



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH  
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FINAL DE GRADO

**Grado en Ingeniería Electrónica Industrial y Automática**

**SISTEMA DE GESTIÓN DOMÓTICA PARA OPTIMIZAR EL  
CONSUMO ENERGÉTICO DE UNA VIVIENDA**



**Volumen IV - Anexos**

**Autor/a:** Oriol Riera Rovira  
**Director/a:** Manuel Andrés Manzanares Brotons  
**Convocatoria:** Barcelona, 18 de Junio de 2020



## Índice

<b>1.</b>	<b>CÓDIGO UNIDAD CENTRAL EN C</b>	<b>1</b>
1.1.	Código principal .....	1
1.2.	Configuraciones .....	22
1.3.	Código Eeprom 24512 .....	23
1.3.1.	Leer/Escribir datos.....	23
1.3.2.	Leer/Escribir floats.....	23
1.4.	Código Reloj DS1307 .....	24
<b>2.</b>	<b>CÓDIGO MÓDULO ENERGÍA EN C</b>	<b>25</b>



# 1. Código Unidad Central en C

## 1.1. Código principal

```

1  /// Autor: ORIOL RIERA ROVIRA ///
2  /// SISTEMA DE GESTIÓN DOMÓTICA PARA OPTIMIZAR EL CONSUMO ENERGÉTICO DE UNA
  VIVIENDA ///
3  /// TRABAJO DE FINAL DE GRADO ///
4
5  #include <18F4550.h>
6  #device HIGH_INTS = TRUE
7  #include <config.h>
8
9  char c; //RS232 DATA
10 int16 address = 0;
11 int count_day = 0; //Flag for the start of the day's address
12
13
14 unsigned int state = 1, first = 1; //Define state of functionality
15 char k; //KBD Data
16 int yr, month, date, day, hr, min, sec, newt; //CLK Time
17 char day_arr[7][4] = {"DOM", "LUN", "MAR", "MIE", "JUE", "VIE", "SAB"};
18
19
20 int temp_h = 24, temp_l = 0; //Temp desired for the room 24.0
21 int time_change = 1; //Day by default
22 int season = 1; //Warm season
23 int cont = 0; //Contar 20 veces 0.5 segundos para el timer
24 unsigned int alarm = 0;
25
26 int temp_h_s, temp_l_s; //Temp read by sensor
27 unsigned int16 P;
28
29 // Control manual/automatico
30 int LC1 = 1, ALC1 = 0xc6, LState = 0; //Luces control = 1 Automatico de serie +
  Address + General state (OFF)
31 int LC2 = 1, ALC2 = 0xc2;
32 int LC3 = 1, ALC3 = 0xc1;
33 int ToC = 1, AToC = 0xc5; //Toldo control = 1 Automatico de serie + Address
34 int BlindsC = 1, ABlindsC = 0xc7; //Blinds control = 1 Automatico de serie + Address
35 int AirC = 1, AAir = 0xc4; //Air control = 1 Automatico de serie + Address for i2c
36 int HeatC = 1, AHeat = 0xc3; //Heat control = 1 Automatic + Address
37
38
39 #int TIMER1
40 void control_time(void){
41   if (cont%2 == 0){ //Cada 1 segundo
42     rtc_get_time(yr,month,date,day,hr,min,sec);
43     newt = 1;
44     if (bit_test(alarm,0) == 1) output_toggle(PIN_A0); //Ventana
45     if (bit_test(alarm,2) == 1) output_toggle(PIN_A1); //Calidad del aire
46   }
47
48   if (cont%20 == 0 && cont != 0 && cont != 40){ //Cada 10 segundos
49     short int status;
50     do{
51       i2c_start();
52       i2c_write(0x40);
53       status = i2c_write(0x15);
54       i2c_stop();
55       if (status == 1) delay_ms(50); //If no answer wait 1s
56     }
57     while (status == 1);
58
59     i2c_start();
60     i2c_write(0x41);
61     alarm = i2c_read(0);
62     i2c_stop();
63
64     if (bit_test(alarm,0) == 0) output_low(PIN_A0);
65     if (bit_test(alarm,2) == 0) output_low(PIN_A1);
66   }
67
68   if ((cont+5)%10 == 0 && cont != 0){ //Cada 5 segundos
69     short int status;
70
71     do{

```

```

72     i2c_start();
73     i2c_write(0x40);
74     status = i2c_write(0x14);
75     i2c_stop();
76     if (status == 1) delay_ms(50); //If no answer wait 1s
77 }
78 while (status == 1);
79
80     i2c_start();//Restart the bus
81     i2c_write(0x41);//R/W bit high for a read
82     delay_cycles(10);
83     temp_h_s = i2c_read();//Read the data from the slave
84     temp_l_s = i2c_read();//Read the data from the slave
85     LState = i2c_read();
86     P = i2c_read();
87     P = P+(i2c_read()<<4);
88     int16 aux;
89     aux = i2c_read(0);
90     aux = aux<<8;
91     P = P+aux;
92
93     i2c_stop();
94
95     update_time(); //Mirar si se ha cambiado de franja horaria
96
97     if (P<1000) {
98         output_high(PIN_C0);
99         output_low(PIN_C1);
100        output_low(PIN_C2);
101    }
102    else if (P>4000){
103        output_low(PIN_C0);
104        output_low(PIN_C1);
105        output_high(PIN_C2);
106    }
107    else
108    {
109        output_low(PIN_C0);
110        output_high(PIN_C1);
111        output_low(PIN_C2);
112    }
113
114    first = 1; //So data can be updated
115 }
116
117 if (cont == 40 && cont!= 0){ //Cada 20 segundos
118     //New day to store data
119     if (hr == 0 && min <1){ //at 00:01 declare new day
120         count_day = ++address;
121     }
122     int P_aux;
123     P_aux = P/100;
124     write_ext_eeprom(address++,P_aux);
125     cont = 0;
126 }
127 cont++;
128 set_timer1(3036);
129 }
130
131
132 //Interruptions
133 #int_rda HIGH //For app communication: To develop
134 void rda_isr(void)
135 {
136     c = fgetc(PC);
137     output_toggle(PIN_A0);
138 }
139
140 #int_ext HIGH
141 void ext_isr (void){
142     if (bit_test(LState,2) == 1 ){ //3 bit is 1 means light is on so close it and
143         modo manual
144         LC1 = 3; //Change to OFF light

```

```

144     change_data(ALC1,LC1);
145     bit_clear(LState,2); //Change internal data
146 }
147 else{
148     LC1 = 2;
149     change_data(ALC1,LC1);
150     bit_set(LState,2);
151 }
152 first = 1;
153 }
154
155 #int_ext1 HIGH
156 void ext1_isr (void){
157     if (bit_test(LState,1) == 1 ){ //2 bit is 1 means light is on so close it and
        modo manual
158         LC2 = 3; //Change to OFF light
159         change_data(ALC2,LC2);
160         bit_clear(LState,1); //Change internal data
161     }
162     else{
163         LC2 = 2;
164         change_data(ALC2,LC2);
165         bit_set(LState,1);
166     }
167     first = 1;
168 }
169
170 #int_ext2 HIGH
171 void ext2_isr (void){
172     if (bit_test(LState,0) == 1 ){ //3 bit is 1 means light is on so close it and
        modo manual
173         LC3 = 3; //Change to OFF light
174         change_data(ALC3,LC3);
175         bit_clear(LState,0); //Change internal data
176     }
177     else{
178         LC3 = 2;
179         change_data(ALC3,LC3);
180         bit_set(LState,0);
181     }
182     first = 1;
183 }
184
185
186 #int_rb HIGH //Menu interface
187 void rb_isr (void)
188 {
189
190     if (input(PIN_B5) == 1){ // State == 0010 (ESC mode)
191         state = 10; //Specific state for ESC
192         first = 1;
193     }
194     else if (input(PIN_B6) == 1){ // State == 0100 (right pressed)
195         ++state;
196         first = 1;
197     }
198     else if (input(PIN_B7) == 1){ // State == 1000 (left pressed)
199         --state;
200         first = 1;
201     }
202 }
203
204 // Main
205 void main() {
206     init_routine();
207     //Variables
208     state = 5;
209     while (TRUE) {
210         switch(state){
211             case 0: state = 5; first = 1; break;
212             case 1: menu_temperatura(); break; //Temp
213             case 2: menu_potencia(); break; //Power
214             case 3: menu_mode(); break; //Out

```

```

215     case 4: menu_config(); break;
216     case 5:
217         menu_clock();
218         break;
219     case 10: state = 5; break; //Esc pressed
220     default: state = 1; first = 1; break;
221 }
222 if (address==0xffff){
223     address = 0; //Cuando se termina la EEPROM vuelve al principio
224     count_day = 0;
225 }
226 }
227
228 }
229
230 void init_routine(void){
231     lcd_init();
232     kbd_init();
233
234     //Interruptions
235     disable_interrupts(int_rda);
236     enable_interrupts(int_rda);
237     disable_interrupts(int_rb);
238     enable_interrupts(int_rb);
239     enable_interrupts(int_ext);
240     enable_interrupts(int_ext1);
241     enable_interrupts(int_ext2);
242
243     setup_timer_1(T1_INTERNAL);
244     set_timer1(3036); //0.5 segundos
245
246     //I/O Ports
247     bit_set(TRISB,4); //B0 as input
248     bit_set(TRISB,5); //B1 as input
249     bit_set(TRISB,6); //B2 as input
250     bit_set(TRISC,7); //Set pin_C7 RX as input
251     bit_clear(TRISA,0); //Set A0 as output: LED window open
252     bit_clear(TRISA,1); //Set A1 as output: LED bad air cuality
253     bit_clear(TRISC,0); //C0 output
254     bit_clear(TRISC,1); //C0 output
255     bit_clear(TRISC,2); //C0 output
256     bit_set(TRISC,5); //C5 / SCL input
257     bit_set(TRISC,4); //C4 / SDA input
258
259     lcd_putc("Welcome Home\nUpdating system");
260     delay_ms(50); //Let other pic initialize
261     enable_interrupts(INT_TIMER1);
262     enable_interrupts(global);
263     while(P == 0){}
264     state = 5;
265 }
266
267
268 void menu_clock(void){
269     if (state == 5 && newt == 1){
270         newt = 0;
271         printf(lcd_putc, "\f%2u:%2u:%2u\n%s
272             %2u/%2u/%2u",hr,min,sec,day_arr[day-1],date,month,yr);
273     }
274     if (input(PIN_B4) == 1){ //Okay pressed, change time
275         state = 0;
276         first = 1;
277         delay_ms(300);
278         while(TRUE){
279             int j = 0;
280             int new = 0;
281             switch(state){
282                 case 0:
283                     if (first == 1 && state == 0){
284                         printf(lcd_putc, "\fSet time <=\nSec:");
285                         first = 0;
286                     }
287                     while (j<2 && state == 0){

```



```

287         k = kbd_getc();
288         if (k!=0){
289             lcd_putc(k);
290             j++;
291             if (j == 1) new = (k-48)*10;
292             else new = new + (k-48);
293         }
294     }
295     if (new > 59)
296         new = 59;
297     while (j == 2 && state == 0){
298         if (input(pin_B4) == 1){ //Summit new time
299             sec = new;
300             rtc_write(state,sec);
301             ++state;
302             first = 1;
303             delay_ms(300);
304         }
305     }
306     break;
307 case 1: //Min
308     if (first == 1 && state == 1){
309         printf(lcd_putc,"\fSet time <=\nMin:");
310         first = 0;
311     }
312     while (j<2 && state == 1){
313         k = kbd_getc();
314         if (k!=0){
315             lcd_putc(k);
316             j++;
317             if (j == 1) new = (k-48)*10;
318             else new = new + (k-48);
319         }
320     }
321     if (new > 59)
322         new = 59;
323     while (j == 2 && state == 1){
324         if (input(pin_B4) == 1){ //Summit new time
325             min = new;
326             rtc_write(state,min);
327             ++state;
328             first = 1;
329             delay_ms(300);
330         }
331     }
332     break;
333 case 2: //Hour
334     if (first == 1 && state == 2){
335         printf(lcd_putc,"\fSet time <=\nHour:");
336         first = 0;
337     }
338     while (j<2 && state == 2){
339         k = kbd_getc();
340         if (k!=0){
341             lcd_putc(k);
342             j++;
343             if (j == 1) new = (k-48)*10;
344             else new = new + (k-48);
345         }
346     }
347     if (new > 23)
348         new = 23;
349     while (j == 2 && state == 2){
350         if (input(pin_B4) == 1){ //Summit new time
351             hr = new;
352             rtc_write(state,hr);
353             ++state; //Change next setting
354             first = 1;
355             delay_ms(300);
356         }
357     }
358     break;
359 case 3: //Day

```

```

360     if (first == 1 && state == 3){
361         printf(lcd_putc, "\fSet time <=\nDay:");
362         first = 0;
363     }
364     while (j<1 && state == 3){
365         k = kbd_getc();
366         if (k!=0){
367             lcd_putc(k);
368             j++;
369             new = (k-48)+1; //Day goes from 1(sunday) to 7(saturday);
370         }
371     }
372     if (new > 7)
373         new = 7;
374     while (j == 1 && state == 3){
375         if (input(pin_B4) == 1){ //Summit new time
376             day = new-1;
377             rtc_write(state,day);
378             state++; //Change next setting
379             first = 1;
380             delay_ms(300);
381         }
382     }
383     break;
384     case 4: //Date
385         if (first == 1 && state == 4){
386             printf(lcd_putc, "\fSet time <=\nDate:");
387             first = 0;
388         }
389         while (j<2 && state == 4){
390             k = kbd_getc();
391             if (k!=0){
392                 lcd_putc(k);
393                 j++;
394                 if (j == 1) new = (k-48)*10;
395                 else new = new + (k-48);
396             }
397         }
398         if (new > 31)
399             new = 31;
400         while (j == 2 && state == 4){
401             if (input(pin_B4) == 1){ //Summit new time
402                 date = new;
403                 rtc_write(state,date);
404                 ++state; //Change next setting
405                 first = 1;
406                 delay_ms(300);
407             }
408         }
409         break;
410     case 5: //Month
411         if (first == 1 && state == 5){
412             printf(lcd_putc, "\fSet time <=\nMonth:");
413             first = 0;
414         }
415         while (j<2 && state == 5){
416             k = kbd_getc();
417             if (k!=0){
418                 lcd_putc(k);
419                 j++;
420                 if (j == 1) new = (k-48)*10;
421                 else new = new + (k-48);
422             }
423         }
424         if (new > 12)
425             new = 12;
426         while (j == 2 && state == 5){
427             if (input(pin_B4) == 1){ //Summit new time
428                 month = new;
429                 rtc_write(state,month);
430                 ++state;
431                 first = 1;
432                 delay_ms(300);

```

```

433     }
434 }
435 break;
436 case 6: //Year
437     if (first == 1 && state == 6){
438         printf(lcd_putc, "\fSet time <=\nYear:");
439         first = 0;
440     }
441     while (j<2 && state == 6){
442         k = kbd_getc();
443         if (k!=0){
444             lcd_putc(k);
445             j++;
446             if (j == 1) new = (k-48)*10;
447             else new = new + (k-48);
448         }
449     }
450     while (j == 2 && state == 6){
451         if (input(pin_B4) == 1){ //Summit new time
452             yr = new;
453             rtc_write(state,yr);
454             state = 5;
455             first = 1;
456             delay_ms(300);
457             return;
458         }
459     }
460     break;
461 case 7: state = 0; first = 1; break;
462 case 10:
463     state = 5;
464     first = 1;
465     delay_ms(300);
466     return;
467
468     }
469 }
470 if(state>250) state = 6; //If we try to go under 0
471 }
472 }
473
474 void menu_temperatura(void){
475     if (first == 1 && state == 1){
476         first = 0;
477         printf(lcd_putc, "\fT Sens %2u.%u °C\nT Def: %2u.%u
478 °C", temp_h_s, temp_l_s, temp_h, temp_l);
479     }
480     if (input(PIN_B4) == 1){ //Okay pressed
481         lcd_putc("\fDefine new temp:\n");
482         int j = 0;
483         int temp_n[3]; //Guardar los datos de temperatura
484         while (j<2 && input(PIN_B5) == 0){
485             k = kbd_getc();
486             if (k!=0){
487                 lcd_putc(k);
488                 temp_n[j] = k-48;
489                 j++;
490             }
491         }
492         lcd_putc(".");
493         while (j == 2 && input(PIN_B5) == 0){
494             k = kbd_getc();
495             if (k!=0){
496                 lcd_putc(k);
497                 lcd_putc(" °C");
498                 temp_n[j] = k-48;
499                 j++;
500             }
501         }
502         while (TRUE){
503             if(input(PIN_B4) == 1){ //OKAY pressed
504                 first = 1;

```

```

505         temp_h = temp_n[0]*10+temp_n[1];
506         temp_l = temp_n[2];
507         short int status;
508         do{
509             i2c_start();
510             i2c_write(0x40); //Address of pic
511             i2c_write(0xbb); //Instruction to change temp
512             i2c_write(temp_h);
513             delay_ms(10);
514             status = i2c_write(temp_l);
515             i2c_stop();
516             if (status == 1) delay_ms(100); //If data not read properly try
                again with delay
517         }
518         while ( status == 1);
519         delay_ms(300);
520         break;
521     }
522     else if (input(PIN_B5) == 1){ //ESC pressed
523         state = 1;
524         break;
525     }
526 }
527 }
528 }
529
530 void menu_potencia(void){
531     if (first == 1 && state == 2){
532         printf(lcd_putc,"\fPower consumed:\n%lu W",P);
533         first = 0;
534     }
535     if (input(PIN_B4) == 1){ //Se pulsa OKAY
536         first = 1;
537         while(state == 2){
538             if (first == 1 && state == 2){
539                 int16 i;
540                 float E = 0.0;
541                 for (i=count_day;i<address;i++){
542                     E += read_ext_eeprom(i);
543                 }
544                 E = E/(address-count_day);
545                 E = E/10.0;
546                 printf(lcd_putc,"\fDaily E cons.:\n %.2f KWh",E);
547                 first = 0;
548             }
549         }
550         state = 2; first = 1; //Reset global screen
551     }
552 }
553
554
555 void menu_mode(void){
556     if(first == 1 && state == 3){
557         lcd_putc("\fChoose scene");
558         first = 0;
559     }
560     if (input(PIN_B4) == 1){
561         first = 1;
562         state = 1;
563         lcd_putc("\f <=");
564         delay_ms(150);
565         while(TRUE){
566             switch(state){
567                 case 0: state = 3; first = 1; break;
568                 case 1:
569                     if (first == 1 && state == 1){
570                         lcd_putc("\fGood Morning");
571                         first = 0;
572                     }
573                     if (input(PIN_B4) == 1){
574                         lcd_putc("\f <=");
575                         send_action(0x74);
576                         first = 1;

```

```

577
578     if(season == 1) temp_h = temp_h-2;
579     else temp_h = temp_h+2;
580
581     short int status;
582     do{
583         i2c_start();
584         i2c_write(0x40); //Address of pic
585         i2c_write(0xbb); //Instruction to change temp
586         i2c_write(temp_h);
587         delay_ms(10);
588         status = i2c_write(temp_l);
589         i2c_stop();
590         if (status == 1) delay_ms(100); //If data not read properly try
           again with delay
591     }
592     while ( status == 1);
593
594     //New day = Reset all devices to automatic (user's choice)
595     if (ToC != 1){
596         ToC = 1; //Everything back to auto
597         change_data(AToC, ToC);
598     }
599     if (AirC != 1){
600         AirC = 1;
601         change_data(AAir, AirC);
602     }
603     if(HeatC != 1){
604         HeatC = 1;
605         change_data(AHeat, HeatC);
606     }
607
608     delay_ms(300);
609     return;
610 }
611 break;
612 case 2:
613     if (first == 1 && state == 2){
614         lcd_putc("\fGood Night");
615         first = 0;
616     }
617     if (input(PIN_B4) == 1){
618         lcd_putc("<=");
619         //Good night
620         send_action(0x73);
621
622         //Send temp
623         if(season == 1) temp_h = temp_h+2; //In warm season temp can be a
           bit higher during night
624         else temp_h = temp_h-2;
625
626         short int status;
627         do{
628             i2c_start();
629             i2c_write(0x40); //Address of pic
630             i2c_write(0xbb); //Instruction to change temp
631             i2c_write(temp_h);
632             delay_ms(10);
633             status = i2c_write(temp_l);
634             i2c_stop();
635             if (status == 1) delay_ms(100); //If data not read properly try
           again with delay
636         }
637         while ( status == 1);
638
639         //Set lights to auto again
640         LC1=1;
641         LC2=1;
642         LC3=1;
643         change_data(ALC1,LC1);
644         delay_ms(10);
645         change_data(ALC2,LC2);
646         delay_ms(10);

```

```

647         change_data(ALC3,LC3);
648         delay_ms(10);
649
650         first = 1;
651         state = 1; //Show GM option
652         delay_ms(300);
653         return;
654     }
655     break;
656 case 3:
657     if (first == 1 && state == 3){
658         lcd_putc("\fOut Mode");
659         first = 0;
660     }
661     if (input(PIN_B4) == 1){
662         first = 1;
663         lcd_putc("\fOut Mode <=");
664         disable_interrupts(int_timer1); //Desactivar que se envie el
        //cambio de estado
        //Change mode
        change_data(0xcc,0x20); //Modo OUT
665
666         //Change lights to auto
667         LC1=1;
668         LC2=1;
669         LC3=1;
670         change_data(ALC1,LC1);
671         delay_ms(10);
672         change_data(ALC2,LC2);
673         delay_ms(10);
674         change_data(ALC3,LC3);
675         delay_ms(10);
676
677         //Lower/higher the temp while outside depending on the season
678         if(season == 1) temp_h = temp_h+2; //In warm season temp can be a
679         bit higher during night
680         else temp_h = temp_h-2;
681         short int status;
682         do{
683             i2c_start();
684             i2c_write(0x40); //Address of pic
685             i2c_write(0xbb); //Instruction to change temp
686             i2c_write(temp_h);
687             delay_ms(10);
688             status = i2c_write(temp_l);
689             i2c_stop();
690             if (status == 1) delay_ms(20); //If data not read properly try
691             again with delay
692         }
693         while ( status == 1);
694         delay_ms(200);
695
696         //Ask for new pass
697         int i; //For loops
698         char data[3]; //store eeprom data
699         char clave[3]; //store keyboard data
700         lcd_putc("\fSystem blocked\nPassword:");
701         while(TRUE){
702             if (first == 1){
703                 first = 0;
704                 lcd_putc("\fSystem blocked\nPassword:");
705                 i = 0;
706                 while (i<=2){
707                     k = kbd_getc();
708                     if (k!=0){
709                         i++;
710                         clave[i]=k;
711                         lcd_putc("*");
712                     }
713                 }
714             }
715         }
716         if (input(PIN_B4) == 1){ //OKAY Pressed

```

```

717     int h;
718     for (h=0;h<=2;h++) data[h] = read_eeprom(h);
719     if ((clave[0]==data[0]) || (clave[1]==data[1]) ||
        (clave[2]==data[2]))
720     {
721         lcd_putc("\fPassword correct\nWelcome home");
722         enable_interrupts(int_timer1);
723         change_data(0xcc,0x10); //Modo auto
724
725         //Go back to normal temp
726         if(season == 1) temp_h = temp_h-2;
727         else temp_h = temp_h+2;
728
729         short int status;
730         do{
731             i2c_start();
732             i2c_write(0x40); //Address of pic
733             i2c_write(0xbb); //Instruction to change temp
734             i2c_write(temp_h);
735             delay_ms(10);
736             status = i2c_write(temp_h);
737             i2c_stop();
738             if (status == 1) delay_ms(100); //If data not read
                properly try again with delay
739         }
740         while ( status == 1);
741
742
743
744         //Ask for any alarm
745         do{
746             i2c_start();
747             i2c_write(0x40);
748             status = i2c_write(0x15);
749             i2c_stop();
750             if (status == 1) delay_ms(50); //If no answer wait 1s
751         }
752         while (status == 1);
753
754         i2c_start();
755         i2c_write(0x41);
756         alarm = i2c_read(0);
757         i2c_stop();
758
759
760         if (bit_test(alarm,0) == 1 || bit_test(alarm,1) == 1){
761             lcd_putc("\fAlarma in\n");
762             if (bit_test(alarm,0) == 1) lcd_putc("Windw ");
763             if (bit_test(alarm,1) == 1) lcd_putc("Inside");
764             while(bit_test(alarm,0) == 1 || bit_test(alarm,1) == 1){
765                 if (input(PIN_B4) == 1){
766                     send_action(0xa4); //clear alarm
767                     bit_clear(alarm,0);
768                     bit_clear(alarm,1);
769                 }
770             }
771         }
772
773         delay_ms(500);
774         break;
775     }
776     else{
777         lcd_putc("\fWrong password");
778         first = 1;
779         delay_ms(500);
780     }
781 }
782 }
783 first = 1;
784 state = 3;
785 return;
786 }
787 break;

```

```

788         case 10: state = 3; first = 1; return; break;
789         default: state = 1; first = 1; break;
790     }
791 }
792
793 }
794 }
795
796
797
798 void config_luz(void);
799
800 void menu_config(void){
801     if (first == 1 && state == 4){
802         lcd_putc("\fConfig \ndevice?");
803         first = 0;
804     }
805
806     if (input(PIN_B4) == 1){ //OKAY pressed
807         first = 1;
808         state = 1;
809         lcd_putc("<=");
810         delay_ms(150);
811         while(TRUE){
812             switch(state){
813                 case 0: state = 5; first = 1; break;
814                 case 1:
815                     config_luz();
816                     break;
817                 case 2:
818                     if (first == 1 && state == 2){
819                         first = 0;
820                         lcd_putc("\fToldo:");
821                         lcd_gotoxy(1,2);
822                         if (ToC == 1) lcd_putc("A");
823                         else if (ToC == 2) lcd_putc("UP");
824                         else if (ToC == 3) lcd_putc("DOWN");
825                         else lcd_putc("STOP");
826                     }
827                     if (input(PIN_B4) == 1){ //Okay pressed
828                         state = ToC;
829                         first = 1;
830                         lcd_gotoxy(8,1);
831                         lcd_putc("<=");
832                         delay_ms(100);
833                         while(TRUE){
834                             switch(state){
835                                 case 0: state = 4; break;
836                                 case 1: //Automatic
837                                     if (first == 1 && state == 1){
838                                         first = 0;
839                                         lcd_putc("\fToldo: <=\nA");
840                                     }
841                                     if (input(PIN_B4) == 1){ //Okay = Summit choice
842                                         ToC = 1;
843                                         lcd_putc("<=");
844                                         first = 1;
845                                         delay_ms(300);
846                                         state = 2;
847                                         change_data(AToC,ToC);
848                                         return;
849                                     }
850                                     break;
851                                 case 2: //On
852                                     if (first == 1 && state == 2){
853                                         first = 0;
854                                         lcd_putc("\fToldo: <=\nUP");
855                                     }
856                                     if (input(PIN_B4) == 1){ //Okay = Summit choice
857                                         ToC = 2;
858                                         lcd_putc("<=");
859                                         first = 1;

```



```

861         delay_ms(300);
862         state = 2;
863         change_data(AToC,ToC);
864         return;
865     }
866     break;
867
868     case 3: //OFF
869         if (first == 1 && state == 3){
870             first = 0;
871             lcd_putc("\fToldo: <=\nDOWN");
872         }
873         if (input(PIN_B4) == 1){ //Okay = Summit choice
874             ToC = 3;
875             lcd_putc(" <=");
876             first = 1;
877             delay_ms(300);
878             state = 2;
879             change_data(AToC,ToC);
880             return;
881         }
882         break;
883     case 4:
884         if (first == 1 && state == 4){
885             first = 0;
886             lcd_putc("\fToldo: <=\nSTOP");
887         }
888         if (input(PIN_B4) == 1){ //Okay = Summit choice
889             ToC = 4;
890             lcd_putc(" <=");
891             first = 1;
892             delay_ms(300);
893             state = 2;
894             change_data(AToC,ToC);
895             return;
896         }
897         break;
898
899     case 10:
900         state = 2; //Volver al estado general
901         return;
902         break;
903
904     default: state = 1; break;
905 }
906 }
907 }
908 break;
909 case 3:
910     config_air();
911     break;
912 case 4:
913     if (first == 1 && state == 4){
914         first = 0;
915         lcd_putc("\fHeat:");
916         lcd_gotoxy(1,2);
917         if (HeatC == 1) lcd_putc("A");
918         else if (HeatC == 2) lcd_putc("ON");
919         else lcd_putc("OFF");
920     }
921     if (input(PIN_B4) == 1){ //Okay pressed
922         state = HeatC;
923         first = 1;
924         lcd_gotoxy(7,1);
925         lcd_putc("<=");
926         delay_ms(100);
927         while(TRUE){
928             switch(state){
929                 case 0: state = 3; break;
930                 case 1:
931                     if (first == 1 && state == 1){
932                         first = 0;
933                         lcd_putc("\fHeat: <=\nA");

```

```

934     }
935     if (input(PIN_B4) == 1){ //Okay = Summit choice
936         HeatC = 1;
937         lcd_putc(" <=");
938         first = 1;
939         state = 4;
940         change_data(AHeat,HeatC);
941         delay_ms(300);
942         return;
943     }
944     break;
945 case 2:
946     if (first == 1 && state == 2){
947         first = 0;
948         lcd_putc("\fHeat: <=\nON");
949     }
950     if (input(PIN_B4) == 1){ //Okay = Summit choice
951         HeatC = 2;
952         lcd_putc(" <=");
953         first = 1;
954         state = 4;
955         change_data(AHeat,HeatC);
956         delay_ms(300);
957         return;
958     }
959     break;
960 case 3:
961     if (first == 1 && state == 3){
962         first = 0;
963         lcd_putc("\fHeat: <=\nOFF");
964     }
965     if (input(PIN_B4) == 1){ //Okay = Summit choice
966         HeatC = 3;
967         lcd_putc(" <=");
968         first = 1;
969         state = 3;
970         change_data(AHeat,HeatC);
971         delay_ms(300);
972         return;
973     }
974     break;
975 case 10:
976     state = 4;
977     return;
978     break;
979 default:state = 1; break;
980 }
981 }
982 }
983
984
985 break;
986 case 5:
987     if (first == 1 && state == 5){
988         first = 0;
989         lcd_putc("\fBlinds:");
990         lcd_gotoxy(1,2);
991         if (BlindsC == 1) lcd_putc("A");
992         else if (BlindsC == 2) lcd_putc("UP");
993         else if (BlindsC == 3) lcd_putc("DOWN");
994         else lcd_putc("STOP");
995     }
996     if (input(PIN_B4) == 1){ //Okay pressed
997         state = ToC;
998         first = 1;
999         lcd_gotoxy(9,1);
1000        lcd_putc("<=");
1001        delay_ms(100);
1002        while(TRUE){
1003            switch(state){
1004                case 0: state = 4; break;
1005                case 1: //Automatic
1006                    if (first == 1 && state == 1){

```

```

1007         first = 0;
1008         lcd_putc("\fBlinds: <=\nA");
1009     }
1010     if (input(PIN_B4) == 1){ //Okay = Summit choice
1011         BlindsC = 1;
1012         lcd_putc(" <=");
1013         first = 1;
1014         delay_ms(300);
1015         state = 5;
1016         change_data(ABlindsC,BlindsC);
1017         return;
1018     }
1019     break;
1020
1021     case 2: //On
1022         if (first == 1 && state == 2){
1023             first = 0;
1024             lcd_putc("\fBlinds: <=\nUP");
1025         }
1026         if (input(PIN_B4) == 1){ //Okay = Summit choice
1027             BlindsC = 2;
1028             lcd_putc(" <=");
1029             first = 1;
1030             delay_ms(300);
1031             state = 5;
1032             change_data(ABlindsC,BlindsC);
1033             return;
1034         }
1035         break;
1036
1037     case 3: //OFF
1038         if (first == 1 && state == 3){
1039             first = 0;
1040             lcd_putc("\fBlinds: <=\nDOWN");
1041         }
1042         if (input(PIN_B4) == 1){ //Okay = Summit choice
1043             BlindsC = 3;
1044             lcd_putc(" <=");
1045             first = 1;
1046             delay_ms(300);
1047             state = 5;
1048             change_data(ABlindsC,BlindsC);
1049             return;
1050         }
1051         break;
1052     case 4:
1053         if (first == 1 && state == 4){
1054             first = 0;
1055             lcd_putc("\fBlinds: <=\nSTOP");
1056         }
1057         if (input(PIN_B4) == 1){ //Okay = Summit choice
1058             BlindsC = 4;
1059             lcd_putc(" <=");
1060             first = 1;
1061             delay_ms(300);
1062             state = 5;
1063             change_data(ABlindsC,BlindsC);
1064             return;
1065         }
1066         break;
1067
1068     case 10:
1069         state = 5; //Volver al estado general
1070         return;
1071         break;
1072
1073     default: state = 1; break;
1074 }
1075 }
1076 }
1077 break;
1078 case 10: state = 4; first = 1; return; break; //Reset menu and global
state variable

```

```

1079         default: state = 1; first = 1; break;
1080     }
1081 }
1082
1083 }
1084
1085 }
1086
1087 void config_luz(void){
1088     if (first == 1 && state == 1){
1089         first = 0;
1090         lcd_putc("\fLights: ");
1091         lcd_putc("EXT:");
1092         if (LC1 == 1){
1093             lcd_putc("A");
1094             if(bit_test(Lstate,2)==1) lcd_putc("O ");
1095             else lcd_putc("X ");
1096         }
1097         else if (LC1 == 2) lcd_putc("MO ");
1098         else lcd_putc("MX ");
1099
1100         lcd_gotoxy(1,2);
1101         lcd_putc("CR:");
1102         if (LC2 == 1){
1103             lcd_putc("A");
1104             if(bit_test(Lstate,1)==1) lcd_putc("O ");
1105             else lcd_putc("X ");
1106         }
1107         else if (LC2 == 2) lcd_putc("MO ");
1108         else lcd_putc("MX ");
1109         lcd_putc("TL:");
1110         if (LC3 == 1){
1111             lcd_putc("A");
1112             if(bit_test(Lstate,0)==1) lcd_putc("O ");
1113             else lcd_putc("X ");
1114         }
1115         else if (LC3 == 2) lcd_putc("MO ");
1116         else lcd_putc("MX ");
1117     }
1118
1119     if (input(PIN_B4) == 1){ //Okay pressed
1120         first = 1;
1121         state = 1;
1122         delay_ms(150);
1123         while(TRUE){
1124             switch(state){
1125                 case 0: state = 3; first = 1; break;
1126                 case 1:
1127                     exterior();
1128                     break;
1129                 case 2:
1130
1131                     if (first == 1 && state == 2){
1132                         first = 0;
1133                         lcd_putc("\fCorridor:");
1134                         lcd_gotoxy(1,2);
1135                         if (LC2 == 1) lcd_putc("A");
1136                         else if (LC2 == 2) lcd_putc("ON");
1137                         else lcd_putc("OFF");
1138                     }
1139                     if (input(PIN_B4) == 1){ //Okay pressed
1140                         state = LC2;
1141                         first = 1;
1142                         lcd_gotoxy(11,1);
1143                         lcd_putc("<=");
1144                         delay_ms(100);
1145                         while(TRUE){
1146                             switch(state){
1147                                 case 0: state = 3; break;
1148                                 case 1: //Automatic
1149                                     if (first == 1 && state == 1){
1150                                         first = 0;
1151                                         lcd_putc("\fCorridor: <=\nA");

```

```

1152     }
1153     if (input(PIN_B4) == 1){ //Okay = Summit choice
1154         LC2 = 1;
1155         bit_set(Lstate,1);
1156         lcd_putc(" <=");
1157         first = 1;
1158         state = 1;
1159         change_data(ALC2,LC2);
1160         delay_ms(300);
1161         return;
1162     }
1163     break;
1164
1165     case 2: //On
1166         if (first == 1 && state == 2){
1167             first = 0;
1168             lcd_putc("\fCorridor: <=\nM ON");
1169         }
1170         if (input(PIN_B4) == 1){ //Okay = Summit choice
1171             LC2 = 2;
1172             bit_clear(Lstate,1);
1173             lcd_putc(" <=");
1174             first = 1;
1175             state = 1;
1176             change_data(ALC2,LC2);
1177             delay_ms(300);
1178             return;
1179         }
1180         break;
1181
1182     case 3: //OFF
1183         if (first == 1 && state == 3){
1184             first = 0;
1185             lcd_putc("\fCorridor: <=\nM OFF");
1186         }
1187         if (input(PIN_B4) == 1){ //Okay = Summit choice
1188             LC2 = 3;
1189             lcd_putc(" <=");
1190             first = 1;
1191             state = 1;
1192             change_data(ALC2,LC2);
1193             delay_ms(300);
1194             return;
1195         }
1196         break;
1197
1198     case 10: //Esc pressed
1199         state = 1;
1200         return;
1201         break;
1202
1203     default: state = 1; break;
1204 }
1205 }
1206 }
1207
1208
1209
1210 break;
1211 case 3:
1212
1213     if (first == 1 && state == 3){
1214         first = 0;
1215         lcd_putc("\fToilet:");
1216         lcd_gotoxy(1,2);
1217         if (LC3 == 1) lcd_putc("A");
1218         else if (LC3 == 2) lcd_putc("ON");
1219         else lcd_putc("OFF");
1220     }
1221     if (input(PIN_B4) == 1){ //Okay pressed
1222         state = LC3;
1223         first = 1;
1224         lcd_gotoxy(9,1);

```

```

1225     lcd_putc("<=");
1226     delay_ms(100);
1227     while(TRUE){
1228         switch(state){
1229             case 0: state = 3; break;
1230             case 1: //Automatic
1231                 if (first == 1 && state == 1){
1232                     first = 0;
1233                     lcd_putc("\fToilet: <=\nA");
1234                 }
1235                 if (input(PIN_B4) == 1){ //Okay = Summit choice
1236                     LC3 = 1;
1237                     lcd_putc(" <=");
1238                     first = 1;
1239                     state = 1;
1240                     change_data(ALC3,LC3);
1241                     delay_ms(300);
1242                     return;
1243                 }
1244                 break;
1245             case 2: //On
1246                 if (first == 1 && state == 2){
1247                     first = 0;
1248                     lcd_putc("\fToilet: <=\nM ON");
1249                 }
1250                 if (input(PIN_B4) == 1){ //Okay = Summit choice
1251                     LC3 = 2;
1252                     bit_set(Lstate,0);
1253                     lcd_putc(" <=");
1254                     first = 1;
1255                     state = 1;
1256                     change_data(ALC3,LC3);
1257                     delay_ms(300);
1258                     return;
1259                 }
1260                 break;
1261             case 3: //OFF
1262                 if (first == 1 && state == 3){
1263                     first = 0;
1264                     lcd_putc("\fToilet: <=\nM OFF");
1265                 }
1266                 if (input(PIN_B4) == 1){ //Okay = Summit choice
1267                     LC3 = 3;
1268                     bit_clear(Lstate,0);
1269                     lcd_putc(" <=");
1270                     first = 1;
1271                     state = 1;
1272                     change_data(ALC3,LC3);
1273                     delay_ms(300);
1274                     return;
1275                 }
1276                 break;
1277             case 10: //Esc pressed
1278                 state = 1;
1279                 return;
1280                 break;
1281             default: state = 1; break;
1282         }
1283     }
1284     break;
1285     case 10: state = 1; first = 1; return; break;
1286     default: state = 1; first = 1; break;
1287 }
1288 }
1289 }
1290 }
1291 }
1292 }
1293 }
1294 }
1295 }
1296 }
1297 }

```

```

1298
1299 void exterior(void){
1300     if (first == 1 && state == 1){
1301         first = 0;
1302         lcd_putc("\fExterior:");
1303         lcd_gotoxy(1,2);
1304         if (LC1 == 1) lcd_putc("A");
1305         else if (LC1 == 2) lcd_putc("ON");
1306         else lcd_putc("OFF");
1307     }
1308
1309     if (input(PIN_B4) == 1){ //Okay pressed
1310         state = LC1;
1311         first = 1;
1312         lcd_gotoxy(11,1);
1313         lcd_putc("<=");
1314         delay_ms(100);
1315         while(TRUE){
1316             switch(state){
1317                 case 0: state = 3; break;
1318                 case 1: //Automatic
1319                     if (first == 1 && state == 1){
1320                         first = 0;
1321                         lcd_putc("\fExterior: <=\nA");
1322                     }
1323                     if (input(PIN_B4) == 1){ //Okay = Summit choice
1324                         LC1 = 1;
1325                         lcd_putc(" <=");
1326                         first = 1;
1327                         state = 1;
1328                         change_data(ALC1,LC1);
1329                         delay_ms(300);
1330                         return;
1331                     }
1332                     break;
1333
1334                 case 2: //On
1335                     if (first == 1 && state == 2){
1336                         first = 0;
1337                         lcd_putc("\fExterior: <=\nM ON");
1338                     }
1339                     if (input(PIN_B4) == 1){ //Okay = Summit choice
1340                         LC1 = 2;
1341                         bit_set(Lstate,2);
1342                         lcd_putc(" <=");
1343                         first = 1;
1344                         state = 1;
1345                         change_data(ALC1,LC1);
1346                         delay_ms(300);
1347                         return;
1348                     }
1349                     break;
1350
1351                 case 3: //OFF
1352                     if (first == 1 && state == 3){
1353                         first = 0;
1354                         lcd_putc("\fExterior: <=\nM OFF");
1355                     }
1356                     if (input(PIN_B4) == 1){ //Okay = Summit choice
1357                         LC1 = 3;
1358                         bit_clear(Lstate,2);
1359                         lcd_putc(" <=");
1360                         first = 1;
1361                         state = 1;
1362                         change_data(ALC1,LC1);
1363                         delay_ms(300);
1364                         return;
1365                     }
1366                     break;
1367
1368                 case 10: //Esc pressed
1369                     state = 1;
1370                     return;

```

```

1371         break;
1372
1373         default: state = 1; break;
1374     }
1375 }
1376 }
1377
1378 }
1379
1380
1381 void config_toldo(void){
1382
1383 }
1384
1385 void config_air(void){
1386     if (first == 1 && state == 3){
1387         first = 0;
1388         lcd_putc("\fAir:");
1389         lcd_gotoxy(1,2);
1390         if (AirC == 1) lcd_putc("A");
1391         else if (AirC == 2) lcd_putc("ON");
1392         else lcd_putc("OFF");
1393     }
1394     if (input(PIN_B4) == 1){ //Okay pressed
1395         state = AirC;
1396         first = 1;
1397         lcd_gotoxy(6,1);
1398         lcd_putc("<=");
1399         delay_ms(100);
1400         while(TRUE){
1401             switch(state){
1402                 case 0: state = 3; break;
1403                 case 1: //Automatic
1404                     if (first == 1 && state == 1){
1405                         first = 0;
1406                         lcd_putc("\fAir: <=\nA");
1407                     }
1408                     if (input(PIN_B4) == 1){ //Okay = Summit choice
1409                         AirC = 1;
1410                         lcd_putc("<=");
1411                         first = 1;
1412                         state = 3;
1413                         change_data(AAir,AirC);
1414                         delay_ms(300);
1415                         return;
1416                     }
1417                     break;
1418
1419                 case 2: //On
1420                     if (first == 1 && state == 2){
1421                         first = 0;
1422                         lcd_putc("\fAir: <=\nON");
1423                     }
1424                     if (input(PIN_B4) == 1){ //Okay = Summit choice
1425                         AirC = 2;
1426                         lcd_putc("<=");
1427                         first = 1;
1428                         state = 3;
1429                         change_data(AAir,AirC);
1430                         delay_ms(300);
1431                         return;
1432                     }
1433                     break;
1434
1435                 case 3: //OFF
1436                     if (first == 1 && state == 3){
1437                         first = 0;
1438                         lcd_putc("\fAir: <=\nOFF");
1439                     }
1440                     if (input(PIN_B4) == 1){ //Okay = Summit choice
1441                         AirC = 3;
1442                         lcd_putc("<=");
1443                         first = 1;

```



```

1444         state = 3;
1445         change_data(AAir,AirC);
1446         delay_ms(300);
1447         return;
1448     }
1449     break;
1450
1451     case 10: //Esc pressed
1452         state = 3;
1453         return;
1454         break;
1455
1456     default: state = 1; break;
1457 }
1458 }
1459 }
1460
1461 }
1462
1463
1464
1465 void change_data(int address, int number){
1466     BYTE status1, status2;
1467     do{
1468         i2c_start();
1469         i2c_write(0x40);
1470         status1 = i2c_write(address);
1471         delay_ms(10);
1472         status2 = i2c_write(number);
1473         i2c_stop();
1474         if (status1 == 1 || status2 == 1) delay_ms(40);
1475     }
1476     while(status1 == 1 || status2 == 1);
1477 }
1478
1479 void send_action(int number){
1480     BYTE status;
1481     do{
1482         i2c_start();
1483         i2c_write(0x40);
1484         status = i2c_write(number);
1485         i2c_stop();
1486         if (status == 1) delay_ms(40);
1487     }
1488     while(status == 1);
1489 }
1490
1491
1492 void update_time(void)
1493 {
1494     if (hr < 12){ //De las 00 a 12 de la mañana mas barato
1495         if (time_change == 1){ //Before it was day
1496             send_action(0x67); // cheap mode
1497             time_change = 3;
1498         }
1499     }
1500     else if (time_change == 3) { //outside of night span and before it was night
1501         send_action(0x68); //out day
1502         time_change = 1;
1503     } //Day mode
1504
1505     if ( month >= 9 && month <=4 && season == 1){ //Cold season starts in october
1506         season = 2;
1507         send_action(0x66);
1508     }
1509     else if (month >= 5 && month <= 8 && season == 2){ //Warm season starts in may
1510         season = 1;
1511         send_action(0x65);
1512     }
1513 }
1514

```

## 1.2. Configuraciones

```

1  #include <stdio.h>
2  #fuses XT,NOVDT,PUT,NOPROTECT,NOLVP
3  #use delay(clock = 4000000)
4
5  //LCD init.
6  #define LCD_ENABLE PIN   PIN_E0
7  #define LCD_RS PIN      PIN_E1
8  #define LCD_RW PIN      PIN_E2
9  #define LCD_DATA4      PIN_A2
10 #define LCD_DATA5      PIN_A3
11 #define LCD_DATA6      PIN_A4
12 #define LCD_DATA7      PIN_A5
13 #include <lcd.c>
14
15 //KBD init.
16 #include <kbd.c>
17
18 //Outputs
19 #BYTE TRISA = 0x85
20 #BYTE PORTA = 0x05
21 #BYTE TRISB = 0x86
22 #BYTE PORTB = 0x06
23
24 //rs232
25 #BYTE TRISC = 0x87
26 #use rs232(baud=9600, bits=8, parity = N, xmit=PIN_C6, rcv=PIN_C7, stream = PC)
27
28 //I2C
29 #use i2c(master, sda=PIN_C4, scl=PIN_C5)
30 #include <EEPROM_24512.c>
31 #include <DS1307.c>
32 #include <Float_EE.c>
33
34 //Global Variables
35 #rom 0xF00000 = {'3','5','7'} //Contrasenya porta
36
37 //Global functions //////////////////////////////////////////////////
38 void init_routine(void);
39
40 //Menu principal
41 void menu_temperatura(void);
42 void menu_potencia(void);
43 void menu_mode(void);
44 void menu_config(void);
45 void menu_clock(void);
46
47 //Menu mode
48 void modo_out (void);
49 void good_morning(void);
50 void good_night(void);
51
52 //Menu config
53 //void config_luz(void);
54 void config_toldo(void);
55 void config_air(void);
56 //void control_heat(void);
57
58 //Menu luz
59 void exterior(void);
60 //void corredor(void);
61 void toilet(void);
62
63 //Funciones externas de control
64 void update_time(void);
65 void change_data(int address, int number);
66 void send_action(int number);

```

## 1.3. Código Eeprom 24512

### 1.3.1. Leer/Escribir datos

```

1 void write_ext_eeprom(long int address, BYTE data)
2 {
3     short int status;
4     i2c_start();           //Inicializa la Transmisión
5     i2c_write(0xa0);       //Escribe la palabra de control(direccion
6                             //0h + 0 para escritura)
7     i2c_write(address>>8); //Parte alta de la dirección a escribir en la eeprom
8     i2c_write(address);    //Parte baja de la dirección a escribir en la eeprom
9     i2c_write(data);       //Dato a escribir
10    i2c_stop();            //Finalización de la transmisión
11    i2c_start();           //Reinicio
12    status=i2c_write(0xa0); //Lectura del bit ACK, para evitar escrituras
13                             //incorrectas
14    while(status==1)       //Si es "1" se espera para responder al esclavo
15    {
16        i2c_start();
17        status=i2c_write(0xa0);
18    }
19 }
20
21
22 BYTE read_ext_eeprom(long int address) {
23     BYTE data;
24     i2c_start();           //Inicializa la Transmisión
25     i2c_write(0xa0);       //Escribe la palabra de control(direccion
26                             //0h + 0 para escritura)
27     i2c_write(address>>8); //Parte alta de la dirección a escribir en la eeprom
28     i2c_write(address);    //Parte baja de la dirección a escribir en la eeprom
29     i2c_start();           //Reinicio
30     i2c_write(0xa1);       //Escribe la palabra de control(direccion
31                             //0h + 1 para escritura)
32     data=i2c_read(0);      //Lectura del Dato
33     i2c_stop();            //Finalización de la transmisión
34     return(data);
35 }
36

```

### 1.3.2. Leer/Escribir floats

```

1 void WRITE_FLOAT_EXT_EEPROM(long int n, float data) {
2     int i;
3
4     for (i = 0; i < 4; i++)
5         write_ext_eeprom(i + n, *((int8 *)(&data) + i));
6 }
7
8 float READ_FLOAT_EXT_EEPROM(long int n) {
9     int i;
10    float data;
11
12    for (i = 0; i < 4; i++)
13        *((int8 *)(&data) + i) = read_ext_eeprom(i + n);
14
15    return(data);
16 }

```

## 1.4. Código Reloj DS1307

```

1  int BCDA2BIN(BYTE bcd)
2  {
3  int varia;
4  varia=bcd;
5  varia>>=1;
6  varia &= 0x78;
7  return (varia + (varia>>2) + (bcd & 0x0f));
8  }
9
10 BYTE DECA2BCD(int DEC)
11 {
12 return(((DEC/10) << 4) + (DEC % 10));
13 }
14
15 void rtc_get_time(BYTE& yr, BYTE& month, BYTE& date, BYTE& day, BYTE& hr, BYTE& min,
16 BYTE& sec) {
17 i2c_start(); //Escritura
18 i2c_write(0xD0); //Codigo de escritura
19 i2c_write(0x00); //Puntero de la primera direcci3n
20 i2c_start(); //Lectura
21 i2c_write(0xD1); //Codigo de lectura
22 sec = BCDA2BIN(i2c_read() & 0x7f); //Lectura de los 7 bit de los segundos
23 min = BCDA2BIN(i2c_read() & 0x7f); //Lectura de los 7 bit de los minutos
24 hr = BCDA2BIN(i2c_read() & 0x3f); //Lectura de los 6 bit de las horas
25 day = BCDA2BIN(i2c_read() & 0x07); //Lectura de los 3 bits de dia de la semana
26 date = BCDA2BIN(i2c_read() & 0x3f); //Lectura de dia del mes
27 month = BCDA2BIN(i2c_read() & 0x1f); //Lectura de los 5 bits de los meses
28 yr = BCDA2BIN(i2c_read(0)); //Lectura de los a3os
29 i2c_stop(); //Finaliza comunicaci3n
30 }
31 void rtc_write(unsigned int8 address, int data){
32 i2c_start(); // Start I2C
33 i2c_write(0xD0); // DS1307 address
34 i2c_write(address); // Send register address
35 i2c_write(data); // Write data to the selected register
36 i2c_stop();
37 }

```

## 2. Código Módulo Energía en C

```

1  // Autor: ORIOL RIERA ROVIRA //
2  // SISTEMA DE GESTIÓN DOMÓTICA PARA OPTIMIZAR EL CONSUMO ENERGÉTICO DE UNA
  // VIVIENDA //
3  // TRABAJO DE FINAL DE GRADO //
4
5  #include <18F4550.h>
6  #device adc=10
7  #device HIGH_INTS=TRUE
8  #fuses XT, NOWDT
9  #use delay (clock=4000000)
10
11 #include <math.h>
12
13 #BYTE TRISA = 0x85
14 #BYTE TRISB = 0x86
15 #BYTE TRISC = 0x87
16 #BYTE TRISD = 0x88
17 #BYTE TRISE = 0x8A
18 #BYTE PORTA = 0x05
19 #BYTE PORTB = 0x06
20 #BYTE PORTC = 0x07
21 #BYTE PORTD = 0x08
22
23 #define ENCODER_P PIN_B6
24 #define L1 PIN_D0
25 #define L2 PIN_D3
26 #define L3 PIN_B2
27 #define ENCODER_T PIN_B5
28 #define T_UP PIN_D4
29 #define T_DOWN PIN_D1
30 #define P_UP PIN_E0
31 #define P_DOWN PIN_E1
32
33 //I2C Slave mode to comm with central
34 #use i2c(slave, SDA=PIN_B0, SCL=PIN_B1, ADDRESS=0x40, STREAM = TOTAL, FORCE HW)
35
36 //I2C Master mode to comm with sensor
37 #use I2C(master, SDA = PIN_C5, SCL = PIN_C4, STREAM = LOCAL)
38 #include <TEMP_DS1621_AUX.c>
39
40 // Global Variables
41 int state = 0x10, data in, alarm = 0;
42 int level_t = 0, level_p = 0; //level toldo/persiana
43 int level_t_d = 0, level_p_d = 0; //level desired
44
45 int light;
46
47 float P;
48
49 int LC1 = 1; //Luz 1 automatica
50 int LC2 = 1; //Luz 2 automatica
51 int LC3 = 1; //Luz 3 automatica
52 int ToC = 1; //Toldo automatico
53 int BlindsC = 1; //Persiana automatica
54 int AirC = 1; //Aire en modo automatico
55 int HeatC = 1; //Heat in auto mode
56
57 //RTOS mode
58 #use rtos(timer = 1, minor_cycle = 100ms)
59
60
61 // CONTROL DE TEMPERATURA //
62 #task(rate = 400ms, max = 100ms)
63 void control_temp(void);
64
65 //Variables temperatura
66 int temp_h,temp_l;
67 float temp_s = 0; //T Sensada
68 float temp_d = 24.0; //T Definida
69
70 void mover persiana(int posicion);
71
72 void control_temp(void){

```

```

73
74 //Leer temperatura
75 init_temp(0x03);
76 delay_ms(10);
77 temp_s =read_full_temp(0x03); //Lee temperatura del DS1621 - Temperatura sensada
78
79 if (HeatC == 1){ //Heat in auto mode
80     if ((temp_s<(temp_d-1)) && HeatC == 1 && Temp_s<=18 && input(PIN_C6) == 1 &&
        input(PIN_C7) == 1 && P<5000){ //enciende la calefacción si la temp es 1 grado
            inferior a la deseada y modo automatico y menor de 18°
                output_high(PIN_C2);
        }
        else
            output_low(PIN_C2);
    }
86
87 if (AirC == 1){ //Air in auto mode
88     if ((temp_s>(temp_d+1)) && AirC == 1 && Temp_s>=26 && input(PIN_C6) == 1 &&
        input(PIN_C7) == 1 && P<5000){ //Encender aire si la temp sensada es 1 grado
            superior a la deseada y modo automatico y mayor de 26°
                output_high(PIN_C1);
            }
            else
                output_low(PIN_C1);
        }
94
95     rtos_yield();
96 }
97
98 ///////////////////////////////////////////////////
99 #task(rate = 1000ms, max = 10ms)
100 void control_luz(void){
101     set_adc_channel(1);
102     light = read_adc();
103     rtos_yield();
104 }
105
106 ///////////////////////////////////////////////////
107 //CONTROL DEL TOLDO EXTERIOR MEDIANTE SENSOR DE LUZ, VIENTO Y TEMPERATURA
108 ///////////////////////////////////////////////////
109 #task(rate = 400ms, max = 50ms)
110 void control_toldo(void);
111 void mover_toldo(int posicion);
112 void control_toldo(void){
113     //Lee viento
114     float wind;
115     set_adc_channel(2);
116     wind = read_adc();
117     wind = wind*5.0/1023.0; //Escalar a 0-5V
118     wind = wind*6.0; //Mide de 0 a 30km/h
119
120     if (wind < 10 && ToC == 1){ //No wind && Auto mode
121
122         if (light >175 || temp_s > 24) { //Si hace sol o temperatura superior a 24°:
123             baja toldo
124             mover_toldo(10); // Bajar toldo (al final)
125         }
126         else if (light < 100) // Si no hay luz sube toldo
127         {
128             mover_toldo(0); // Subir toldo (al principio)
129         }
130     }
131     else if (wind >= 10){ //aire fuerte = subir toldo
132         mover_toldo(0); // Subir toldo (al principio)
133     }
134
135     if ((temp_s >27 || temp_s < 15) && BlindsC == 1){ //Bajar persiana a la mitad en
136         temperaturas extremas
137         mover_persiana(4);
138     }
139
140     rtos_yield();

```

```

139
140 }
141
142
143 /////// CONTROL DE VENTANAS ///////////////
144 #task(rate = 600ms, max = 20ms)
145 void control_ventanas(void);
146 int before_w1 = 1, before_w2 = 1;
147
148 void control_ventanas(void){
149     if (input(PIN_C6) == 0){ //Control first window (0 = open)
150         bit_set(Alarm,0); //Define que hay una ventana abierta
151         if (state == 0x20 && before_w1 == 1){ //Modo out i s'obre finestra (no cuenta
152             que estuviera abierta): alarma
153             output_high(PIN_D2);
154         }
155         else{ //normal mode -> close air and heat
156             before_w1 = 0;
157         }
158     }
159
160     if (input(PIN_C7) == 0){
161         bit_set(Alarm,0); //Define que hay una ventana abierta
162         if (state == 0x20 && before_w2 == 1){ //Modo out i s'obre finestra (no cuenta
163             que estuviera abierta): alarma
164             output_high(PIN_D2);
165         }
166         else{ //normal mode -> close air and heat
167             before_w2 = 0;
168         }
169     }
170     else before_w2 = 1;
171
172     if (input(PIN_C7) == 1 && input(PIN_C6) == 1 && state == 0x10) bit_clear(Alarm,0);
173
174     rtos_yield();
175 }
176
177 /////// CONTROL CALIDAD DEL AIRE ///////////////
178 #task (rate = 5s, max = 20ms)
179 void control_air(void){
180     float air_q;
181     set_adc_channel(3);
182     air_q = read_adc();
183     air_q = air_q*5.0/1023.0; //Escalar a 0-5V
184     if(air_q>3)
185         bit_set(Alarm,2);
186     else
187         bit_clear(Alarm,2);
188 }
189
190 /////// CONTROL SENSOR DE PRESENCIA ///////
191 #task(rate = 1s, max = 10ms)
192 void control_presencia(void);
193 int cnt1 = 0, cnt2 = 0, cnt3 = 0;
194 void control_presencia(void){ //Si se deja de detectar i modo auto, apaga la luz
195     1(se enciende con interrupcion)
196     if (input(PIN_B4) == 0 && LC1 == 1 && input_state(PIN_D0) == 1){
197         cnt1++;
198         if (cnt1 >= 10){
199             output_low(PIN_D0);
200             cnt1 = 0;
201         }
202     }
203     if (input(PIN_B7) == 0 && LC2 == 1 && input_state(PIN_D3) == 1){
204         cnt2++;
205         if (cnt2 >= 10){
206             output_low(PIN_D3);
207             cnt2 = 0;
208         }
209     }
210     if (input(PIN_B2) == 0 && LC3 == 1 && input_state(PIN_B3) == 1){

```

```

209     cnt3++;
210     if (cnt3 >= 10){
211         output_low(PIN_B3);
212         cnt3 = 0;
213     }
214 }
215 rtos_yield();
216 }
217
218
219 ////// MEDICION DE POTENCIA ////////
220 #task(rate = 1000ms, max = 50ms)
221 void medir_potencia(void);
222
223 float getCorriente(int samples){
224     float voltage, corriente, ieficaz = 0;
225     float SENSIBILITY = 0.100; //modelo 20A
226     int i;
227     for (i=0;i<samples;i++){
228         set_adc_channel(0);
229         delay_us(20);
230         voltage = read_adc()*5.0/1023.0;
231         corriente = (voltage-2.5)/SENSIBILITY;
232         ieficaz += corriente*corriente; //Suma cuadratica
233     }
234     return (sqrt(ieficaz/samples)); //Devolver IRMS
235 }
236
237 void medir_potencia(void){
238     float Irms, Vrms;
239     //Irms = getCorriente(50); //when using the real sensor
240     set_adc_channel(0);
241     delay_us(20);
242     Irms = read_adc();
243     Irms = Irms*5.0/1023.0;
244     Irms = Irms*6.0;
245     set_adc_channel(4);
246     delay_us(20);
247     Vrms = read_adc();
248     Vrms = Vrms*5.0/1023.0;
249     Vrms = Vrms*60.0;
250     P = Irms*Vrms;
251     if(P<0) P = 0;
252
253     if (P>5000){
254         output_low(PIN_C1);
255         output_low(PIN_C2);
256     }
257
258     rtos_yield();
259 }
260
261 #int_ext2
262 void ext2_isr (void){
263     if (state == 0x20){ //Modo OUT i detecta == ALARMA
264         output_high(PIN_D2);
265         bit_set(alarm,1); //Save there was an alarm
266     }
267     else if (LC3 == 1){ //Modo automatic i detecta: encendre llums always
268         output_high(PIN_B3);
269     }
270 }
271
272
273 #int_rb
274 void rb_isr (void)
275 {
276     if (input(PIN_B4)==1){ // Sensor presencia 1: Exterior Light
277         if (LC1 == 1 && light < 100){ //Modo automatic i poca llum: encendre llums
278             (in any mode)
279             output_high(PIN_D0);
280         }
281     }
282 }

```



```

281     if (input(PIN_B5)==1){ // State == 0010 (encoder toldo)
282         if(input_state(T_UP) == 1){ //Mode UP
283             if(--level_t == level_t_d){ //Top is reached stop.
284                 output_low(T_UP);
285             }
286         }
287         else if (input_state(T_DOWN) == 1){ //Mode Down
288             if(++level_t >= level_t_d){ //Top is reached stop
289                 output_low(T_DOWN);
290             }
291         }
292     }
293     if (input(ENCODER_P)==1){ // State == 0100 (encoder persiana)
294         if(input_state(P_UP) == 1){ //Mode UP
295             if(--level_p == level_p_d){ //Top is reached stop
296                 output_low(P_UP);
297             }
298         }
299         else if (input_state(P_DOWN) == 1){ //Mode Down
300             if(++level_p >= level_p_d){ //Top is reached stop
301                 output_low(P_DOWN);
302             }
303         }
304     }
305     if (input(PIN_B7)==1){ // State == 1000 (Sensor presencia 2): Corridor light
306         if (state == 0x20){ //Modo OUT i detecta == ALARMA
307             output_high(PIN_D2);
308             bit_set(alarm,1); //Save there was an alarm
309         }
310         else if (LC2 == 1 && light < 100){ //automatic y poca luz = encenderla
311             output_high(PIN_D3);
312         }
313     }
314 }
315 }
316
317
318 // i2c comm with MASTER
319 void update_ToC(void);
320 void update_BlindsC(void);
321
322 #INT_SSP HIGH
323 void ssp_isr(void)
324 {
325     BYTE incoming, status;
326     status = i2c_isr_state(TOTAL);
327     if(status < 0x80) //Master is sending data
328     {
329         incoming = i2c_read(TOTAL);
330         if (status == 1){
331             data_in = incoming;
332             if (data_in == 0xa4){ //Clear alarm
333                 alarm = 0;
334                 output_low(PIN_D2); //Apagar alarma
335             }
336         }
337         else if (data_in == 0x74){ //rutine for morning
338             rtos_enable(control_toldo);
339             mover_persiana(0); //Subir persiana
340             data_in = 0;
341         }
342         else if (data_in == 0x73){ //Rutine for night
343             rtos_disable(control_toldo);
344             mover_toldo(0); //Subir toldo
345             mover_persiana(8); //Bajar persiana
346             output_low(L1);
347             output_low(L2);
348             output_low(L3);
349             data_in = 0;
350         }
351         else if (data_in == 0xcc){ //Change state
352             state = incoming;
353             data_in = 0;

```

```

354     if (state == 0x10){ //Normal mode && Welcome home
355         rtos_enable(control_toldo);
356         mover_persiana(0); //Subir persiana
357     }
358     else if(state == 0x20){ //OUT MODE
359         mover_toldo(0); //Subir toldo
360         mover_persiana(8); //Bajar persiana
361         rtos_disable(control_toldo);
362         output_low(PIN_D0); //Cerrar luces
363         output_low(PIN_D3);
364         output_low(PIN_B3);
365     }
366 }
367 else if (data_in == 0xBB){ //Define new temp desired
368     if (status == 2) temp_d = incoming*1.0;
369     else{
370         temp_d = temp_d + incoming/10.0 ;
371         data_in = 0;
372     }
373 }
374 else if(data_in == 0x68){ //Hour for high price
375     output_low(PIN_D6); //Apagar lavaplatos
376     output_low(PIN_D7); //Apagar lavadora
377     data_in = 0;
378 }
379 else if (data_in == 0x67){ //Low price hour
380     output_high(PIN_D6); // Activar lavaplatos
381     output_high(PIN_D7); //Activar lavadora
382     data_in = 0;
383 }
384 else if(data_in == 0xc6){
385     LC1 = incoming;
386     if (LC1 == 2){
387         output_high(PIN_D0);
388     }
389     else if (LC1 == 3){
390         output_low(PIN_D0);
391     }
392     data_in = 0;
393 }
394 else if(data_in == 0xc5){
395     ToC = incoming;
396     update_ToC();
397     data_in = 0;
398 }
399 else if(data_in == 0xc7){
400     BlindsC = incoming;
401     update_BlindsC();
402     data_in = 0;
403 }
404 else if(data_in == 0xc4){
405     AirC = incoming;
406     data_in = 0;
407     if (AirC == 3) output_low(PIN_C1); //OFF manual
408     else if (AirC == 2) output_high(PIN_C1); //ON manual
409 }
410 else if(data_in == 0xc3){
411     HeatC = incoming;
412     data_in = 0;
413     if (HeatC == 3) output_low(PIN_C2); //OFF manual
414     else if (HeatC == 2) output_high(PIN_C2); //ON manual
415 }
416 else if(data_in == 0xc2){
417     LC2 = incoming;
418     if (LC2 == 2){
419         output_high(PIN_D3);
420     }
421     else if (LC2 == 3){
422         output_low(PIN_D3);
423     }
424     data_in = 0;
425 }
426 else if(data_in == 0xc1){

```

```

427     LC3 = incoming;
428     if (LC3 == 2){
429         output_high(PIN_B3);
430     }
431     else if (LC3 == 3){
432         output_low(PIN_B3);
433     }
434     data_in = 0;
435 }
436 else if(data_in == 0x2b){ //Blinds up
437     mover_persiana(0); //Subir persiana
438 }
439 else if(data_in == 0x2c){ //Blinds down
440     mover_persiana(8); //Bajar persiana
441 }
442 else if(data_in == 0x2d){
443     output_low(PIN_E0); //Parar persiana
444     output_low(PIN_E1);
445 }
446 }
447 }
448 if(status == 0x80) //Master is requesting data
449 {
450     if(data_in == 0x14){ //Enviar T y P
451         temp_h=temp_s; //Valor entero para enviar por i2c
452         temp_l=(temp_s-temp_h)*10; //Decimales para enviar por i2c
453         i2c_write(TOTAL,temp_h); //Send temp sensor
454         i2c_write(TOTAL,temp_l);
455         int l_state = 0; //3 bits of light' states
456         l_state = input_state(PIN_D0)<<2;
457         l_state = l_state + (input_state(PIN_D3)<<1);
458         l_state = l_state + input_state(PIN_B3);
459         i2c_write(TOTAL,l_state);
460
461         int16 Pint;
462         Pint = P; //Cojer los valores enteros de P p.e: 1234 W
463         i2c_write(TOTAL,Pint&0xF);
464         i2c_write(TOTAL,Pint>>4&0xF);
465         i2c_write(TOTAL,Pint>>8&0xF);
466
467     }
468     if (data_in == 0x15) //Enviar Alarma
469         i2c_write(TOTAL,alarm);
470 }
471 }
472 }
473
474
475 void main() {
476     bit_set(TRISC,4); //C3 / SCL input
477     bit_set(TRISC,5); //C4 / SDA input
478     bit_set(TRISC,6); //V1 input
479     bit_set(TRISC,7); //V2 input
480     set_tris_d(0xFF); //Port_D as output
481
482     bit_set(TRISB,2); //MD3 input
483     bit_clear(TRISB,3); //LLum 3 output
484     bit_set(TRISB,4);
485     bit_set(TRISB,5);
486     bit_set(TRISB,6);
487     bit_set(TRISB,7);
488
489     clear_interrupt(int_ssp);
490     disable_interrupts(int_ssp);
491     enable_interrupts(int_ssp);
492     enable_interrupts(int_rb);
493     enable_interrupts(int_ext2);
494     enable_interrupts(global);
495
496     bit_clear(TRISE,0); //output for blinds up
497     bit_clear(TRISE,1); //output for blinds down
498
499     setup_adc_ports(AN0_TO_AN4);

```

```

500     setup_adc(ADC_CLOCK_INTERNAL);
501
502     //Medida de temp
503     bit_clear(TRISC,2); //outputs
504     bit_clear(TRISC,1);
505
506     rtos_run();
507 }
508
509 void update_ToC(void){
510     if (ToC == 4){
511         output_low(PIN_D4);
512         output_low(PIN_D1);
513     }
514     else if (ToC == 2){ //Up
515         mover_toldo(0);
516     }
517     else if (ToC == 3){ //Down
518         mover_toldo(10);
519     }
520     return;
521 }
522
523 void update_BlindsC(void){
524     if (BlindsC == 4){
525         output_low(PIN_D4);
526         output_low(PIN_D1);
527     }
528     else if (BlindsC == 2){ //Up
529         mover_persiana(0);
530     }
531     else if (BlindsC == 3){ //Down
532         mover_persiana(8);
533     }
534     return;
535 }
536
537 void mover_toldo(int posicion){
538     //Poner límites en caso de pasarnos de posicion
539     if (posicion > 10) posicion = 10;
540
541     level_t_d = posicion; //update level desired
542     output_low(PIN_D4);
543     output_low(PIN_D1);
544
545     if (posicion < level_t){ //Si se desea mandar al principio:
546         //Subir toldo
547         if (level_t>0){ //Top not reached
548             output_high(PIN_D4);
549         }
550     }
551     else if (posicion > level_t){ //Si se desea mandar abajo del todo:
552         //Bajar toldo
553         if (level_t<10){ //Si no esta bajado
554             output_high(PIN_D1);
555         }
556     }
557 }
558
559 void mover_persiana(int posicion){
560     if (posicion > 8) posicion = 8;
561
562     level_p_d = posicion; //update level desired
563     output_low(PIN_E1);
564     output_low(PIN_E0);
565
566     if (posicion < level_p){ //Si se desea mandar al principio:
567         //Subir persiana
568         if (level_p>0){ //Open blinds if not up
569             output_high(PIN_E0);
570         }
571     }
572     else if (posicion > level_p){ //Si se desea mandar abajo del todo:

```

```
573     //Bajar persiana
574     if (level_p<8){ //Bot not reached -> Down blinds
575         output_high(PIN_E1);
576     }
577 }
578 }
579
580
```