min initialisation
A3	1010 0011 XXXX XXXX	N	N	data transfer register (DTR)	X..X to DTR of all luminaires (followed by set commands)
A5	1010 0101 XXXX XXXX	Y	N	initialise	Starts a 30 min. window for address find/set
A7	1010 0111 0000 0000	Y	N	randomise	All luminaires generate a new random 24 bit address
A9	1010 1001 0000 0000	N	Y	compare	Compares 24 addresses with 24 bit search address
AB	1010 1011 0000 0000	N	N	withdraw	Remove address-matched luminaire from search
AD	1010 1101 0000 0000			reserved	
AF	1010 1111 0000 0000			reserved	
B1	1011 0001 HHHH HHHH	N	N	SEARCHADDRH	Send search address high byte to all luminaires
B3	1011 0011 MMMM MMMM	N	N	SEARCHADDRM	Send search address middle byte to all luminaires
B5	1011 0101 LLLL LLLL	N	N	SEARCHADDRL	Send search address low byte to all luminaires
B7	1011 0111 0AAA AAA1	N	N	program short address	Program add AAAAAA to 24 bit matching luminaire
B9	1011 1001 0AAA AAA1	N	Y	verify short address	Not used at present
BB	1011 1011 0000 0000	N	Y	query short address	Read short address of 24 bit matching luminaire
BD	1011 1101 0000 0000	N	N	physical selection	
BF	1011 1111 XXXX XXXX			reserved	
C1	1100 0001 XXXX XXXX	N	N	enable device type x	
	110X XXX1 XXXX XXXX			reserved	

MAIN

Configuracion inicial

```
configuracion_PIC();
    inicbuff();
    Init_LCD();
    aux_menu=1;
    pintaMenu(7000);

    int g, parp=0;

    for(int p_nodos=0;p_nodos<64;p_nodos++){
        nodos[p_nodos]=0;
        lista_nodos_activos[p_nodos]=0;
    }
    enable_interrupts(INT_RDA);
```

WHILE 1

Condicionales

```
if(salvapantallas >= 30){
    salvapantallas=0;
    pintaMenu(7000);
}

efecto de parpadeo durante la pantalla de espera
```

M.E. main

envio_2==1

envio_2==0

Envio de 1 trama

envio == 0

putc (0x00)

envio == 1

aux_menu == 30

aux_menu != 30

Escaneo

```
putc (tx[0])
delay_us(1)
putc (tx[1])
delay_us(1)
putc (tx[2])
delay_us(1)
putc (tx[3])
```

rx[0]==1 && rx[2]==0xFF

hit<64

nodos[x] = 1
hit++

hit>=64

rx[0]==1 && rx[2]==0x00

hit<64

nodos[x] = 1
hit++

hit>=64

ON EXIT: pintaMenu(1), envio = 0

Envio normal

```
putc (tx[0])
delay_us(1)
putc (tx[1])
delay_us(1)
putc (tx[2])
delay_us(1)
putc (tx[3])
```

aux_menu == 4125 && rx[0] == 1

aux_menu == 4135 && rx[0] == 1

aux_menu == 5135 && rx[0] == 1

```
scroll=0;
aux_scroll_1=0;
aux_scroll_3=0;
aux_scroll_4=0;
pintaMenu(menu_anterior);
```

ON EXIT: envio = 0

Envio de 2 tramas

segunda_parte == 0

```
putc (tx[0])
delay_us(1)
putc (tx[1])
delay_us(1)
putc (tx[2])
delay_us(1)
putc (tx[3])
```

rx[0] == 1

segunda_parte = 1

else

aux_menu == 4110 && rx[2] == 0xFF

g<8

```
if ((rx[3]&0x1<<g)!=0)
    v_grupos[g]=1
else
    v_grupos[g]=0;

g++
```

g>=8

segunda_parte == 1

```
putc (tx_2[0])
delay_us(1)
putc (tx_2[1])
delay_us(1)
putc (tx_2[2])
delay_us(1)
putc (tx_2[3])
```

rx[0] == 1

segunda_parte = 0

else

aux_menu == 4110 && rx[2] == 0xFF

g<8

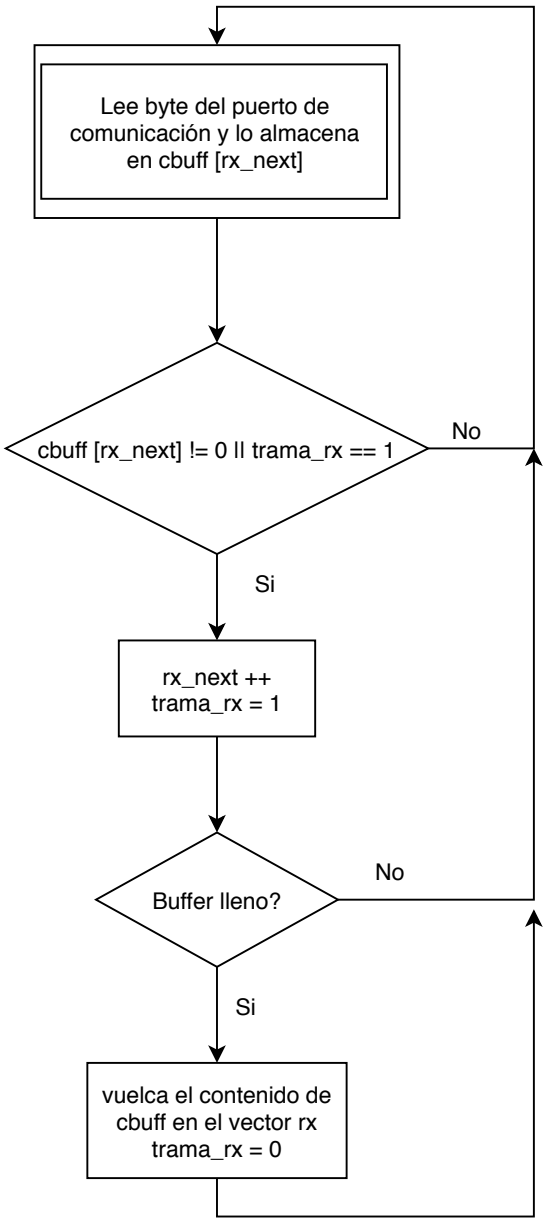
```
if ((rx[3]&0x1<<g)!=0)
    v_grupos[g+8]=1
else
    v_grupos[g+8]=0;

g++
```

else

aux_menu == 4155|| aux_menu == 4160 || aux_menu == 4170

[g>=8 && (aux_menu == 4155|| aux_menu == 4160 || aux_menu == 4170)]

envio_2 = 0


Lee byte del puerto de comunicación y lo almacena en cbuff [rx_next]
cbuff [rx_next] != 0 ll trama_rx == 1
No
Si
rx_next ++
trama_rx = 1
Buffer lleno?
No
Si
vuelca el contenido de cbuff en el vector rx
trama_rx = 0

Interrupt port B

/ON ENTRY: salvapantallas = 0, guardar estado de los pines 5, 6 y 7 del puerto B
RB5_actual = bit_test(port_B,5); //boton arriba
RB6_actual = bit_test(port_B,6); //boton OK
RB7_actual = bit_test(port_B,7); //boton abajo

[RB5_actual==0 && RB5_anterior==1]

[RB7_actual==0 && RB7_anterior==1]

[RB6_actual==0 && RB6_anterior==1]

Flanco en RB5

/ON ENTRY: contador_menu++;
del_cursor=pos_cursor;
pos_cursor=pos_cursor-48;

[aux_menu == 4170] valor_escena = valor_escena - 5;
[aux_menu == 4160] valor_inf = valor_inf - 1;
[aux_menu == 4155] valor_sup = valor_sup - 5;
else

[condicion_menu == 1] max_opciones = 4
[condicion_menu == 0] max_opciones = 1

else efecto de mover flecha hacia abajo
[contador_menu > max_opciones] contador_menu = max_opciones;

else
scroll++;
if(aux_menu==1 || aux_menu==410) Clear_LCD(0x0000);
if(aux_menu == 40) aux_scroll=scroll;
if(aux_menu == 4120) aux_scroll_1=scroll;
if(aux_menu == 50) aux_scroll_2=scroll;
if(aux_menu == 5130) aux_scroll_3=scroll;
if(aux_menu == 4130 || aux_menu == 4165) aux_scroll_4=scroll;

[contador_menu+scroll > opciones_menu] pos_cursor = 0

En este estado entra cuando se acciona el pulsador de abajo (selección de opción inferior)

Flanco en RB7

/ON ENTRY: contador_menu--;
del_cursor=pos_cursor;
pos_cursor=pos_cursor+48;

[aux_menu == 4170] valor_escena = valor_escena + 5;
[aux_menu == 4160] valor_inf = valor_inf + 5;
[aux_menu == 4155] valor_sup = valor_sup + 5;
else

[condicion_menu == 1] max_opciones = 4
[condicion_menu == 0] max_opciones = 1

else efecto de mover flecha hacia arriba
[contador_menu < 1] contador_menu = 1;

else
scroll--;
if(aux_menu==1 || aux_menu==410) Clear_LCD(0x0000);
if(aux_menu == 40) aux_scroll=scroll;
if(aux_menu == 4120) aux_scroll_1=scroll;
if(aux_menu == 50) aux_scroll_2=scroll;
if(aux_menu == 5130) aux_scroll_3=scroll;
if(aux_menu == 4130 || aux_menu == 4165) aux_scroll_4=scroll;

[contador_menu+scroll > 1] pos_cursor = 144

En este estado entra cuando se acciona el pulsador de arriba (selección de opción superior)

Flanco en RB6

/ON ENTRY: pos_menu = contador_menu + scroll;
actual = pos_menu+aux_menu;
envio=0;
envio_2=0;
botonOK=1;

[(pos_menu >= opciones_menu && aux_menu != 1) || aux_menu == 4140 || aux_menu == 20 || aux_menu==7000] scroll=0; Clear_LCD(0x0000); pintaMenu(menu_anterior);

else

[aux_menu==40] scroll=0; Clear_LCD(0x0000); pintaMenu(410);
[bloqueo && (actual == 411 || actual == 412 || actual == 413 || actual == 4153)]
[aux_menu == 4120 && !bloqueo] if(v_grupos[pos_menu-1] == 0) v_grupos[pos_menu-1] = 1; else v_grupos[pos_menu-1] = 0; scroll=0; pintaMenu(4125);
[aux_menu == 4130 && !bloqueo] scroll=0; pintaMenu(4135);
[aux_menu == 4165] aux_scroll_4=0; Clear_LCD(0x0000); scroll=0; pintaMenu(4170);
[aux_menu == 50] scroll=0; pintaMenu(510);
[actual == 511] scroll=0; Clear_LCD(0x0000); pintaMenu(4140);
[actual == 512] scroll=0; pintaMenu(4150);
[aux_menu == 5130] if(v_nodos[pos_menu-1] == 0) v_nodos[pos_menu-1] = 1; else v_nodos[pos_menu-1] = 0; scroll=0; Clear_LCD(0x0000); pintaMenu(5135);
[actual == 61] write_eeprom(0,0); pintaMenu(60);
[actual == 62] write_eeprom(0,1); pintaMenu(60);

else
if(aux_menu == 1) aux_menu=0;
aux_menu=aux_menu+scroll+contador_menu;
aux_menu=aux_menu*10;
scroll=0;
if(aux_menu==40) aux_scroll=0;
if(aux_menu==4120) aux_scroll_1=0;
if(aux_menu==4130) aux_scroll_4=0;
if(aux_menu==50) aux_scroll_2=0;
if(aux_menu==5130) aux_scroll_3=0;
Clear_LCD(0x0000);
pintaMenu(aux_menu);

contador_menu = 1;
pos_cursor=144;
colocar cursor al inicio del menu
botonOK=0;

En este estado entra cuando se acciona el pulsador central (confirmación de selección)

/ON EXIT: guardar estado de los pines 5, 6 y 7 del puerto B, limpiar flag de interrupción
RB5_anterior = RB5_actual;
RB6_anterior = RB6_actual;
RB7_anterior = RB7_actual;

aux_menu == 1

MENU

N INSTALACION
EXTENDER
DETECTAR
NODOS
GRUPOS
IDIOMAS

aux_menu == 10

ESPERE

aux_menu == 20

ESPERE

aux_menu == 30

ESPERE

aux_menu == 40

NODOS

ELIJA NODO

AA

aux_menu == 410

NODO AA

SOLICITAR GRUPOS
ASIG GRUPOS
ASIG ESCENA
LOCALIZAR
CONFIG

aux_menu == 4110

ESPERE

aux_menu == 4120

GRUPOS

ELIJA GRUPO

AA

aux_menu == 4120

NODOS AA

ELIJA GRUPO

AA

aux_menu == 4125

ESPERE

aux_menu == 4130

NODO AA

ELIJA ESCENA

AA

aux_menu == 4135

ESPERE

aux_menu == 510

ROTULO

CONFIG
LOCALIZAR
ASIG NODO

aux_menu == 4150

ROTULO

LIM SUPERIOR
LIM INFERIOR
VALOR ESCENA

aux_menu == 4140

LOCALIZANDO

aux_menu == 510

IDIOMAS

ES - ES
EN - EN

aux_menu == 5130

NODOS

ELIJA NODO

AA

aux_menu == 5135

ESPERE

aux_menu == 4155

ROTULO

LIM SUPERIOR

+

000

-

aux_menu == 4160

ROTULO

LIM INFERIOR

+

000

-

aux_menu == 4165

NODO AA

ELIJA ESCENA

AA

aux_menu == 4170

ROTULO

VALOR ESCENA

+

000

-

En esta ventana se entra desde cualquier otra, excepto desde las de espera, después de un minuto de inactividad, y se regresa a la misma accionando el pulsador central.

aux_menu == 7000

GW611100
PASARELA KNX-DALI

ingenium *
Bes
www.besknx.com

default

ERROR

En caso de entrar en un estado no contemplado, se entra en este estado, con lo que se detienen las transmisiones, para continuar se pulsa el botón central volviendo al menú inicial

MENÚ DALIK

/ON ENTRY: envio=0, envio_2=0, flag_espera=0;

MENU

Clear_LCD(0x0000);
menu_anterior = 1;
aux_menu = 1;
opciones_menu = 6;
condicion_menu=1;

[aux_menu == 10]

aux_menu = 10;
menu_anterior = 1;
condicion_menu=0;
pintaEspera();
tx[0]=EXTENDIDO_1;
tx[2]=0x12;
tx[3]=0x34;
envio=1;

[aux_menu == 20]

aux_menu = 20;
menu_anterior = 1;
condicion_menu=0;
pintaEspera();
tx[0]=REPETIDO;
tx[2]=0xA5;
tx[3]=0x00;
envio=1;

[aux_menu == 30]

aux_menu = 30;
menu_anterior = 1;
condicion_menu=0;
pintaEspera();
tx[0]=NORMAL;
tx[3]=0X99;
envio=1;

[aux_menu == 40]

ELECCION DE NODO

menu_anterior = 1;
condicion_menu=0;
aux_menu = 40;
salto_linea=0;
tipo_dir=0;
for(int i=0;i<16;i++)
v_grupos[i]=0;
opciones_menu = salto_linea+2;

MENU DE NODO XX

condicion_menu=1;
if(aux_scroll == salto_linea && aux_menu == 40){
bloqueo=1;
nodo = 0xFF;
}
else if(aux_scroll != salto_linea && aux_menu == 40){
bloqueo=0;
nodo_real = lista_nodos_activos[aux_scroll];
nodo = 2*nodo_real+1;
}
menu_anterior = 40;
aux_menu = 410;

[aux_menu == 4110]

aux_menu = 4110;
menu_anterior = 410;
condicion_menu=0;
opciones_menu = 1;
pintaEspera(); tx_2[0]=NORMAL;
tx[0]=NORMAL; tx_2[2]=nodo;
tx[2]=nodo; tx_2[3]=0xC1;
tx[3]=0xC0; envio_2=1;

[aux_menu == 4120]

ELECCIÓN DE GRUPO

if(aux_menu == 4125)
Clear_LCD(0x0000);
condicion_menu=0;
aux_menu = 4120;
pintaGrupos();
pintaRotuloNodos();
menu_anterior = 410;
opciones_menu = 17;
tx[0]=REPETIDO;
tx[2]=nodo;

aux_menu = 4125;
menu_anterior = 4120;
opciones_menu=1;
condicion_menu=0;
if(v_grupos[aux_scroll_1] == 1)
tx[3]=0x60+(aux_scroll_1);
else if(v_grupos[aux_scroll_1] == 0)
tx[3]=0x70+(aux_scroll_1);
envio = 1;
pintaEspera();

[aux_menu == 4130]

ELECCIÓN DE ESCENA

if(aux_menu == 4135)
Clear_LCD(0x0000);
condicion_menu=0;
aux_menu = 4130;
menu_anterior = 410;
opciones_menu=17;
pintaRotuloNodos();
pintaEscenas();

aux_menu = 4135;
menu_anterior = 4130;
condicion_menu=0;
tx[0]=NORMAL;
tx[2]=nodo;
tx[3]=0x10+aux_scroll_4;
envio=1;
pintaEspera();

[aux_menu == 4140]

LOCALIZANDO

condicion_menu=0;
aux_menu = 4140;
opciones_menu = 1;
if(tipo_dir==0){
menu_anterior = 410;
dir=nodo;
}
else{
menu_anterior = 510;
dir=grupo;
} tx_2[0]=NORMAL;
tx[0]=NORMAL; tx_2[2]=dir;
tx[2]=dir; tx_2[3]=0x05;
tx[3]=0x00; envio_2=1;

[aux_menu == 4150]

CONFIGURACIÓN

Clear_LCD(0x0000);
condicion_menu=1;
pintaRotuloNodos();
if(tipo_dir==0){
menu_anterior = 410;
dir=nodo;
}
else{
menu_anterior = 510;
dir=grupo;
}
aux_menu = 4150;
opciones_menu = 4;

[aux_menu == 60]

ELECCION DE GRUPO

condicion_menu=0;
menu_anterior = 1;
aux_menu = 50;
salto_linea=0;
opciones_menu=17;
tipo_dir=1;
pintaRotuloNodos();
for(int k=0;k<64;k++)
v_nodos[k]=0;
pintaGrupos();

MENU DE GRUPO XX

Clear_LCD(0x0000);
condicion_menu=1;
aux_menu = 510;
pintaRotuloNodos();
grupo = 2*aux_scroll_2+129;
menu_anterior = 50;
opciones_menu = 4;
pintaRetorno(0,8,3);

[aux_menu == 4150]

[aux_menu == 4140]

[aux_menu == 5130]

ELECCIÓN DE NODO

condicion_menu=0;
if(aux_menu == 5135)
Clear_LCD(0x0000);
menu_anterior = 510;
aux_menu = 5130;
salto_linea=0;
pintaNodosActivos();
pintaRotuloNodos();
opciones_menu = salto_linea+1;

[aux_menu == 5135]

condicion_menu=0;
aux_menu = 5135;
menu_anterior = 5130;

tx[0]=REPETIDO;
tx[2]=2*(lista_nodos_activos[aux_scroll_3])+1;
if(v_nodos[aux_scroll_3] == 1)
tx[3]=0x60+(aux_scroll_2);
else if(v_nodos[aux_scroll_3] == 0)
tx[3]=0x70+(aux_scroll_2);
envio = 1;
pintaEspera();

[aux_menu == 4155]

LIMITE SUPERIOR

condicion_menu=0;
menu_anterior = 4150;
aux_menu = 4155;
pintaRotuloNodos();
segunda_parte = 0;
if(valor_sup>100) valor_sup=100;
else if(valor_sup<0) valor_sup=0;
pintaNivel(valor_sup);
aux_iluminacion = ((unsigned int16)valor_sup*254)/100;

tx[0]=NORMAL; tx_2[0]=REPETIDO;
tx[2]=0xA3; tx_2[2]=dir;
tx[3]=aux_iluminacion; tx_2[3]=0x2A;
envio_2=1;

[aux_menu == 4160]

LIMITE INFERIOR

condicion_menu=0;
menu_anterior = 4150;
aux_menu = 4160;
pintaRotuloNodos();
segunda_parte = 0;
if(valor_inf>100) valor_inf=100;
else if(valor_inf<0) valor_inf=0;
pintaNivel(valor_inf);
aux_iluminacion = ((unsigned int16)valor_inf*254)/100;

tx[0]=NORMAL; tx_2[0]=REPETIDO;
tx[2]=0xA3; tx_2[2]=dir;
tx[3]=aux_iluminacion; tx_2[3]=0x2B;
envio_2=1;

[aux_menu == 4165]

ELECCIÓN DE ESCENA

icondicion_menu=0;
menu_anterior = 4150;
aux_menu = 4165;
opciones_menu = 17;
segunda_parte = 0;
pintaRotuloNodos();
pintaEscenas();

ESTABLECER VALOR DE ESCENA

condicion_menu=0;
menu_anterior = 4165;
aux_menu = 4170;
pintaRotuloNodos();
segunda_parte = 0;
if(valor_sup>100) valor_sup=100;
else if(valor_sup<0) valor_sup=0;
pintaNivel(valor_sup);
aux_iluminacion = ((unsigned int16)valor_sup*254)/100;

tx[0]=NORMAL; tx_2[0]=REPETIDO;
tx[2]=0xA3; tx_2[2]=dir;
tx[3]=aux_iluminacion; tx_2[3]=0x40+aux_scroll_4;
envio_2=1;

[aux_menu == 5130]

ELECCIÓN DE IDIOMA

aux_menu = 60;
menu_anterior = 1;
opciones_menu = 3;
condicion_menu=1;
pintaRetorno(0,56,3);

H*

Interrupción por pulsación del botón central

Entrada en un estado no contemplado

[salvapantallas >= 30]

Interrupción por pulsación del botón central

ERROR

opciones_menu = 1;
menu_anterior = 1;
envio=0;
envio_2=0;

aux_menu == 7000

GW611100
PASARELA KNX-DALI
ingenium *
Bes
www.besknx.com

```
#include "header.h"
#include "8x16_recortado.h"
#include "16x32_recortado.h"
#include "24x48_recortado.h"
#include "56x80_recortado.h"
#include "Simbolos24x48.h"

#fuses INTRC_IO,PUT,PROTECT,BROWNOUT,/*BORV27,*/NOWRT,NOMCLR,WDT1024,NOLVP,NOWDT,NODEBUG,NOFCMEN,CPB,NOIESO//,NOPLLEN

#define LCD_RS      PIN_C0
#define LCD_WR      PIN_C3
 #define LCD_CS     PIN_C1
#define LCD_PORT    PORT_C

#define FILAS_VISIBLES 5
#define ALTURA_CHAR 48

int1 nodos[64], v_grupos[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, v_nodos[64];

unsigned int16 scroll=0, pos_cursor = 144;
unsigned int8 del_cursor=0, aux_scroll=0, aux_scroll_1=0, aux_scroll_2=0, aux_scroll_3=0, aux_scroll_4=0;
int1 segunda_parte=0, envio=0, envio_2=0;

char tx[4]={0x00,0xFE,0x00,0x00}, tx_2[4]={0x00,0xFE,0x00,0x00}, rx[4]={0,0,0,0}, cadena[14];
unsigned int lista_nodos_activos[64], i_nodo=0, i_grupo=0;
unsigned int contador_menu = 1, opciones_menu = 1, nodo, grupo, nodo_real, hit=0;
unsigned int16 aux_menu=0, menu_anterior = 1, salto_linea, aux_iluminacion;
signed int16  dir, fila;
unsigned int valor_sup=100, valor_inf=20, valor_escena=100;

#define BUF_SIZE 4
//#define inc(pos) {pos++; pos&=(BUF_SIZE-1);}
char cbuff[BUF_SIZE];                    // Buffer
unsigned int rx_next=0, rx_read=0, salvapantallas = 0;
int1 tipo_dir, redireccion = 0, limite, flag_espera=0, botonOK=0, bloqueo=0, recibido = 0, cabecera=0, condicion_menu=0;
int1 RB5_anterior=1, RB6_anterior=1, RB7_anterior=1, RB5_actual=0, RB6_actual=0, RB7_actual=0;

const unsigned char init_cmd[18]={0xCB,0xCF,0xE8,0xEA,0xED,0xF7,0xC0,0xC1,0xC5,0xC7,0x36,0x3A,0xB1,0xB6,0xF2,0x26,0xE0,0xE1};
const unsigned char init_data_count[18]={5,3,3,2,4,1,1,1,2,1,1,1,2,3,1,1,15,15};
const unsigned char init_data[62]={0x39,0x2C,0x00,0x34,0x02,
                      0x00,0xC1,0x30,
                      0x85,0x00,0x78,
                      0x00,0x00,
                      0x64,0x03,0x12,0x81,
                      0x20,
                      0x23,
                      0x10,
                      0x3E,0x28,
                      0x86,
                      0x2C,                        // registro a modificar para cambiar la orientación
                      0x55,
                      0x00,0x18,
                      0x08,0x82,0x27,
                      0x00,
                      0x01,

                      0x0F,0x31,0x2B,0x0C,0x0E,0x08,0x4E,0xF1,0x37,0x07,0x10,0x03,0x0E,0x09,0x00,

                      0x00,0x0E,0x14,0x03,0x11,0x07,0x31,0xC1,0x48,0x08,0x0F,0x0C,0x31,0x36,0x0F};

/*****************************************************************************/
/*****************************FUNCIONES***************************************/
```

```
61  /*************************************************************************/
62
63  void configuracion_PIC(){
64
65      setup_oscillator(OSC_32MHZ);
66
67      setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);      //2,1 s overflow
68
69      setup_ccp1(CCP_PWM);
70      set_pwm1_duty((int16)970);
71      setup_timer_2(T2_DIV_BY_16,255,1);      //256 us overflow, 256 us interrupt
72
73      SETUP_SPI(SPI_MASTER | SPI_H_TO_L | SPI_CLK_DIV_4 | SPI_XMIT_L_TO_H);
74
75      enable_interrupts(INT_TIMER0);
76      //enable_interrupts(INT_EXT_H2L);
77      //enable_interrupts(INT_EXT1_H2L);
78      //enable_interrupts(INT_EXT2_H2L);
79      enable_interrupts(INT_RB);
80      disable_interrupts(INT_RDA);
81      enable_interrupts(GLOBAL);
82
83      //estado inicial
84      output_high(LCD_CS);
85      output_high(LCD_WR);
86      output_high(LCD_RS);
87  }
88
89  void LCD_Command(int8 comando)
90  {
91      output_low(LCD_RS);
92      spi_write(comando);
93  }
94
95  void LCD_Data(int8 dato)
96  {
97      output_high(LCD_RS);
98      spi_write(dato);
99  }
100
101 //void LCD_SetPos(int16 y1,int16 y2,char x1,char x2)  //Para pantalla horizontal.
    Así nuestro eje x es el horizontal (para la pantalla es al revés)
102 void LCD_SetPos(int16 x1,int16 x2,int16 y1,int16 y2)
103     {
104     // sitúo coordenada x
105     LCD_Command(0x2A);
106     LCD_Data(x1>>8);
107     LCD_Data(x1&0xFF);
108     LCD_Data(x2>>8);                    // tengo que enviar este dato (320=0x140)
    para configurar EC[15:0], sino
109     LCD_Data(x2&0xFF);              // llega a un punto en el que SC es igual o
    mayor a EC e ignora el resto de datos
110     LCD_Command(0x2C);
111
112     // sitúo coordenada y  (Hago la resta al valor máximo del eje porque el (0,0)
    está en la esquina superior
113     // izquierda del TFT y en el TFT de Truly está en la esquina inferior izquierda.
    Así las hago totalmente compatibles
114     LCD_Command(0x2B);
115     LCD_Data((239-y2)>>8);
116     LCD_Data((239-y2)&0xFF);
117     LCD_Data((239-y1)>>8);
118     LCD_Data((239-y1)&0xFF);
119     LCD_Command(0x2C);
120 }
121
122 void LCD_PutChar(int16 x, int16 y, char c, int16 fColor, int16 bColor, int8 tam,
    int1 tipo) //Para pantalla horizontal. Así nuestro eje x es el horizontal (para la
    pantalla es al revés)
123 {
124                 unsigned int16 offset,i,j,range=16*tam,k=0, aux_x=x;
125                 int recF_LCD, min, recC_LCD;// max;
126                 unsigned char m, comparador, aux_c=c;
```

```
127
128             if(tam == 3 || tam == 7){
129                 comparador='+';
130                 min=4;
131             }
132             else{
133                 comparador='%';
134                 min=8;
135             }
136
137             //condicional para evitar la escritura en pantalla para aquello que esta
                fuera de contorno
138             if((y>152 || y<min) && tipo)
139                 return;
140
141         if(tam != 7){
142             for(k=0;k<tam;k++){
143                 x=aux_x+8*k;
144                 c=(aux_c-comparador)*tam+(char)k;;
145                 LCD_SetPos(x,x+8-1,y,y+range-1);
146
147
148                 for(i=0; i<12*tam;i++)
149                 {
150                     offset=c;
151                     offset*=range;
152                     offset+=i;
153                     if(tam==1) m=Font8x16[offset];
154                     else if(tam==2) m=Font16x32[offset];
155                     else if(tam==3) m=Font24x48[offset];
156
157
158                     for(j=0;j<8;j++)
159                     {
160                         if((m&0x80)==0x80)
161                         {
162                             LCD_Data(fColor>>8);
163                             LCD_Data(fColor&0xFF);
164                         }
165                         else
166                         {
167                             LCD_Data(bColor>>8);
168                             LCD_Data(bColor&0xFF);
169                         }
170                         m<<=1;
171                     }
172                 }
173             }
174         }
175         else{
176             for(k=0;k<3;k++){
177                 x=aux_x+16*k;
178                 c=(aux_c-comparador)*3+(char)k;
179                 LCD_SetPos(x,x+16-1,y,y+96-1);
180
181                 for(i=0; i<48;i++)
182                 {
183                     offset=c;
184                     offset*=48;
185                     offset+=i;
186                     m=Font24x48[offset];
187
188                     for(j=0;j<8;j++)
189                     {
190                         for(recF_LCD=0;recF_LCD<2;recF_LCD++){
191                             for(recC_LCD=0;recC_LCD<2;recC_LCD++){
192                                 if((m&0x80)==0x80)
193                                 {
194                                     LCD_Data(fColor>>8);
195                                     LCD_Data(fColor&0xFF);
196                                 }
197                                 else
198                                 {
```

```
                                    LCD_Data(bColor>>8);
                                    LCD_Data(bColor&0xFF);
                                }
                            }
                            m<<=1;
                        }
                    }
                }
            }
        }
    }

    void LCD_PutSim(int16 x, int16 y, char c, int16 fColor, int16 bColor, int1
    tipo)        //Para pantalla horizontal. Así nuestro eje x es el horizontal (para la
    pantalla es al revés)
    //void LCD_PutChar8x16(int16 y, int16 x, char c, int16 fColor, int16 bColor)
    //Para pantalla vertical
    {
                    unsigned int16 i,j,offset, aux_x=x, k=0, aux_c=c;
                    unsigned char m, comparador='0';

        //condicional para evitar la escritura en pantalla para aquello que esta fuera
        de contorno
        if((y>144 || y<0) && tipo)
            return;

        for(k=0;k<3;k++){
            x=aux_x+8*k;
            c=(aux_c-comparador)*3+(char)k;
            LCD_SetPos(x,x+8-1,y,y+48-1);

            for(i=0; i<48;i++)
            {
                offset=c;
                offset*=48;
                offset+=i;

                m=FontSim24x48[offset];

                for(j=0;j<8;j++)
                {
                    if((m&0x80)==0x80)
                    {
                        LCD_Data(fColor>>8);
                        LCD_Data(fColor&0xFF);
                    }
                    else
                    {
                        LCD_Data(bColor>>8);
                        LCD_Data(bColor&0xFF);
                    }

                    m<<=1;
                }
            }
        }
    }

    void LCD_PutChar56x80(int16 x, int16 y, char c, int16 fColor, int16 bColor)
    //Para pantalla horizontal. Así nuestro eje x es el horizontal (para la pantalla es
    al revés)
    //void LCD_PutChar8x16(int16 y, int16 x, char c, int16 fColor, int16 bColor)
    //Para pantalla vertical
    {
                    unsigned int16 i,j,offset, aux_x=x, k=0, aux_c=c;
                    unsigned char m, comparador='0';

        for(k=0;k<7;k++){
            x=aux_x+8*k;
            c=(aux_c-comparador)*7+(char)k;
            LCD_SetPos(x,x+8-1,y,y+112-1);

            for(i=0; i<80;i++)
```

```
265             {
266                 offset=c;
267                 offset*=80;
268                 offset+=i;
269 
270                 m=Font56x80[offset];
271 
272                 for(j=0;j<8;j++)
273                 {
274                     if((m&0x80)==0x80)
275                     {
276                         LCD_Data(fColor>>8);
277                         LCD_Data(fColor&0xFF);
278                     }
279                     else
280                     {
281                         LCD_Data(bColor>>8);
282                         LCD_Data(bColor&0xFF);
283                     }
284 
285                     m<<=1;
286                 }
287             }
288         }
289     }
290 
291     void Clear_LCD(int16 color)
292         {
293                                     int8 d1,d2, max;
294                                     int16 i,j;
295         if(aux_menu == 7000 || botonOK) max=239;
296         else max=199;
297 
298             LCD_SetPos(0,319,0,max);
299 
300             d1=color>>8;
301             d2=color&0xFF;
302             LCD_Data(d1);
303             LCD_Data(d2);
304 
305 
306         for (j=0;j<320;j++)
307             {
308                 for(i=0;i<max+1;i++)
309                 {
310                     LCD_Data(d1);
311                     LCD_Data(d2);
312 
313                 }
314             }
315         }
316 
317     void Init_LCD(void)
318     {
319         char byta=0, bit_num=0, h_index=0, w_index=0;
320 
321         output_low(LCD_CS);
322 
323         while(byta<sizeof(init_cmd))                    // byte is init_cmd index
324         {
325             LCD_Command(init_cmd[byta++]);
326             while(h_index<init_data_count[bit_num])         // h_index is size of
            init_data command part counter
327             {                                       // bit_num is init_data_count index
328                  LCD_Data(init_data[w_index++]);   // w_index is init_data index
329                  h_index++;
330             }
331             bit_num++;
332             h_index=0;
333         }
334         LCD_Command(0x11);
335         delay_ms(120);
336         LCD_Command(0x2C);
```

```
337         LCD_Command(0x29);
338         LCD_Command(0x2C);
339     }
340
341     void pintaMenu(unsigned int16 escena){
342
343         envio=0, envio_2=0, flag_espera=0;
344
345         switch(escena){
346             case 1:
347                 Clear_LCD(0x0000);
348                 menu_anterior = 1;
349                 aux_menu = 1;
350                 opciones_menu = 6;
351                 condicion_menu=1;
352
353                 //cabecera de la pantalla
354                 cabecera=1;
355                     cadena="MENU";
356                     StringToChar(cadena, 120, 200, 2, 0);
357                     Linea_LCD(0, 200, 320, 2, 0xFB00);
358                 cabecera=0;
359
360                 fila=152+48*scroll;
361
362                 /*//cadena="N INSTALACION";
363                 cadenas(0);
364                 StringToChar(cadena, 0, fila, 2, 1);
365
366                 fila-=48;
367                 //cadena="EXTENDER";
368                 cadenas(1);
369                 StringToChar(cadena, 0, fila, 2, 1);
370
371                 fila-=48;
372                 //cadena="DETECTAR";
373                 cadenas(2);
374                 StringToChar(cadena, 0, fila, 2, 1);
375
376                 fila-=48;
377                 //cadena="NODOS";
378                 cadenas(3);
379                 StringToChar(cadena, 0, fila, 2, 1);
380
381                 fila-=48;
382                 //cadena="GRUPOS";
383                 cadenas(4);
384                 StringToChar(cadena, 0, fila, 2, 1);
385
386                 fila-=48;
387                 //cadena="IDIOMAS";
388                 cadenas(5);
389                 StringToChar(cadena, 0, fila, 2, 1);*/
390
391                 for(int row=0; row<opciones_menu; row++){
392                     cadenas(row);
393                     StringToChar(cadena, 0, fila, 2, 1);
394                     fila-=48;
395                 }
396             break;
397
398             case 10:
399                 aux_menu = 10;
400                 menu_anterior = 1;
401                 condicion_menu=0;
402                 pintaEspera();
403
404                 //TRAMA DE NUEVA INSTALACION//
405                 tx[0]=EXTENDIDO_1;
406                 tx[2]=0x12;
407                 tx[3]=0x34;
408
409                 envio=1;
```

```
            ////////////////////////////
        break;

        case 20:
            aux_menu = 20;
            menu_anterior = 1;
            condicion_menu=0;
            pintaEspera();

            ////TRAMA DE NUEVO BALASTRO (extender instalacion)////
            tx[0]=REPETIDO;
            tx[2]=0xA5;
            tx[3]=0x00;
            envio=1;
            ////////////////////////////////
        break;

        case 30:
            aux_menu = 30;
            menu_anterior = 1;
            condicion_menu=0;
            pintaEspera();

            //TRAMA DE DETECCION//
            tx[3]=0x99;
            tx[0]=NORMAL;
            hit=0;
            envio = 1;
            //la direccion se hace en el main xq aqui se llega por llamada a
            interrupcion y la marca de lectura esta en otra distinta
            ////////////////////////////////
        break;

        case 40:
            menu_anterior = 1;
            condicion_menu=0;
            //if(aux_menu!=40) aux_scroll=0;
            aux_menu = 40;
            salto_linea=0;
            tipo_dir=0;

            //inicializo el vector de grupos, no es relevante a la hora de mandar
            ordenes de asignacion
            //de esta forma esta limpio independientemente del nodo que se elija
            //de otro modo el vector quedaria guardado y daria lugar a confusiones
            en la visualizacion de pantalla
            for(int i=0;i<16;i++)
                v_grupos[i]=0;

            //cadena="ESCOJA NODO";
            cadenas(6);
            StringToChar(cadena, 0, 148, 2, 1);
            pintaNodosActivos();

            //cabecera de la pantalla
            //cadena="N O D O S";
            cabecera=1;
                cadenas(3);
                StringToChar(cadena, 116, 200, 2, 0);
                Linea_LCD(0, 200, 320, 2, 0xFB00);
            cabecera=0;

            opciones_menu = salto_linea+2;
        break;

        case 410:
            condicion_menu=1;
            //poner la direccion del nodo en la trama SOLAMENTE AQUI,
            //se va a ir modificando en funcion del cursor
            //se guarda en una lista ordenada el numero de nodo y un indice, la
            direccion es el contenido de la lista en el indice
            if(aux_scroll == salto_linea && aux_menu == 40){
                bloqueo=1;
```

```
                nodo = 0xFF;
            }
            else if(aux_scroll != salto_linea && aux_menu == 40){
                bloqueo=0;
                nodo_real = lista_nodos_activos[aux_scroll];
                nodo = 2*nodo_real+1;  //para el nodo n hay que escribir en dato1 2n+1
            }
            //cabecera de la pantalla
            pintaRotuloNodos();

            menu_anterior = 40;
            aux_menu = 410;
            opciones_menu = 6;

            fila=152+48*scroll;

            for(row=7; row<10; row++){
                limite=1;
                cadenas(row);
                StringToChar(cadena, 0, fila, 2, 1);
                fila-=48;
            }
            for(row=10; row<12; row++){
                cadenas(row);
                StringToChar(cadena, 0, fila, 2, 1);
                fila-=48;
            }
        /*  limite=1;
            //cadena="SOLICITAR GRUPOS";
            cadenas(7);
            StringToChar(cadena, 0, fila, 2, 1);

            limite=1;
            fila-=48;
            //cadena="ASIGNAR GRUPO";
            cadenas(8);
            StringToChar(cadena, 0, fila, 2, 1);

            limite=1;
            fila-=48;
            //cadena="ASIGNAR ESCENA";
            cadenas(9);
            StringToChar(cadena, 0, fila, 2, 1);

            //fila-=48;
            //cadena="LOCALIZAR";
            cadenas(10);
            StringToChar(cadena, 0, fila, 2, 1);

            fila-=48;
            //cadena="CONFIGURAR";
            cadenas(11);
            StringToChar(cadena, 0, fila, 2, 1);*/

            //fila-=48;
            pintaRetorno(0,fila,3);
        break;

        case 4110:  //obtener los grupos a los que pertenece un nodo
            condicion_menu=0;
            menu_anterior = 410;
            aux_menu = 4110;
            opciones_menu = 1;

            // OBTENER GRUPOS 0-7
            tx[0]=170;
            tx[2]=nodo;
            tx[3]=0xC0;

            // OBTENER GRUPOS 8-15
            tx_2[0]=170;
            tx_2[2]=nodo;
            tx_2[3]=0xC1;
```

```
552
553                 segunda_parte = 0;
554                 envio_2=1;
555
556                 pintaEspera();
557             break;
558
559             case 4120: //asignacion de nodos a grupos
560                 if(aux_menu == 4125) Clear_LCD(0x0000);
561                 //if(aux_menu!=4120)    aux_scroll_1=0;
562                 condicion_menu=0;
563                 aux_menu = 4120;
564                 pintaGrupos();
565                 pintaRotuloNodos();
566                 menu_anterior = 410;
567                 opciones_menu = 17;
568                 tx[0]=REPETIDO;
569                 tx[2]=nodo;
570             break;
571
572             case 4125:
573                 aux_menu = 4125;
574                 menu_anterior = 4120;
575                 opciones_menu=1;
576                 condicion_menu=0;
577                 if(v_grupos[aux_scroll_1] == 1)
578                     //TRAMA DE ASIGNACION
579                     tx[3]=0x60+(aux_scroll_1);
580
581                 else if(v_grupos[aux_scroll_1] == 0)
582                     //TRAMA DE DESASIGNACION
583                     tx[3]=0x70+(aux_scroll_1);
584
585                 envio = 1;
586
587                 pintaEspera();
588             break;
589
590             case 4130:
591                 if(aux_menu == 4135) Clear_LCD(0x0000);
592                 /*if(aux_menu!=4130){
593                     scroll=0;
594                     aux_scroll_4=0;
595                 }*/
596                 condicion_menu=0;
597                 aux_menu = 4130;
598                 menu_anterior = 410;
599                 opciones_menu=17;
600                 pintaRotuloNodos();
601
602                 //cadena="ESTABLECE ESCENA";
603                 cadenas(22);
604                 StringToChar(cadena, 0, 148, 2, 1);
605
606                 pintaEscenas();
607             break;
608
609             case 4135:
610                 aux_menu = 4135;
611                 menu_anterior = 4130;
612                 condicion_menu=0;
613
614                 tx[0]=NORMAL;
615                 tx[2]=nodo;
616                 tx[3]=0x10+aux_scroll_4;
617
618                 envio=1;
619                 pintaEspera();
620             break;
621
622             case 4140: //blink
623                 condicion_menu=0;
624                 aux_menu = 4140;
```

```
            opciones_menu = 1;
            if(tipo_dir==0){
                menu_anterior = 410;
                dir=nodo;
            }
            else{
                menu_anterior = 510;
                dir=grupo;
            }

            //cadena="LOCALIZANDO";
            cadenas(13);
            StringToChar(cadena, 0, 100, 2, 1);

            tx[0]=NORMAL;
            tx[2]=dir;
            tx[3]=0x00;

            tx_2[0]=NORMAL;
            tx_2[2]=dir;
            tx_2[3]=0x05;

            envio_2=1;
        break;

        case 4150:
            Clear_LCD(0x0000);
            condicion_menu=1;
            pintaRotuloNodos();

            if(tipo_dir==0){
                menu_anterior = 410;
                dir=nodo;
            }
            else{
                menu_anterior = 510;
                dir=grupo;
            }

            aux_menu = 4150;
            opciones_menu = 4;

            fila=152;
            for(row=14; row<17; row++){
                cadenas(row);
                StringToChar(cadena, 0, fila, 2, 1);
                fila-=48;
            }
            //cadena="LIM SUPERIOR";
            /*cadenas(14);
            StringToChar(cadena, 0, 152, 2, 1);

            //cadena="LIM INFERIOR";
            cadenas(15);
            StringToChar(cadena, 0, 104, 2, 1);

            limite=1;
            //cadena="VALOR ESCENA";
            cadenas(16);
            StringToChar(cadena, 0, 56, 2, 1);*/

            pintaRetorno(0,8,3);
        break;

        case 4155:
            condicion_menu=0;
            menu_anterior = 4150;
            aux_menu = 4155;
            pintaRotuloNodos();
            segunda_parte = 0;
            opciones_menu = 1;

            if(valor_sup>100) valor_sup=100;
```

```
698             else if(valor_sup<0) valor_sup=0;
699
700             pintaNivel(valor_sup);
701
702             //cadena="LIM SUPERIOR(%)";
703             cadenas(14);
704             StringToChar(cadena, 0, 152, 2, 1);
705
706             //me preocupa que al multiplicar valor_XXXXXX por 255 primero vaya a dar
                un numero de mas de 8 bits, desborde y haga mal la cuenta
707             //por eso se calcula en una variable tipo int16 y luego se castea a
                char, porque se sabe que va aquedar un numero menor o igual a 255
708             aux_iluminacion = ((unsigned int16)valor_sup*254)/100;
709
710             //ASIGNAR VALOR SUPERIOR
711             //primero se guarda en DTR
712             tx[0]=NORMAL;
713             tx[2]=0xA3;
714             tx[3]=aux_iluminacion;
715
716             //el valor del DTR se introduce en max level
717             tx_2[0]=REPETIDO;
718             tx_2[2]=dir;
719             tx_2[3]=0x2A;
720
721             envio_2=1;
722         break;
723
724         case 4160:
725             condicion_menu=0;
726             menu_anterior = 4150;
727             aux_menu = 4160;
728             pintaRotuloNodos();
729             segunda_parte = 0;
730             opciones_menu = 1;
731
732             if(valor_inf>100) valor_inf=100;
733             else if(valor_inf<0) valor_inf=0;
734
735             pintaNivel(valor_inf);
736
737             //cadena="LIM INFERIOR(%)";
738             cadenas(15);
739             StringToChar(cadena, 0, 152, 2, 1);
740
741             //me preocupa que al multiplicar valor_XXXXXX por 255 primero vaya a dar
                un numero de mas de 8 bits, desborde y haga mal la cuenta
742             //por eso se calcula en una variable tipo int16 y luego se castea a
                char, porque se sabe que va aquedar un numero menor o igual a 255
743             aux_iluminacion = ((unsigned int16)valor_inf*254)/100;
744
745             //ASIGNAR VALOR INFERIOR
746             //primero se guarda en DTR
747             tx[0]=NORMAL;
748             tx[2]=0xA3;
749             tx[3]=aux_iluminacion;
750
751             //el valor del DTR se introduce en min level
752             tx_2[0]=REPETIDO;
753             tx_2[2]=dir;
754             tx_2[3]=0x2B;
755
756             envio_2=1;
757         break;
758
759         case 4165:
760             condicion_menu=0;
761             menu_anterior = 4150;
762             /*if(aux_menu!=4165){
763                 scroll=0;
764                 aux_scroll_4=0;
765             }*/
766
```

```
767             aux_menu = 4165;
768             opciones_menu = 17;
769             segunda_parte = 0;
770 
771             pintaRotuloNodos();
772 
773             //cadena="ESCOJA ESCENA";
774             cadenas(17);
775             StringToChar(cadena, 0, 148, 2, 1);
776 
777             pintaEscenas();
778         break;
779 
780         case 4170:
781             condicion_menu=0;
782             opciones_menu = 1;
783             menu_anterior = 4165;
784             aux_menu = 4170;
785             pintaRotuloNodos();
786 
787             //cadena="ESCENA";
788             cadenas(17);
789             StringToChar(cadena, 0, 152, 2, 0);
790             pintaNumeros(114,152,aux_scroll_4,0xFFFF,0x0000,2,1,2);
791 
792             cabecera=1;
793                 cadena="(%)";
794                 StringToChar(cadena, 160, 152, 2, 0);
795             cabecera=0;
796 
797             if(valor_escena>100) valor_escena=100;
798             else if(valor_escena<0) valor_escena=0;
799 
800             pintaNivel(valor_escena);
801 
802             //me preocupa que al multiplicar valor_XXXXXX por 255 primero vaya a dar
                un numero de mas de 8 bits, desborde y haga mal la cuenta
803             //por eso se calcula en una variable tipo int16 y luego se castea a
                char, porque se sabe que va aquedar un numero menor o igual a 255
804             aux_iluminacion = ((unsigned int16)valor_escena*254)/100;
805 
806             //primero se guarda en DTR
807             tx[0]=NORMAL;
808             tx[2]=0xA3;
809             tx[3]=aux_iluminacion;
810 
811             //el valor del DTR se introduce en escena
812             tx_2[0]=REPETIDO;
813             tx_2[2]=dir;
814             tx_2[3]=0x40+aux_scroll_4;
815 
816             envio_2=1;
817         break;
818 
819         case 50:
820             condicion_menu=0;
821             menu_anterior = 1;
822             //if(aux_menu!=50) aux_scroll_2=0;
823             aux_menu = 50;
824             salto_linea=0;
825             opciones_menu=17;
826             tipo_dir=1;
827 
828             pintaRotuloNodos();
829 
830             cabecera=1;
831                 //cadena="G R U P O S";
832                 cadenas(4);
833                 StringToChar(cadena, 70, 200, 2, 0);
834                 Linea_LCD(0, 200, 320, 2, 0xFB00);
835             cabecera=0;
836 
837             //reinicia vector de nodos al igual que en case 40
```

```
838             for(int k=0;k<64;k++)
839                 v_nodos[k]=0;
840 
841             pintaGrupos();
842         break;
843 
844         case 510:
845             Clear_LCD(0x0000);
846             condicion_menu=1;
847             aux_menu = 510;
848             pintaRotuloNodos();
849             //poner la direccion del grupo en la trama tx[2] SOLAMENTE AQUI,
850             //se va a ir modificando en funcion del cursor
851             //grupo = 2*(aux_scroll_2+64)+1;
852             grupo = 2*aux_scroll_2+129;
853 
854             menu_anterior = 50;
855             opciones_menu = 4;
856 /*
857             //cadena="LOCALIZAR";
858             cadenas(10);
859             StringToChar(cadena, 0, 152, 2, 1);
860 
861             //cadena="CONFIG";
862             cadenas(11);
863             StringToChar(cadena, 0, 104, 2, 1);
864 
865             //cadena="ASIGNA NODO";
866             cadenas(12);
867             StringToChar(cadena, 0, 56, 2, 1);
868 */
869             fila=152;
870             for(row=10; row<13; row++){
871                 cadenas(row);
872                 StringToChar(cadena, 0, fila, 2, 1);
873                 fila-=48;
874             }
875 
876             pintaRetorno(0,8,3);
877         break;
878 
879         case 5130:
880             condicion_menu=0;
881             if(aux_menu == 5135) Clear_LCD(0x0000);
882             //if(aux_menu!=5130) aux_scroll_3=0;
883             menu_anterior = 510;
884             aux_menu = 5130;
885             salto_linea=0;
886 
887             //cadena="ASIGNA NODO";
888             cadenas(12);
889             StringToChar(cadena, 0, 148, 2, 1);
890 
891             pintaNodosActivos();
892             pintaRotuloNodos();
893 
894             for(i_grupo=0;i_grupo<salto_linea;i_grupo++){
895                 //fila=80-240*i_grupo+240*scroll;
896                 fila=60-240*(i_grupo-scroll);
897                 if(v_nodos[i_grupo] == 1)
898                     LCD_PutChar(210,fila,'@',0x0F00,0x0000,3, 1);
899 
900                 else if(v_nodos[i_grupo] == 0)
901                     LCD_PutChar(210,fila,'@',0x0000,0x0000,3, 1);
902             }
903             opciones_menu = salto_linea+1;
904         break;
905 
906         case 5135:
907             condicion_menu=0;
908             aux_menu = 5135;
909             menu_anterior = 5130;
910 
```

```
911                 tx[0]=REPETIDO;
912                 tx[2]=2*(lista_nodos_activos[aux_scroll_3])+1; //direccion virtual del
                    nodo
913
914                 if(v_nodos[aux_scroll_3] == 1)
915                     //TRAMA DE ASIGNACION
916                     tx[3]=0x60+(aux_scroll_2);
917
918                 else if(v_nodos[aux_scroll_3] == 0)
919                     //TRAMA DE DESASIGNACION
920                     tx[3]=0x70+(aux_scroll_2);
921
922                 envio = 1;
923
924                 pintaEspera();
925             break;
926
927             case 60:
928                 aux_menu = 60;
929                 menu_anterior = 1;
930                 opciones_menu = 3;
931                 condicion_menu=1;
932                 //cadena="I D I O M A S";
933                 LCD_PutChar(192,200,'-',0x0000,0x0000,2, 0);
934                 cabecera=1;
935                     cadenas(5);
936                     StringToChar(cadena, 80, 200, 2, 0);
937                     Linea_LCD(0, 200, 320, 2, 0xFB00);
938                 cabecera=0;
939
940                 cadena="ES - ES";
941                 StringToChar(cadena, 0, 152, 2, 1);
942
943                 cadena="EN - EN";
944                 StringToChar(cadena, 0, 104, 2, 1);
945
946                 pintaRetorno(0,56,3);
947             break;
948
949             case 7000:
950                 condicion_menu=0;
951                 if(aux_menu == 7000)
952                     menu_anterior = 1;
953                 else
954                     menu_anterior=aux_menu;
955                 aux_menu = 7000;
956
957                 Clear_LCD(0x0000);
958
959                 //LCD_PutChar(110,88,'B',0xFB03,0x0000,7, 1);
960                 LCD_PutChar56x80(70, 36,'0',0xFB00,0x0000);
961                 //LCD_PutChar(164,90,'E',0xFFFF,0x0000,5, 1);
962                 LCD_PutChar56x80(124,36,'1',0xFFFF,0x0000);
963                 //LCD_PutChar(204,90,'S',0xFFFF,0x0000,5, 1);
964                 LCD_PutChar56x80(180,36,'2',0xFFFF,0x0000);
965
966                 LCD_PutChar(248,125,'*',0xFB03,0x0000,2, 1);
967
968                 cabecera=1;
969                     cadena="INGENIUM";
970                     StringToChar(cadena, 118, 124, 2, 1);
971                     cadena="GW611100";
972                     StringToChar(cadena, 10, 190, 2, 0);
973                     cadenas(29);
974                     StringToChar(cadena, 10, 170, 1, 0);
975                     cadena="www.besknx.com";
976                     StringToChar(cadena, 134, 48, 1, 1);
977                 cabecera=0;
978
979             break;
980
981             default:
982                 opciones_menu = 1;
```

```
983             menu_anterior = 1;
984             cabecera=1;
985                 cadena="ERROR";
986                 StringToChar(cadena, 100, 100, 3, 1);
987             cabecera=0;
988             envio=0, envio_2=0;
989         break;
990      }
991      return;
992   }
993
994   ////////////////////////////////////////////////////////////////////////////
995
996   void inicbuff(void){                    // Inicia a  cbuff -------------------
997      int i;
998
999      for(i=0;i<BUF_SIZE;i++){             // Bucle que pone a AA todos los
1000        cbuff[i]=0x00;                    // caracteres en el buffer
1001     }
1002  }
1003
1004  void cadenas(int seleccion){
1005      int idioma = read_eeprom(0);
1006      switch(idioma){
1007          case 0: //español
1008              switch(seleccion){
1009                  case 0:
1010                      cadena="N INSTALACION";
1011                  break;
1012
1013                  case 1:
1014                      cadena="EXTENDER";
1015                  break;
1016
1017                  case 2:
1018                      cadena="DETECTAR";
1019                  break;
1020
1021                  case 3:
1022                      cadena="NODOS";
1023                  break;
1024
1025                  case 4:
1026                      cadena="GRUPOS";
1027                  break;
1028
1029                  case 5:
1030                      cadena="IDIOMAS";
1031                  break;
1032
1033                  case 6:
1034                      cadena="ESCOJA NODO";
1035                  break;
1036
1037                  case 7:
1038                      cadena="SOLICITAR GRUPOS";
1039                  break;
1040
1041                  case 8:
1042                      cadena="ASIGNAR GRUPO";
1043                  break;
1044
1045                  case 9:
1046                      cadena="ASIGNAR ESCENA";
1047                  break;
1048
1049                  case 10:
1050                      cadena="LOCALIZAR";
1051                  break;
1052
1053                  case 11:
1054                      cadena="CONFIGURAR";
1055                  break;
```

```
1056
1057                    case 12:
1058                        cadena="ASIGNAR NODO";
1059                    break;
1060
1061                    case 13:
1062                        cadena="LOCALIZANDO";
1063                    break;
1064
1065                    case 14:
1066                        cadena="LIM SUPERIOR(%)";
1067                    break;
1068
1069                    case 15:
1070                        cadena="LIM INFERIOR(%)";
1071                    break;
1072
1073                    case 16:
1074                        limite=1;
1075                        cadena="VALOR ESCENAS";
1076                    break;
1077
1078                    case 17:
1079                        cadena="ESCENA";
1080                    break;
1081
1082                    case 21:
1083                        cadena="ESCOJA GRUPO";
1084                    break;
1085
1086                    case 22:
1087                        cadena="ESTABLECE ESCENA";
1088                    break;
1089
1090                    case 23:
1091                        cadena="NODO";
1092                    break;
1093
1094    //              case 24:
1095    //                  cadena="I D I O M A S";
1096    //              break;
1097
1098    //              case 25:
1099    //                  cadena="N O D O S";
1100    //              break;
1101
1102    //              case 26:
1103    //                  cadena="G R U P O";
1104    //              break;
1105
1106    //              case 27:
1107    //                  cadena="N O D O";
1108    //              break;
1109
1110                    case 28:
1111                        cadena="ESPERE";
1112                    break;
1113
1114                    case 29:
1115                        cadena="PASARELA KNX-DALI";
1116                    break;
1117                }//fin seleccion
1118            break; //fin español
1119
1120            case 1: //ingles
1121                switch(seleccion){
1122                    case 0:
1123                        cadena="NEW INSTALATION";
1124                    break;
1125
1126                    case 1:
1127                        cadena="EXTEND";
1128                    break;
```

```
1129
1130                    case 2:
1131                        cadena="DETECT";
1132                    break;
1133
1134                    case 3:
1135                        cadena="NODES";
1136                    break;
1137
1138                    case 4:
1139                        cadena="GROUPS";
1140                    break;
1141
1142                    case 5:
1143                        cadena="LANGUAJE";
1144                    break;
1145
1146                    case 6:
1147                        cadena="CHOOSE NODE";
1148                    break;
1149
1150                    case 7:
1151                        cadena="REQUEST GROUPS";
1152                    break;
1153
1154                    case 8:
1155                        cadena="ASIGN GROUP";
1156                    break;
1157
1158                    case 9:
1159                        cadena="ASIGN SCENE";
1160                    break;
1161
1162                    case 10:
1163                        cadena="LOCALIZE";
1164                    break;
1165
1166                    case 11:
1167                        cadena="CONFIGURATE";
1168                    break;
1169
1170                    case 12:
1171                        cadena="ASIG NODE";
1172                    break;
1173
1174                    case 13:
1175                        cadena="LOCALIZING";
1176                    break;
1177
1178                    case 14:
1179                        cadena="UPPER LIMIT(%)";
1180                    break;
1181
1182                    case 15:
1183                        cadena="LOWER LIMIT(%)";
1184                    break;
1185
1186                    case 16:
1187                        limite=1;
1188                        cadena="SCENES VALUE";
1189                    break;
1190
1191                    case 17:
1192                        cadena="SCENE";
1193                    break;
1194
1195                    case 21:
1196                        cadena="CHOOSE GROUP";
1197                    break;
1198
1199                    case 22:
1200                        cadena="ESTABLISH SCENE";
1201                    break;
```

```
                case 23:
                    cadena="NODE";
                break;

//              case 24:
//                  cadena="L A N G U A J E S";
//              break;

//              case 25:
//                  cadena="N O D E S";
//              break;

//              case 26:
//                  cadena="G R O U P";
//              break;

//              case 27:
//                  cadena="N O D E";
//              break;

                case 28:
                    cadena="WAIT";
                break;

                case 29:
                    cadena="KNX-DALI GATEWAY";
                break;
            }//fin seleccion
        break; //fin ingles
    }//fin idioma
}

void StringToChar(char* cadena, int x, int y, int aux_tam, int1 tipo){
    int tam, aux_y=y;
    if(cabecera)tam=aux_tam;
    else{
        tam=aux_tam+1;
        y-=4;
    }
    int16 color;
    if(bloqueo && limite){
        color=0xF000;
        limite=0;
    }
    else color=0xFFFF;
    while (*cadena != '\0') {
        LCD_PutChar(x,y,*cadena,color,0x0000,tam,tipo);
        y=aux_y;
        x+=(8*tam);
        cadena++;           /* Vamos a la siguiente letra */
        tam=aux_tam;
        if(*cadena == ' '){
            x+=10;
            cadena++;           /* Vamos a la siguiente letra */
            if(!cabecera){
                tam=aux_tam+1;
                y-=4;
            }
        }
    }
}

void pintaGrupos(){
    //recorrer el vector grupos
    LCD_PutSim(288,190,'2',0xFFFF,0x0000,0);//flecha arriba
    LCD_PutSim(288,8,'3',0xFFFF,0x0000,0);//flecha abajo

    //cadena="CHOOSE GROUP";
    cadenas(21);
    StringToChar(cadena, 0, 148, 2, 1);
    LCD_PutChar(88,104,0x00,0x0000,0x0000,2,1);

```

```
1275
1276        for(i_grupo=0;i_grupo<16;i_grupo++){
1277            fila=40-240*(scroll-i_grupo);
1278
1279            pintaNumeros(120,fila,i_grupo,0xFFFF,0x0000,7,1,2);
1280
1281            if(v_grupos[i_grupo] == 1)// && aux_menu == 4120)
1282                LCD_PutChar(230,fila,'@',0x0F00,0x0000,3, 1);
1283
1284            else /*if(v_grupos[i_grupo] == 0 && aux_menu == 4120)*/
1285                LCD_PutChar(230,fila,'@',0x0000,0x0000,3, 1);
1286        }
1287        fila=240*scroll-3780;
1288        if(fila > 0){
1289            pintaNumeros(120,fila,99,0x0000,0x0000,7,1,2);
1290            pintaRetorno(150,fila,5);
1291        }
1292    }
1293
1294    void pintaEscenas(){
1295        LCD_PutSim(288,190,'2',0xFFFF,0x0000,0);//flecha arriba
1296        LCD_PutSim(288,8,'3',0xFFFF,0x0000,0); //flecha abajo
1297
1298        for(int i_escena=0;i_escena<16;i_escena++){
1299            fila=40-240*(scroll-i_escena);
1300
1301            pintaNumeros(120,fila,i_escena,0xFFFF,0x0000,7,1,2);
1302        }
1303        //fila=60-240*(16-scroll);
1304        fila=240*scroll-3780;
1305        if(fila >= 0){
1306            pintaNumeros(120,fila,99,0x0000,0x0000,7,1,2);
1307            pintaRetorno(148,fila,5);
1308        }
1309    }
1310
1311    void pintaNodosActivos(){
1312
1313        LCD_PutChar(86,80,'-',0x0000,0x0000,3, 1);
1314        LCD_PutChar(212,80,'-',0x0000,0x0000,2, 1);
1315        LCD_PutChar(228,80,'-',0x0000,0x0000,2, 1);
1316        Linea_LCD(212, 80, 32, 32, 0x0000);
1317        salto_linea=0;
1318        for(i_nodo=0;i_nodo<64;i_nodo++){
1319            if(nodos[i_nodo]){
1320                fila=40-240*(scroll-salto_linea);
1321                pintaNumeros(120,fila,i_nodo,0xFFFF,0x0000,7,1,2);
1322                lista_nodos_activos[salto_linea]=i_nodo;
1323                salto_linea++;
1324            }
1325        }
1326
1327        fila=60-240*(salto_linea-scroll);
1328        pintaNumeros(120,fila,99,0x0000,0x0000,7,1,2);
1329        if(aux_menu == 40 && fila==60){
1330            Clear_LCD(0x0000);
1331            //pintaNumeros(100,fila,99,0x0000,0x0000,7,1,2);
1332            cadena="BROADCAST";
1333            StringToChar(cadena, 86, fila+20,2,1);
1334        fila-=240;
1335        }
1336        if(fila >= 0){
1337            Clear_LCD(0x0000);
1338            //pintaNumeros(100,fila,99,0x0000,0x0000,7,1,2);
1339            pintaRetorno(150,fila+20,5);
1340        }
1341
1342        LCD_PutSim(288,8,'3',0xFFFF,0x0000,0); //flecha abajo
1343        LCD_PutSim(288,190,'2',0xFFFF,0x0000,0); //flecha arriba
1344    }
1345
1346    void pintaRetorno(int16 col, int fila,int tam){
1347        //por alguna razon hacer la llamada a esta funcion para pintar el retorno es mas
```

```
        eficiente que pintar el retorno directamente
1348            LCD_PutSim(col,fila-8,'1',0xFFFF,0x0000,1);  //simbolo de retorno
1349    }
1350
1351    void pintaNivel(int nivel){
1352
1353        LCD_PutChar(288,8,'-',0xFFFF,0x0000,3,0);
1354        LCD_PutChar(288,190,'+',0xFFFF,0x0000,3,0);
1355
1356        pintaRetorno(288, 100, 4);
1357
1358        pintaNumeros(58,50,nivel,0xFFFF,0x0000,7,1,3);
1359
1360    }
1361
1362    void pintaRotuloNodos(void){
1363        limite=0;
1364        if(tipo_dir){
1365            cabecera=1;
1366                //cadena="G R U P O";
1367                cadenas(4);
1368                StringToChar(cadena, 70, 200, 2, 0);
1369            cabecera=0;
1370            if(aux_menu!=50)
1371                pintaNumeros(200,200,aux_scroll_2,0xFFFF,0x0000,2,0,2);
1372            else
1373                pintaNumeros(200,200,99,0x0000,0x0000,2,0,2);
1374        }
1375        else if(aux_scroll == salto_linea){
1376            cabecera=1;
1377                cadena="BROADCAST";
1378                StringToChar(cadena, 100, 200, 2, 0);
1379            cabecera=0;
1380        }
1381        else{
1382            //cadena="N O D O";
1383            cabecera=1;
1384                if(aux_menu==40)cadenas(3);
1385                else cadenas(23);
1386                StringToChar(cadena, 90, 200, 2, 0);
1387                pintaNumeros(215,200,nodo_real,0xFFFF,0x0000,2,0,2);
1388            cabecera=0;
1389        }
1390        Linea_LCD(0, 200, 320, 2, 0xFB00);
1391    }
1392
1393    void pintaNumeros(int16 x, int16 y, int16 numero, int16 fColor, int16 bColor, int8
        tam, int1 tipo, int digitos){
1394        int1 cent;
1395        int8 dec, unit;
1396        if(digitos==3){
1397            cent=numero/100;
1398            LCD_PutChar(x,y,(char)cent+'0',fColor,bColor,tam, tipo);
1399            dec=(numero%100)/10;
1400            if(tam==7) x+=48;
1401            else x=x+8*tam;
1402        }
1403        else
1404            dec=numero/10;
1405
1406        LCD_PutChar(x,y,(char)dec+'0',fColor,bColor,tam, tipo);
1407
1408        unit=numero%10;
1409        if(tam==7) x+=48;
1410        else x=x+8*tam;
1411        LCD_PutChar(x,y,(char)unit+'0',fColor,0x0000,tam, tipo);
1412    }
1413
1414    void Linea_LCD(int16 x, int y, int16 largo, int alto, int16 color){
1415        int16 i,j;
1416
1417        LCD_SetPos(x,x+largo-1,y,y+alto-1);
1418
```

```
1419        for (j=y;j<y+alto;j++)
1420        {
1421            for(i=x;i<x+largo;i++)
1422            {
1423                LCD_Data(color>>8);
1424                LCD_Data(color&0xFF);
1425            }
1426        }
1427    }
1428
1429    void pintaEspera(){
1430        flag_espera=1;
1431        Clear_LCD(0x0000);
1432        //cadena="ESPERE";
1433        cadenas(28);
1434        StringToChar(cadena, 100, 104, 2,1);
1435    //  pintaNumeros(0,0,rx[1],0xFFFF,0x0000,3,0,3);
1436    }
1437
1438    /*****************************************************************************/
1439    /***************************INTERRUPCIONES*****************************/
1440    /*****************************************************************************/
1441
1442    #INT_TIMER0
1443    void  TIMER0_isr(void)
1444    {
1445        salvapantallas++;
1446        if(aux_menu == 7000) salvapantallas = 0;
1447        clear_interrupt(INT_TIMER0);
1448    }
1449
1450    //interrupcion por cambio en puerto B
1451    #INT_RB
1452    void  RB_isr(void)
1453    {
1454        salvapantallas = 0;
1455        int max_opciones, pos_menu, actual;
1456
1457        RB5_actual = bit_test(port_B,5);    //boton arriba
1458        RB6_actual = bit_test(port_B,6);    //boton OK
1459        RB7_actual = bit_test(port_B,7);    //boton abajo
1460
1461    //-------------------------COMPROBACION DE FLANCO DE BAJADA EN RB7
        (abajo)-------------------------//
1462        if(!RB7_actual && RB7_anterior){
1463
1464            if(aux_menu != 7000){
1465
1466                if(aux_menu == 4155){
1467                    valor_sup-=5;
1468                    pintaMenu(4155);
1469                }
1470                else if(aux_menu == 4160){
1471                    valor_inf-=5;
1472                    pintaMenu(4160);
1473                }
1474                else if(aux_menu == 4170){
1475                    valor_escena-=5;
1476                    pintaMenu(4170);
1477                }
1478                contador_menu++;
1479
1480                del_cursor=pos_cursor;
1481                pos_cursor=pos_cursor-48;
1482
1483                if(condicion_menu){
1484                    max_opciones = 4;
1485                }
1486                else
1487                    max_opciones = 1;
1488
1489                if (contador_menu > max_opciones && aux_menu != 4155 && aux_menu != 4160
                    && aux_menu != 4170){
```

```
1490                    contador_menu = max_opciones;
1491                    //este if hace que se puedan recorrer las opciones que estan fuera
                        de pantalla (por abajo)
1492                    if((contador_menu+scroll)<opciones_menu){
1493                        scroll++;
1494                        if(aux_menu==1 || aux_menu==410) Clear_LCD(0x0000);
1495                        if(aux_menu == 40) aux_scroll=scroll;
1496                        if(aux_menu == 4120) aux_scroll_1=scroll;
1497                        if(aux_menu == 50) aux_scroll_2=scroll;
1498                        if(aux_menu == 5130) aux_scroll_3=scroll;
1499                        if(aux_menu == 4130 || aux_menu == 4165) aux_scroll_4=scroll;
1500                        pintaMenu(aux_menu);
1501                    }
1502                    //limitador inferior de cursor
1503                    pos_cursor = 0;
1504                }
1505
1506                if(condicion_menu){
1507                    LCD_PutSim(295,del_cursor,'0',0x0000,0x0000,1); //simbolo flecha
1508                    LCD_PutSim(295,pos_cursor,'0',0xFFFF,0x0000,1); //simbolo flecha
1509                }
1510            //  condicion_menu=0;
1511            }
1512        }
1513
1514    //-------------------------------------------------------------------------------------
    -------//
1515
1516    //-------------------------COMPROBACION DE FLANCO DE BAJADA EN RB6
    (ok)-------------------------//
1517        if(!RB6_actual && RB6_anterior){
1518            pos_menu = contador_menu + scroll;
1519            actual = pos_menu+aux_menu;
1520            envio=0;
1521            envio_2=0;
1522            botonOK=1;
1523
1524            //limpia el lugar donde haya estado el cursor anteriormente
1525            LCD_PutSim(295,pos_cursor,'0',0x0000,0x0000,1); //simbolo flecha
1526
1527            if ((pos_menu >= opciones_menu && aux_menu != 1) || aux_menu == 4140 ||
            aux_menu == 20 || aux_menu==7000){
1528                scroll=0;
1529                if(aux_menu==410)   aux_scroll=0;
1530                if(aux_menu==4125)  aux_scroll_1=0;
1531                if(aux_menu==4135 || aux_menu == 4170)  aux_scroll_4=0;
1532                if(aux_menu==510)   aux_scroll_2=0;
1533                if(aux_menu==5135)  aux_scroll_3=0;
1534                Clear_LCD(0x0000);
1535                pintaMenu(menu_anterior);
1536            }
1537
1538            else {
1539                //si esta en la escena 40 (nodos) hacer operacion de bit para hacer que
                    la orden vaya especificamente a un nodo
1540                //con la suma de contador_menu+scroll se sabe a cual se esta apuntando
1541                if(aux_menu==40){
1542                    //aux_scroll=0;
1543                    scroll=0;
1544                    Clear_LCD(0x0000);
1545                    pintaMenu(410);
1546                }
1547                else if(bloqueo && (actual == 411 || actual == 412 || actual == 413 ||
                    actual == 4153)){}
1548                else if(aux_menu == 4120 && !bloqueo){
1549                    if(v_grupos[pos_menu-1] == 0) v_grupos[pos_menu-1] = 1;
1550                    else v_grupos[pos_menu-1] = 0;
1551                    scroll=0;
1552                    pintaMenu(4125);
1553                }
1554                else if(aux_menu == 4130 && !bloqueo){
1555                    scroll=0;
1556                    pintaMenu(4135);
```

```
1557                }
1558                else if(aux_menu == 4150){
1559                    aux_menu=aux_menu+5*contador_menu;
1560                    Clear_LCD(0x0000);
1561                    scroll=0;
1562                    pintaMenu(aux_menu);
1563                }/*
1564                else if(aux_menu == 4155 || aux_menu == 4160 || aux_menu == 4170){
1565                    scroll=0;
1566                    Clear_LCD(0x0000);
1567                    pintaMenu(menu_anterior);
1568                }*/
1569                else if(aux_menu == 4165){
1570                    //aux_scroll_4=0;
1571                    Clear_LCD(0x0000);
1572                    scroll=0;
1573                    pintaMenu(4170);
1574                }
1575                else if(aux_menu==50){
1576                    scroll=0;
1577                    pintaMenu(510);
1578                }
1579                else if(actual==511){
1580                    scroll=0;
1581                    Clear_LCD(0x0000);
1582                    pintaMenu(4140);
1583                }
1584                else if(actual==512){
1585                    scroll=0;
1586                    pintaMenu(4150);
1587                }
1588                else if(aux_menu == 5130){
1589                    if(v_nodos[pos_menu-1] == 0) v_nodos[pos_menu-1] = 1;
1590                    else v_nodos[pos_menu-1] = 0;
1591                    scroll=0;
1592                    Clear_LCD(0x0000);
1593                    pintaMenu(5135);
1594                }
1595                else if(actual == 61){
1596                    //escribir en eeprom valor para español
1597                    write_eeprom(0,0);
1598                    pintaMenu(60);
1599                }
1600                else if(actual == 62){
1601                    //escribir en eeprom valor para ingles
1602                    write_eeprom(0,1);
1603                    pintaMenu(60);
1604                }
1605                else{
1606                    if(aux_menu == 1) aux_menu=0;
1607                    aux_menu=aux_menu+scroll+contador_menu;
1608                    aux_menu=aux_menu*10;
1609                    scroll=0;
1610                    if(aux_menu==40)    aux_scroll=0;
1611                    if(aux_menu==4120)  aux_scroll_1=0;
1612                    if(aux_menu==4130 || aux_menu == 4165)  aux_scroll_4=0;
1613                    if(aux_menu==50)    aux_scroll_2=0;
1614                    if(aux_menu==5130)  aux_scroll_3=0;
1615                    Clear_LCD(0x0000);
1616                    pintaMenu(aux_menu);
1617                }
1618                //if(aux_menu==5130)    aux_scroll_3=0;
1619            }
1620            contador_menu = 1;
1621            pos_cursor=144;
1622
1623            if(condicion_menu)
1624                LCD_PutSim(295,pos_cursor,'0',0xFFFF,0x0000,1); //simbolo flecha
1625        //  condicion_menu=0;
1626            botonOK=0;
1627        }
1628    //-------------------------------------------------------------------------------
        -------//
```

```
1629
1630    //-------------------------COMPROBACION DE FLANCO DE BAJADA EN RB5
        (arriba)-------------------------//
1631        if(!RB5_actual && RB5_anterior){
1632
1633            if(aux_menu != 7000){
1634
1635                if(aux_menu== 4155){
1636                    valor_sup+=5;
1637                    pintaMenu(4155);
1638                }
1639                else if(aux_menu == 4160){
1640                    valor_inf+=5;
1641                    pintaMenu(4160);
1642                }
1643                else if(aux_menu == 4170){
1644                    valor_escena+=5;
1645                    pintaMenu(4170);
1646                }
1647                contador_menu--;
1648
1649                del_cursor=pos_cursor;
1650                pos_cursor=pos_cursor+48;
1651
1652                if (contador_menu < 1 && aux_menu != 4155 && aux_menu != 4160 &&
                    aux_menu != 4170){
1653                    contador_menu = 1;
1654                    //este if hace que se puedan recorrer las opciones que estan fuera
                        de pantalla (por arriba)
1655                    if((contador_menu+scroll)>1){
1656                        scroll--;
1657                        if(aux_menu==1 || aux_menu==410) Clear_LCD(0x0000);
1658                        if(aux_menu == 40) aux_scroll=scroll;
1659                        if(aux_menu == 4120) aux_scroll_1=scroll;
1660                        if(aux_menu == 50) aux_scroll_2=scroll;
1661                        if(aux_menu == 5130) aux_scroll_3=scroll;
1662                        if(aux_menu == 4130 || aux_menu == 4165) aux_scroll_4=scroll;
1663                        pintaMenu(aux_menu);
1664                    }
1665                    pos_cursor = 144;
1666                }
1667
1668                //limitador superior de cursor
1669                if (pos_cursor >= 144)
1670                    pos_cursor = 144;
1671
1672                if(condicion_menu){
1673                    LCD_PutSim(295,del_cursor,'0',0x0000,0x0000,1); //simbolo flecha
1674                    LCD_PutSim(295,pos_cursor,'0',0xFFFF,0x0000,1); //simbolo flecha
1675                }
1676            }
1677        }
1678    //-------------------------------------------------------------------------------
        -------//
1679
1680        RB5_anterior = RB5_actual;
1681        RB6_anterior = RB6_actual;
1682        RB7_anterior = RB7_actual;
1683        clear_interrupt(INT_RB);
1684    }
1685
1686
1687    //interrupcion RX
1688    #INT_RDA
1689    void  RDA_isr(void)
1690    {
1691        int trama_rx;
1692        cbuff[rx_next]=getc(PORT1);
1693
1694        if(cbuff[rx_next]==0x00 && !trama_rx) rx_next=0;
1695
1696        if(cbuff[rx_next]!=0x00 || trama_rx){
1697            rx_next++;
```

```
1698            trama_rx=1;
1699            salvapantallas = 0;
1700        }
1701
1702        if(rx_next>=BUF_SIZE){
1703            rx_next=0;
1704        }
1705
1706        if(rx_read==rx_next && trama_rx){ //no hay nuevos caracteres a procesar
1707
1708            salvapantallas = 0;
1709
1710            rx[0] = cbuff[0];
1711            rx[1] = cbuff[1];
1712            rx[2] = cbuff[2];
1713            rx[3] = cbuff[3];
1714
1715            trama_rx=0;
1716
1717            if(aux_menu == 30){
1718
1719                //ESTO HACE FALTA PARA SABER POR DONDE VA EL ESCANEO
1720                pintaNumeros(240,50,hit,0xFFFF,0x0000,3,1,2);
1721            }
1722
1723            if(rx[2] == 0x34 && rx[3] == 0x12){
1724                redireccion = 1;
1725                envio_2 = 0;
1726                envio = 0;
1727            }
1728        }
1729
1730
1731        //peticion_envio = 0;
1732        if(rx[0] == 1) recibido = 1;
1733
1734        clear_interrupt(int_rda);
1735    }
1736
1737    /*****************************************************************************/
1738    /*****************************************************************************/
1739    /*****************************************************************************/
1740
1741    void main (){
1742        configuracion_PIC();
1743        inicbuff();
1744        Init_LCD();
1745        aux_menu=1;
1746        pintaMenu(7000);
1747
1748        int g, parp=0;
1749
1750        for(int p_nodos=0;p_nodos<64;p_nodos++){
1751            nodos[p_nodos]=0;
1752            lista_nodos_activos[p_nodos]=0;
1753        }
1754        enable_interrupts(INT_RDA);
1755
1756        while(1){
1757
1758            if(redireccion == 1 && cbuff[0] == 0){
1759                redireccion = 0;
1760                pintaMenu(1);
1761            }
1762
1763            if(salvapantallas >= 30){
1764                salvapantallas=0;
1765                pintaMenu(7000);
1766            }
1767
1768            //efecto de movimiento
1769            if(flag_espera || aux_menu == 4140){
1770                if(parp<=50)
```

```
1771                LCD_PutChar(240,104,'*',0xFFFF,0x0000,2,1);
1772            else if(parp > 50 && parp <= 100)
1773                LCD_PutChar(240,104,'*',0x0000,0x0000,2,1);
1774            else if(parp > 100)
1775                parp = 0;
1776            parp++;
1777        }
1778
1779        switch(envio_2){
1780            case 0:
1781                switch(envio){
1782                    case 0:
1783                        putc(0x00,PORT1);
1784                    break;
1785
1786                    case 1:
1787                        if(aux_menu == 30){
1788                            while(hit<64){
1789                                tx[2]=2*hit+1;
1790                                putc(tx[0],PORT1);
1791                                delay_us(1);
1792                                putc(tx[1],PORT1);
1793                                delay_us(1);
1794                                putc(tx[2],PORT1);
1795                                delay_us(1);
1796                                putc(tx[3],PORT1);
1797
1798                                if (rx[0]==0x01){   //se recibe ACK
1799                                    if (rx[2] == 0xFF){  // El dispositivo contesta
1800                                        nodos[hit]=1;
1801                                        hit++;
1802                                    }
1803                                    else if(rx[2] == 0x00){ //el dispositivo no esta
1804                                        nodos[hit]=0;
1805                                        hit++;
1806                                    }
1807                                }
1808                                rx[0]=0x00;
1809                            }
1810                            //fin_deteccion=1;
1811                            inicbuff(); //una vez rellenado el vector de nodos se
                                inicializa el buffer
1812                            pintaMenu(1); //al acabar se vuelve al menu principal
1813                        }
1814
1815                        else{
1816                            putc(tx[0],PORT1);
1817                            delay_us(1);
1818                            putc(tx[1],PORT1);
1819                            delay_us(1);
1820                            putc(tx[2],PORT1);
1821                            delay_us(1);
1822                            putc(tx[3],PORT1);
1823
1824                            if((aux_menu == 4125 || aux_menu == 5135 || aux_menu ==
                                4135) && rx[0]==0x01){
1825                                scroll=0;
1826                                aux_scroll_1=0;
1827                                aux_scroll_3=0;
1828                                aux_scroll_4=0;
1829                                pintaMenu(menu_anterior);
1830                            }
1831                            rx[0]=0;
1832                        }
1833                    break;
1834                }//fin switch(envio)
1835            break;
1836
1837            case 1:
1838                switch(segunda_parte){
1839                    case 0:
1840                        putc(tx[0],PORT1);
1841                        delay_us(1);
```

```
1842                            putc(tx[1],PORT1);
1843                            delay_us(1);
1844                            putc(tx[2],PORT1);
1845                            delay_us(1);
1846                            putc(tx[3],PORT1);
1847
1848                            //fin_trama = 1;
1849                            if(recibido){
1850                                segunda_parte = 1;
1851
1852                                // OBTENER GRUPOS 0-7
1853                                if(aux_menu == 4110 && rx[2] == 0xFF){
1854                                    for(g=0;g<8;g++){
1855                                        if ((rx[3]&0x1<<g)!=0)
1856                                            v_grupos[g]=1;
1857                                        else
1858                                            v_grupos[g]=0;
1859                                    }
1860                                }
1861                                rx[0] = 0;
1862                                recibido=0;
1863                            }
1864                        break;
1865
1866                        case 1:
1867                            putc(tx_2[0],PORT1);
1868                            delay_us(1);
1869                            putc(tx_2[1],PORT1);
1870                            delay_us(1);
1871                            putc(tx_2[2],PORT1);
1872                            delay_us(1);
1873                            putc(tx_2[3],PORT1);
1874
1875                            //fin_trama = 1;
1876                            if(recibido){
1877                                segunda_parte = 0;
1878
1879                                // OBTENER GRUPOS 8-15
1880                                if(aux_menu == 4110 && rx[2] == 0xFF){
1881                                    for(g=0;g<8;g++){
1882                                        if ((rx[3]&0x1<<g)!=0)
1883                                            v_grupos[g+8]=1;
1884                                        else
1885                                            v_grupos[g+8]=0;
1886                                    }
1887                                    envio_2=0;
1888                                    pintaMenu(menu_anterior);
1889                                }
1890
1891                                if((aux_menu == 4155|| aux_menu == 4160 || aux_menu ==
                                    4170)){
1892                                    LCD_PutChar(106,50,0x5B,0x0F00,0x0000,7,1);
                                    //simbolo de check tamaño 7
1893                                    envio_2 = 0;
1894                                }
1895
1896                                rx[0] = 0;
1897                                recibido=0;
1898                            }
1899                        break;
1900                    }//fin switch(segunda_parte)
1901                break;
1902            }//fin switch(envio_2)
1903
1904        restart_wdt();
1905        }//fin while(1)
1906    }
1907
```