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

1. Código Unidad Central en C

1.1. Código principal

```
//// Autor: ORIOL RIERA ROVIRA ///////
//// SISTEMA DE GESTIÓN DOMÓTICA PARA OPTIMIZAR EL CONSUMO ENERGÉTICO DE UNA VIVIENDA /////
//// TRABAJO DE FINAL DE GRADO ///////

#include <18F4550.h>
#device HIGH_INTS = TRUE
#include <config.h>

char c;                          //RS232 DATA
int16 address = 0;
int count_day = 0;      //Flag for the start of the day's address


unsigned int state = 1, first = 1;      //Define state of functionallity
char k;                          //KBD Data
int yr, month, date, day, hr, min, sec, newt;           //CLK Time
char day_arr[7][4] = {"DOM","LUN","MAR","MIE","JUE","VIE","SAB"};


int temp_h = 24, temp_l = 0;                //Temp desired for the room 24.0
int time_change = 1; //Day by default
int season = 1; //Warm season
int cont = 0; //Contar 20 veces 0.5 segundos para el timer
unsigned int alarm = 0;

int temp_h_s, temp_l_s;         //Temp read by sensor
unsigned int16 P;

// Control manual/automatico
int LC1 = 1, ALC1 = 0xc6, LState = 0; //Luces control = 1 Automatico de serie + Address + General state (OFF)
int LC2 = 1, ALC2 = 0xc2;
int LC3 = 1, ALC3 = 0xc1;
int ToC = 1, AToC = 0xc5; //Toldo control = 1 Automatico de serie + Address
int BlindsC = 1, ABlindsC = 0xc7; //Blinds control = 1 Automatico de serie + Address
int AirC = 1, AAir = 0xc4; //Air control = 1 Automatico de serie + Address for i2c
int HeatC = 1, AHeat = 0xc3; //Heat control = 1 Automatic + Address


#int_TIMER1
void control_time(void){
   if (cont%2 == 0){ //Cada 1 segundo
      rtc_get_time(yr,month,date,day,hr,min,sec);
      newt = 1;
      if (bit_test(alarm,0) == 1) output_toggle(PIN_A0); //Ventana
      if (bit_test(alarm,2) == 1) output_toggle(PIN_A1); //Calidad del aire
   }

   if (cont%20 == 0 && cont != 0 && cont != 40){ //Cada 10 segundos
      short int status;
      do{
         i2c_start();
         i2c_write(0x40);
         status = i2c_write(0x15);
         i2c_stop();
         if (status == 1) delay_ms(50); //If no answer wait 1s
      }
      while (status == 1);

      i2c_start();
      i2c_write(0x41);
      alarm = i2c_read(0);
      i2c_stop();

      if (bit_test(alarm,0) == 0) output_low(PIN_A0);
      if (bit_test(alarm,2) == 0) output_low(PIN_A1);
   }

   if ((cont+5)%10 == 0 && cont != 0){ //Cada 5 segundos
      short int status;

      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
      modo manual
143         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
           %2u/%2u/%2u",hr,min,sec,day_arr[day-1],date,month,yr);
272       }
273       if (input(PIN_B4) == 1){ //Okay pressed, change time
274          state = 0;
275          first = 1;
276          delay_ms(300);
277          while(TRUE){
278             int j = 0;
279             int new = 0;
280             switch(state){
281                case 0:
282                   if (first == 1 && state == 0){
283                      printf(lcd_putc,"\fSet time <=\nSec:");
284                      first = 0;
285                   }
286                   while (j<2 && state == 0){
```

```
                k = kbd_getc();
                if (k!=0){
                   lcd_putc(k);
                   j++;
                   if (j == 1) new = (k-48)*10;
                   else new = new + (k-48);
                }
             }
             if (new > 59)
                new = 59;
             while (j == 2 && state == 0){
                if (input(pin_B4) == 1){ //Summit new time
                   sec = new;
                   rtc_write(state,sec);
                   ++state;
                   first = 1;
                   delay_ms(300);
                }
             }
             break;
          case 1: //Min
             if (first == 1 && state == 1){
                printf(lcd_putc,"\fSet time <=\nMin:");
                first = 0;
             }
             while (j<2 && state == 1){
                k = kbd_getc();
                if (k!=0){
                   lcd_putc(k);
                   j++;
                   if (j == 1) new = (k-48)*10;
                   else new = new + (k-48);
                }
             }
             if (new > 59)
                new = 59;
             while (j == 2 && state == 1){
                if (input(pin_B4) == 1){ //Summit new time
                   min = new;
                   rtc_write(state,min);
                   ++state;
                   first = 1;
                   delay_ms(300);
                }
             }
             break;
          case 2: //Hour
             if (first == 1 && state == 2){
                printf(lcd_putc,"\fSet time <=\nHour:");
                first = 0;
             }
             while (j<2 && state == 2){
                k = kbd_getc();
                if (k!=0){
                   lcd_putc(k);
                   j++;
                   if (j == 1) new = (k-48)*10;
                   else new = new + (k-48);
                }
             }
             if (new > 23)
                new = 23;
             while (j == 2 && state == 2){
                if (input(pin_B4) == 1){ //Summit new time
                   hr = new;
                   rtc_write(state,hr);
                   ++state; //Change next setting
                   first = 1;
                   delay_ms(300);
                }
             }
             break;
          case 3: //Day
```

```
                if (first == 1 && state == 3){
                   printf(lcd_putc,"\fSet time <=\nDay:");
                   first = 0;
                }
                while (j<1 && state == 3){
                   k = kbd_getc();
                   if (k!=0){
                      lcd_putc(k);
                      j++;
                      new = (k-48)+1; //Day goes from 1(sunday) to 7(saturday);
                   }
                }
                if (new > 7)
                   new = 7;
                while (j == 1 && state == 3){
                   if (input(pin_B4) == 1){ //Summit new time
                      day = new-1;
                      rtc_write(state,day);
                      state++; //Change next setting
                      first = 1;
                      delay_ms(300);
                   }
                }
                break;
             case 4: //Date
                if (first == 1 && state == 4){
                   printf(lcd_putc,"\fSet time <=\nDate:");
                   first = 0;
                }
                while (j<2 && state == 4){
                   k = kbd_getc();
                   if (k!=0){
                      lcd_putc(k);
                      j++;
                      if (j == 1) new = (k-48)*10;
                      else new = new + (k-48);
                   }
                }
                if (new > 31)
                   new = 31;
                while (j == 2 && state == 4){
                   if (input(pin_B4) == 1){ //Summit new time
                      date = new;
                      rtc_write(state,date);
                      ++state; //Change next setting
                      first = 1;
                      delay_ms(300);
                   }
                }
                break;
             case 5: //Month
                if (first == 1 && state == 5){
                   printf(lcd_putc,"\fSet time <=\nMonth:");
                   first = 0;
                }
                while (j<2 && state == 5){
                   k = kbd_getc();
                   if (k!=0){
                      lcd_putc(k);
                      j++;
                      if (j == 1) new = (k-48)*10;
                      else new = new + (k-48);
                   }
                }
                if (new > 12)
                   new = 12;
                while (j == 2 && state == 5){
                   if (input(pin_B4) == 1){ //Summit new time
                      month = new;
                      rtc_write(state,month);
                      ++state;
                      first = 1;
                      delay_ms(300);
```

```
                    }
                }
                break;
            case 6: //Year
                if (first == 1 && state == 6){
                    printf(lcd_putc,"\fSet time <=\nYear:");
                    first = 0;
                }
                while (j<2 && state == 6){
                    k = kbd_getc();
                    if (k!=0){
                        lcd_putc(k);
                        j++;
                        if (j == 1) new = (k-48)*10;
                        else new = new + (k-48);
                    }
                }
                while (j == 2 && state == 6){
                    if (input(pin_B4) == 1){ //Summit new time
                        yr = new;
                        rtc_write(state,yr);
                        state = 5;
                        first = 1;
                        delay_ms(300);
                        return;
                    }
                }
                break;
            case 7: state = 0; first = 1; break;
            case 10:
                state = 5;
                first = 1;
                delay_ms(300);
                return;

        }
    }
    if(state>250) state = 6; //If we try to go under 0
  }
}

void menu_temperatura(void){
  if (first == 1 && state == 1){
    first = 0;
    printf(lcd_putc,"\fT Sens %2u.%u °C\nT Def: %2u.%u
    °C",temp_h_s,temp_l_s,temp_h,temp_l);
  }
  if (input(PIN_B4) == 1){ //Okay pressed
        lcd_putc("\fDefine new temp:\n");
        int j = 0;
        int temp_n[3];    //Guardar los datos de temperatura
        while (j<2 && input(PIN_B5) == 0){
            k = kbd_getc();
            if (k!=0){
                lcd_putc(k);
                temp_n[j] = k-48;
                j++;
            }
        }
        lcd_putc(".");
        while (j == 2 && input(PIN_B5) == 0){
            k = kbd_getc();
            if (k!=0){
                lcd_putc(k);
                lcd_putc(" °C");
                temp_n[j] = k-48;
                j++;
            }
        }
        while (TRUE){

            if(input(PIN_B4) == 1){ //OKAY pressed
                first = 1;
```

```
                temp_h = temp_n[0]*10+temp_n[1];
                temp_l = temp_n[2];
                short int status;
                do{
                   i2c_start();
                   i2c_write(0x40); //Address of pic
                   i2c_write(0xbb); //Instruction to change temp
                   i2c_write(temp_h);
                   delay_ms(10);
                   status = i2c_write(temp_l);
                   i2c_stop();
                   if (status == 1) delay_ms(100); //If data not read properly try
                   again with delay
                }
                while ( status == 1);
                delay_ms(300);
                break;
             }
             else if (input(PIN_B5) == 1){  //ESC pressed
                state = 1;
                break;
             }
          }
       }
    }

    void menu_potencia(void){
       if (first == 1 && state == 2){
          printf(lcd_putc,"\fPower consumed:\n%lu W",P);
          first = 0;
       }
       if (input(PIN_B4) == 1){ //Se pulsa OKAY
          first = 1;
          while(state == 2){
             if (first == 1 && state == 2){
                int16 i;
                float E = 0.0;
                for (i=count_day;i<address;i++){
                   E += read_ext_eeprom(i);
                }
                E = E/(address-count_day);
                E = E/10.0;
                printf(lcd_putc,"\fDaily E cons.:\n %.2f KWh",E);
                first = 0;
             }
          }
          state = 2; first = 1; //Reset global screen
       }
    }


    void menu_mode(void){
       if(first == 1 && state == 3){
          lcd_putc("\fChoose scene");
          first = 0;
       }
       if (input(PIN_B4) == 1){
          first = 1;
          state = 1;
          lcd_putc(" <=");
          delay_ms(150);
          while(TRUE){
             switch(state){
                case 0: state = 3; first = 1; break;
                case 1:
                   if (first == 1 && state == 1){
                      lcd_putc("\fGood Morning");
                      first = 0;
                   }
                   if (input(PIN_B4) == 1){
                      lcd_putc(" <=");
                      send_action(0x74);
                      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);
```

```
                    change_data(ALC3,LC3);
                    delay_ms(10);

                    first = 1;
                    state = 1; //Show GM option
                    delay_ms(300);
                    return;
                 }
                 break;
              case 3:
                 if (first == 1 && state == 3){
                    lcd_putc("\fOut Mode");
                    first = 0;
                 }
                 if (input(PIN_B4) == 1){
                    first = 1;
                    lcd_putc("\fOut Mode <=");
                    disable_interrupts(int_timer1); //Desactivar que se envie el
                    cambio de estado
                    //Change mode
                    change_data(0xcc,0x20); //Modo OUT

                    //Change lights to auto
                    LC1=1;
                    LC2=1;
                    LC3=1;
                    change_data(ALC1,LC1);
                    delay_ms(10);
                    change_data(ALC2,LC2);
                    delay_ms(10);
                    change_data(ALC3,LC3);
                    delay_ms(10);


                    //Lower/higher the temp while outside depending on the season
                    if(season == 1) temp_h = temp_h+2; //In warm season temp can be a
                    bit higher during night
                    else temp_h = temp_h-2;
                    short int status;
                    do{
                       i2c_start();
                       i2c_write(0x40); //Address of pic
                       i2c_write(0xbb); //Instruction to change temp
                       i2c_write(temp_h);
                       delay_ms(10);
                       status = i2c_write(temp_l);
                       i2c_stop();
                       if (status == 1) delay_ms(20);//If data not read properly try
                       again with delay
                    }
                    while ( status == 1);
                    delay_ms(200);

                    //Ask for new pass
                    int i; //For loops
                    char data[3]; //store eeprom data
                    char clave[3]; //store keybord data
                    lcd_putc("\fSystem blocked\nPassword:");
                    while(TRUE){
                       if (first == 1){
                          first = 0;
                          lcd_putc("\fSystem blocked\nPassword:");
                          i = 0;
                          while (i<=2){
                             k = kbd_getc();
                             if (k!=0){
                                i++;
                                clave[i]=k;
                                lcd_putc("*");
                             }
                          }
                       }
                       if (input(PIN_B4) == 1){ //OKAY Pressed
```

```
                        int h;
                        for (h=0;h<=2;h++) data[h] = read_eeprom(h);
                        if ((clave[0]==data[0]) || (clave[1]==data[1]) ||
                        (clave[2]==data[2]))
                        {
                           lcd_putc("\fPassword correct\nWelcome home");
                           enable_interrupts(int_timer1);
                           change_data(0xcc,0x10); //Modo auto

                           //Go back to normal temp
                           if(season == 1) temp_h = temp_h-2;
                           else temp_h = temp_h+2;

                           short int status;
                           do{
                              i2c_start();
                              i2c_write(0x40); //Address of pic
                              i2c_write(0xbb); //Instruction to change temp
                              i2c_write(temp_h);
                              delay_ms(10);
                              status = i2c_write(temp_l);
                              i2c_stop();
                              if (status == 1) delay_ms(100); //If data not read
                              properly try again with delay
                           }
                           while ( status == 1);



                           //Ask for any alarm
                           do{
                              i2c_start();
                              i2c_write(0x40);
                              status = i2c_write(0x15);
                              i2c_stop();
                              if (status == 1) delay_ms(50); //If no answer wait 1s
                           }
                           while (status == 1);

                           i2c_start();
                           i2c_write(0x41);
                           alarm = i2c_read(0);
                           i2c_stop();


                           if (bit_test(alarm,0) == 1 || bit_test(alarm,1) == 1){
                             lcd_putc("\fAlarma in\n");
                             if (bit_test(Alarm,0) == 1) lcd_putc("Windw ");
                             if (bit_test(Alarm,1) == 1) lcd_putc("Inside");
                             while(bit_test(alarm,0) == 1 || bit_test(alarm,1) == 1){
                                if (input(PIN_B4) == 1){
                                   send_action(0xa4); //clear alarm
                                   bit_clear(alarm,0);
                                   bit_clear(alarm,1);
                                }
                             }
                           }

                           delay_ms(500);
                           break;
                        }
                        else{
                           lcd_putc("\fWrong password");
                           first = 1;
                           delay_ms(500);
                        }
                     }
                  }
                  first = 1;
                  state = 3;
                  return;
               }
               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 \ndevices?");
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
852                            case 2:  //On
853                               if (first == 1 && state == 2){
854                                  first = 0;
855                                  lcd_putc("\fToldo: <=\nUP");
856                               }
857                               if (input(PIN_B4) == 1){ //Okay = Summit choice
858                                  ToC = 2;
859                                  lcd_putc(" <=");
860                                  first = 1;
```

```
                        delay_ms(300);
                        state = 2;
                        change_data(AToC,ToC);
                        return;
                     }
                     break;

                  case 3:  //OFF
                     if (first == 1 && state == 3){
                        first = 0;
                        lcd_putc("\fToldo: <=\nDOWN");
                     }
                     if (input(PIN_B4) == 1){ //Okay = Summit choice
                        ToC = 3;
                        lcd_putc(" <=");
                        first = 1;
                        delay_ms(300);
                        state = 2;
                        change_data(AToC,ToC);
                        return;
                     }
                     break;
                  case 4:
                     if (first == 1 && state == 4){
                        first = 0;
                        lcd_putc("\fToldo: <=\nSTOP");
                     }
                     if (input(PIN_B4) == 1){ //Okay = Summit choice
                        ToC = 4;
                        lcd_putc(" <=");
                        first = 1;
                        delay_ms(300);
                        state = 2;
                        change_data(AToC,ToC);
                        return;
                     }
                     break;

                  case 10:
                     state = 2; //Volver al estado general
                     return;
                     break;

                  default: state = 1; break;
               }
            }
         }
         break;
      case 3:
         config_air();
         break;
      case 4:
         if (first == 1 && state == 4){
            first = 0;
            lcd_putc("\fHeat:");
            lcd_gotoxy(1,2);
            if (HeatC == 1) lcd_putc("A");
            else if (HeatC == 2) lcd_putc("ON");
            else lcd_putc("OFF");
         }
         if (input(PIN_B4) == 1){ //Okay pressed
            state = HeatC;
            first = 1;
            lcd_gotoxy(7,1);
            lcd_putc("<=");
            delay_ms(100);
            while(TRUE){
               switch(state){
                  case 0: state = 3; break;
                  case 1:
                     if (first == 1 && state == 1){
                        first = 0;
                        lcd_putc("\fHeat: <=\nA");
```

```
                        }
                        if (input(PIN_B4) == 1){ //Okay = Summit choice
                           HeatC = 1;
                           lcd_putc(" <=");
                           first = 1;
                           state = 4;
                           change_data(AHeat,HeatC);
                           delay_ms(300);
                           return;
                        }
                        break;
                     case 2:
                        if (first == 1 && state == 2){
                           first = 0;
                           lcd_putc("\fHeat: <=\nON");
                        }
                        if (input(PIN_B4) == 1){ //Okay = Summit choice
                           HeatC = 2;
                           lcd_putc(" <=");
                           first = 1;
                           state = 4;
                           change_data(AHeat,HeatC);
                           delay_ms(300);
                           return;
                        }
                        break;
                     case 3:
                        if (first == 1 && state == 3){
                           first = 0;
                           lcd_putc("\fHeat: <=\nOFF");
                        }
                        if (input(PIN_B4) == 1){ //Okay = Summit choice
                           HeatC = 3;
                           lcd_putc(" <=");
                           first = 1;
                           state = 3;
                           change_data(AHeat,HeatC);
                           delay_ms(300);
                           return;
                        }
                        break;
                     case 10:
                        state = 4;
                        return;
                        break;
                     default:state = 1; break;
                  }
               }
            }


            break;
         case 5:
            if (first == 1 && state == 5){
               first = 0;
               lcd_putc("\fBlinds:");
               lcd_gotoxy(1,2);
               if (BlindsC == 1) lcd_putc("A");
               else if (BlindsC == 2) lcd_putc("UP");
               else if (BlindsC == 3) lcd_putc("DOWN");
               else lcd_putc("STOP");
            }
            if (input(PIN_B4) == 1){ //Okay pressed
               state = ToC;
               first = 1;
               lcd_gotoxy(9,1);
               lcd_putc("<=");
               delay_ms(100);
               while(TRUE){
                  switch(state){
                     case 0: state = 4; break;
                     case 1:  //Automatic
                        if (first == 1 && state == 1){
```

```
                        first = 0;
                        lcd_putc("\fBlinds: <=\nA");
                     }
                     if (input(PIN_B4) == 1){ //Okay = Summit choice
                        BlindsC = 1;
                        lcd_putc(" <=");
                        first = 1;
                        delay_ms(300);
                        state = 5;
                        change_data(ABlindsC,BlindsC);
                        return;
                     }
                     break;

                  case 2:  //On
                     if (first == 1 && state == 2){
                        first = 0;
                        lcd_putc("\fBlinds: <=\nUP");
                     }
                     if (input(PIN_B4) == 1){ //Okay = Summit choice
                        BlindsC = 2;
                        lcd_putc(" <=");
                        first = 1;
                        delay_ms(300);
                        state = 5;
                        change_data(ABlindsC,BlindsC);
                        return;
                     }
                     break;

                   case 3:  //OFF
                     if (first == 1 && state == 3){
                        first = 0;
                        lcd_putc("\fBlinds: <=\nDOWN");
                     }
                     if (input(PIN_B4) == 1){ //Okay = Summit choice
                        BlindsC = 3;
                        lcd_putc(" <=");
                        first = 1;
                        delay_ms(300);
                        state = 5;
                        change_data(ABlindsC,BlindsC);
                        return;
                     }
                     break;
                   case 4:
                     if (first == 1 && state == 4){
                        first = 0;
                        lcd_putc("\fBlinds: <=\nSTOP");
                     }
                     if (input(PIN_B4) == 1){ //Okay = Summit choice
                        BlindsC = 4;
                        lcd_putc(" <=");
                        first = 1;
                        delay_ms(300);
                        state = 5;
                        change_data(ABlindsC,BlindsC);
                        return;
                     }
                     break;

                   case 10:
                     state = 5; //Volver al estado general
                     return;
                     break;

                   default: state = 1; break;
                }
             }
          }
          break;
       case 10: state = 4; first = 1; return; break; //Reset menu and global
       state variable
```

```
            default:  state = 1; first = 1; break;
         }
      }

   }

}

void config_luz(void){
   if (first == 1 && state == 1){
      first = 0;
      lcd_putc("\fLights: ");
      lcd_putc("EXT:");
      if (LC1 == 1){
         lcd_putc("A");
         if(bit_test(Lstate,2)==1) lcd_putc("O ");
         else lcd_putc("X ");
      }
      else if (LC1 == 2) lcd_putc("MO ");
      else lcd_putc("MX ");

      lcd_gotoxy(1,2);
      lcd_putc("CR:");
      if (LC2 == 1){
         lcd_putc("A");
         if(bit_test(Lstate,1)==1) lcd_putc("O ");
         else lcd_putc("X ");
      }
      else if (LC2 == 2) lcd_putc("MO ");
      else lcd_putc("MX ");
      lcd_putc("TL:");
      if (LC3 == 1){
         lcd_putc("A");
         if(bit_test(Lstate,0)==1) lcd_putc("O ");
         else lcd_putc("X ");
      }
      else if (LC3 == 2) lcd_putc("MO ");
      else lcd_putc("MX ");
   }

   if (input(PIN_B4) == 1){ //Okay pressed
      first = 1;
      state = 1;
      delay_ms(150);
      while(TRUE){
         switch(state){
            case 0: state = 3; first = 1; break;
            case 1:
               exterior();
               break;
            case 2:

               if (first == 1 && state == 2){
                  first = 0;
                  lcd_putc("\fCorridor:");
                  lcd_gotoxy(1,2);
                  if (LC2 == 1) lcd_putc("A");
                  else if (LC2 == 2) lcd_putc("ON");
                  else lcd_putc("OFF");
               }
               if (input(PIN_B4) == 1){ //Okay pressed
                  state = LC2;
                  first = 1;
                  lcd_gotoxy(11,1);
                  lcd_putc("<=");
                  delay_ms(100);
                  while(TRUE){
                     switch(state){
                        case 0: state = 3; break;
                        case 1:  //Automatic
                           if (first == 1 && state == 1){
                              first = 0;
                              lcd_putc("\fCorridor: <=\nA");
```

```
                        }
                        if (input(PIN_B4) == 1){ //Okay = Summit choice
                           LC2 = 1;
                           bit_set(Lstate,1);
                           lcd_putc(" <=");
                           first = 1;
                           state = 1;
                           change_data(ALC2,LC2);
                           delay_ms(300);
                           return;
                        }
                        break;

                     case 2:  //On
                        if (first == 1 && state == 2){
                           first = 0;
                           lcd_putc("\fCorridor: <=\nM ON");
                        }
                        if (input(PIN_B4) == 1){ //Okay = Summit choice
                           LC2 = 2;
                           bit_clear(Lstate,1);
                           lcd_putc(" <=");
                           first = 1;
                           state = 1;
                           change_data(ALC2,LC2);
                           delay_ms(300);
                           return;
                        }
                        break;

                      case 3:  //OFF
                        if (first == 1 && state == 3){
                           first = 0;
                           lcd_putc("\fCorridor: <=\nM OFF");
                        }
                        if (input(PIN_B4) == 1){ //Okay = Summit choice
                           LC2 = 3;
                           lcd_putc(" <=");
                           first = 1;
                           state = 1;
                           change_data(ALC2,LC2);
                           delay_ms(300);
                           return;
                        }
                        break;

                     case 10: //Esc pressed
                        state = 1;
                        return;
                        break;

                     default: state = 1; break;
                  }
               }
            }


         break;
      case 3:

         if (first == 1 && state == 3){
            first = 0;
            lcd_putc("\fToilet:");
            lcd_gotoxy(1,2);
            if (LC3 == 1) lcd_putc("A");
            else if (LC3 == 2) lcd_putc("ON");
            else lcd_putc("OFF");
         }
         if (input(PIN_B4) == 1){ //Okay pressed
            state = LC3;
            first = 1;
            lcd_gotoxy(9,1);
```

```
                     lcd_putc("<=");
                     delay_ms(100);
                     while(TRUE){
                        switch(state){
                           case 0: state = 3; break;
                           case 1:  //Automatic
                              if (first == 1 && state == 1){
                                 first = 0;
                                 lcd_putc("\fToilet: <=\nA");
                              }
                              if (input(PIN_B4) == 1){ //Okay = Summit choice
                                 LC3 = 1;
                                 lcd_putc(" <=");
                                 first = 1;
                                 state = 1;
                                 change_data(ALC3,LC3);
                                 delay_ms(300);
                                 return;
                              }
                              break;

                           case 2:  //On
                              if (first == 1 && state == 2){
                                 first = 0;
                                 lcd_putc("\fToilet: <=\nM ON");
                              }
                              if (input(PIN_B4) == 1){ //Okay = Summit choice
                                 LC3 = 2;
                                 bit_set(Lstate,0);
                                 lcd_putc(" <=");
                                 first = 1;
                                 state = 1;
                                 change_data(ALC3,LC3);
                                 delay_ms(300);
                                 return;
                              }
                              break;

                           case 3:  //OFF
                              if (first == 1 && state == 3){
                                 first = 0;
                                 lcd_putc("\fToilet: <=\nM OFF");
                              }
                              if (input(PIN_B4) == 1){ //Okay = Summit choice
                                 LC3 = 3;
                                 bit_clear(Lstate,0);
                                 lcd_putc(" <=");
                                 first = 1;
                                 state = 1;
                                 change_data(ALC3,LC3);
                                 delay_ms(300);
                                 return;
                              }
                              break;

                           case 10: //Esc pressed
                              state = 1;
                              return;
                              break;

                           default: state = 1; break;
                        }
                     }
                  }


                  break;
               case 10: state = 1; first = 1; return; break;
               default: state = 1; first = 1; break;
            }
         }
      }
   }
```

```
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;
```

```
                break;

             default: state = 1; break;
          }
       }
    }

}


void config_toldo(void){

}

void config_air(void){
   if (first == 1 && state == 3){
      first = 0;
      lcd_putc("\fAir:");
      lcd_gotoxy(1,2);
      if (AirC == 1) lcd_putc("A");
      else if (AirC == 2) lcd_putc("ON");
      else lcd_putc("OFF");
   }
   if (input(PIN_B4) == 1){ //Okay pressed
      state = AirC;
      first = 1;
      lcd_gotoxy(6,1);
      lcd_putc("<=");
      delay_ms(100);
      while(TRUE){
         switch(state){
            case 0: state = 3; break;
            case 1:  //Automatic
               if (first == 1 && state == 1){
                  first = 0;
                  lcd_putc("\fAir: <=\nA");
               }
               if (input(PIN_B4) == 1){ //Okay = Summit choice
                  AirC = 1;
                  lcd_putc(" <=");
                  first = 1;
                  state = 3;
                  change_data(AAir,AirC);
                  delay_ms(300);
                  return;
               }
               break;

            case 2:  //On
               if (first == 1 && state == 2){
                  first = 0;
                  lcd_putc("\fAir: <=\nON");
               }
               if (input(PIN_B4) == 1){ //Okay = Summit choice
                  AirC = 2;
                  lcd_putc(" <=");
                  first = 1;
                  state = 3;
                  change_data(AAir,AirC);
                  delay_ms(300);
                  return;
               }
               break;

            case 3:  //OFF
               if (first == 1 && state == 3){
                  first = 0;
                  lcd_putc("\fAir: <=\nOFF");
               }
               if (input(PIN_B4) == 1){ //Okay = Summit choice
                  AirC = 3;
                  lcd_putc(" <=");
                  first = 1;
```

```
                    state = 3;
                    change_data(AAir,AirC);
                    delay_ms(300);
                    return;
                }
                break;

            case 10: //Esc pressed
                state = 3;
                return;
                break;

            default: state = 1; break;
        }
    }
}

}


void change_data(int address, int number){
    BYTE status1, status2;
    do{
        i2c_start();
        i2c_write(0x40);
        status1 = i2c_write(address);
        delay_ms(10);
        status2 = i2c_write(number);
        i2c_stop();
        if (status1 == 1 || status2 == 1) delay_ms(40);
    }
    while(status1 == 1 || status2 == 1);
}

void send_action(int number){
    BYTE status;
    do{
        i2c_start();
        i2c_write(0x40);
        status = i2c_write(number);
        i2c_stop();
        if (status == 1) delay_ms(40);
    }
    while(status == 1);
}


void update_time(void)
{
    if (hr < 12){ //De las 00 a 12 de la mañana mas barato
        if (time_change == 1){ //Before it was day
            send_action(0x67); // cheap mode
            time_change = 3;
        }
    }
    else if (time_change == 3) { //outside of night span and before it was night
        send_action(0x68); //out day
        time_change = 1;
    } //Day mode

    if ( month >= 9 && month <=4 && season == 1){ //Cold season starts in october
        season = 2;
        send_action(0x66);
    }
    else if (month >= 5 && month <= 8 && season == 2){ //Warm season starts in may
        season = 1;
        send_action(0x65);
    }
}
```

1.2. Configuraciones

```
#include <stdio.h>
#fuses XT,NOWDT,PUT,NOPROTECT,NOLVP
#use delay(clock = 4000000)

//LCD init.
#define LCD_ENABLE_PIN  PIN_E0
#define LCD_RS_PIN      PIN_E1
#define LCD_RW_PIN      PIN_E2
#define LCD_DATA4       PIN_A2
#define LCD_DATA5       PIN_A3
#define LCD_DATA6       PIN_A4
#define LCD_DATA7       PIN_A5
#include <lcd.c>

//KBD init.
#include <kbd.c>

//Outputs
#BYTE TRISA = 0x85
#BYTE PORTA = 0x05
#BYTE TRISB = 0x86
#BYTE PORTB = 0x06

//rs232
#BYTE TRISC = 0x87
#use rs232(baud=9600, bits=8, parity = N, xmit=PIN_C6, rcv=PIN_C7, stream = PC)

//I2C
#use i2c(master, sda=PIN_C4, scl=PIN_C5)
#include <EEPROM_24512.c>
#include <DS1307.c>
#include <Float_EE.c>

//Global Variables
#rom 0xF00000 = {'3','5','7'} //Contrasenya porta

///// Global functions ////////////////
void init_routine(void);

//Menu principal
void menu_temperatura(void);
void menu_potencia(void);
void menu_mode(void);
void menu_config(void);
void menu_clock(void);

//Menu mode
void modo_out (void);
void good_morning(void);
void good_night(void);

//Menu config
//void config_luz(void);
void config_toldo(void);
void config_air(void);
//void control_heat(void);

//Menu luz
void exterior(void);
//void corridor(void);
void toilet(void);

//Funciones externas de control
void update_time(void);
void change_data(int address, int number);
void send_action(int number);
```

1.3. Código Eeprom 24512

1.3.1. Leer/Escribir datos

```
void write_ext_eeprom(long int address, BYTE data)
{
   short int status;
   i2c_start();            //Inicializa la Transmisión
   i2c_write(0xa0);        //Escribe la palabra de control(direccion
                           //0h + 0 para escritura)
   i2c_write(address>>8); //Parte alta de la dirección a escribir en la eeprom
   i2c_write(address);    //Parte baja de la dirección a escribir en la eeprom
   i2c_write(data);      //Dato a escribir
   i2c_stop();           //Finalización de la transmisión
   i2c_start();          //Reinicio
   status=i2c_write(0xa0); //Lectura del bit ACK, para evitar escrituras
                           //incorrectas
   while(status==1)      //Si es "1" se espera para responder al esclavo
   {
      i2c_start();
      status=i2c_write(0xa0);
   }
}


BYTE read_ext_eeprom(long int address) {
   BYTE data;
   i2c_start();            //Inicializa la Transmisión
   i2c_write(0xa0);        //Escribe la palabra de control(direccion
                           //0h + 0 para escritura)
   i2c_write(address>>8); //Parte alta de la dirección a escribir en la eeprom
   i2c_write(address);    //Parte baja de la dirección a escribir en la eeprom
   i2c_start();          //Reinicio
   i2c_write(0xa1);        //Escribe la palabra de control(direccion
                           //0h + 1 para escritura)
   data=i2c_read(0);     //Lectura del Dato
   i2c_stop();           //Finalización de la transmisión
   return(data);
}

```

1.3.2. Leer/Escribir floats

```
void WRITE_FLOAT_EXT_EEPROM(long int n, float data) {
   int i;

   for (i = 0; i < 4; i++)
     write_ext_eeprom(i + n, *((int8 *)(&data) + i));
}

float READ_FLOAT_EXT_EEPROM(long int n) {
   int i;
   float data;

   for (i = 0; i < 4; i++)
     *((int8 *)(&data) + i) = read_ext_eeprom(i + n);

   return(data);
}
```

1.4. Código Reloj DS1307

```
int BCDaBIN(BYTE bcd)
{
int varia;
varia=bcd;
varia>>=1;
varia &= 0x78;
return (varia + (varia>>2) + (bcd & 0x0f));
}

BYTE DECaBCD(int DEC)
{
return(((DEC/10) << 4) + (DEC % 10));
}

void rtc_get_time(BYTE& yr, BYTE& month, BYTE& date, BYTE& day, BYTE& hr, BYTE& min,
BYTE& sec) {
i2c_start(); //Escritura
i2c_write(0xD0); //Codigo de escritura
i2c_write(0x00); //Puntero de la primera dirección
i2c_start(); //Lectura
i2c_write(0xD1); //Codigo de lectura
sec = BCDaBIN(i2c_read()&0x7f); //Lectura de los 7 bit de los segundos
min = BCDaBIN(i2c_read()&0x7f); //Lectura de los 7 bit de los minutos
hr = BCDaBIN(i2c_read()&0x3f); //Lectura de los 6 bit de las horas
day = BCDaBIN(i2c_read()&0x07); //Lectura de los 3 bits de dia de la semana
date = BCDaBIN(i2c_read()&0x3f); //Lectura de dia del mes
month = BCDaBIN(i2c_read()&0x1f); //Lectura de los 5 bits de los meses
yr = BCDaBIN(i2c_read(0)); //Lectura de los años
i2c_stop(); //Finaliza comunicación
}

void rtc_write(unsigned int8 address, int data){
  i2c_start();                          // Start I2C
  i2c_write(0xD0);                      // DS1307 address
  i2c_write(address);                   // Send register address
  i2c_write(data);                     // Write data to the selected register
  i2c_stop();
}
```

2. Código Módulo Energía en C

```
//// Autor: ORIOL RIERA ROVIRA ///////
//// SISTEMA DE GESTIÓN DOMÓTICA PARA OPTIMIZAR EL CONSUMO ENERGÉTICO DE UNA VIVIENDA /////
//// TRABAJO DE FINAL DE GRADO ///////

#include <18F4550.h>
#device adc=10
#device HIGH_INTS=TRUE
#fuses XT, NOWDT
#use delay (clock=4000000)

#include <math.h>

#BYTE TRISA = 0x85
#BYTE TRISB = 0X86
#BYTE TRISC = 0X87
#BYTE TRISD = 0X88
#BYTE TRISE = 0X8A
#BYTE PORTA = 0X05
#BYTE PORTB = 0X06
#BYTE PORTC = 0X07
#BYTE PORTD = 0X08

#define ENCODER_P PIN_B6
#define L1 PIN_D0
#define L2 PIN_D3
#define L3 PIN_B2
#define ENCODER_T PIN_B5
#define T_UP PIN_D4
#define T_DOWN PIN_D1
#define P_UP PIN_E0
#define P_DOWN PIN_E1

//I2C Slave mode to comm with central
#use i2c(slave, SDA=PIN B0, SCL=PIN B1, ADDRESS=0x40, STREAM = TOTAL, FORCE HW)

//I2C Master mode to comm with sensor
#use I2C(master, SDA = PIN C5, SCL = PIN C4, STREAM = LOCAL)
#include <TEMP DS1621 AUX.c>

// Global Variables
int state = 0x10, data in, alarm = 0;
int level t = 0, level p = 0; //level toldo/persiana
int level_t_d = 0, level_p_d = 0; //level desired

int light;

float P;

int LC1 = 1; //Luz 1 automatica
int LC2 = 1; //Luz 2 automatica
int LC3 = 1; //Luz 3 automatica
int ToC = 1; //Toldo automatico
int BlindsC = 1; //Persiana automatica
int AirC = 1; //Aire en modo automatico
int HeatC = 1; //Heat in auto mode

//RTOS mode
#use rtos(timer = 1, minor_cycle = 100ms)


/////// CONTROL DE TEMPERATURA //////////////////
#task(rate = 400ms, max = 100ms)
void control_temp(void);

//Variables temperatura
int temp_h,temp_l;
float temp_s = 0; //T Sensada
float temp_d = 24.0; //T Definida

void mover persiana(int posicion);

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°
81             output_high(PIN_C2);
82          }
83          else
84             output_low(PIN_C2);
85       }
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°
89             output_high(PIN_C1);
90          }
91          else
92             output_low(PIN_C1);
93       }
94
95       rtos_yield();
96    }
97
98    /////// MEDIR LUZ EXTERIOR //////////////
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   /////// CONTROL DEL TOLDO EXTERIOR MEDIANTE SENSOR DE LUZ, VIENTO Y TEMPERATURA
      ////////////////
107   #task(rate = 400ms, max = 50ms)
108   void control_toldo(void);
109   void mover_toldo(int posicion);
110
111   void control_toldo(void){
112      //Lee viento
113      float wind;
114      set_adc_channel(2);
115      wind = read_adc();
116      wind = wind*5.0/1023.0;  //Escalar a 0-5V
117      wind = wind*6.0;        //Mide de 0 a 30km/h
118
119      if (wind < 10 && ToC == 1){    //No wind && Auto mode
120
121         if (light >175 || temp_s > 24) {  //Si hace sol o temperatura superior a 24°:
            baja toldo
122            mover_toldo(10); // Bajar toldo (al final)
123         }
124         else if (light < 100) // Si no hay luz sube toldo
125         {
126            mover_toldo(0); // Subir toldo (al principio)
127         }
128      }
129      else if (wind >= 10){ //aire fuerte = subir toldo
130         mover_toldo(0); // Subir toldo (al principio)
131      }
132
133
134      if ((temp_s >27 || temp_s < 15) && BlindsC == 1){ //Bajar persiana a la mitad en
         temperaturas extremas
135         mover_persiana(4);
136      }
137
138      rtos_yield();
```

```
}


////// CONTROL DE VENTANAS  ////////////
#task(rate = 600ms, max = 20ms)
void control_ventanas(void);
int before_w1 = 1,before_w2 = 1;

void control_ventanas(void){
   if (input(PIN_C6) == 0){ //Control first window (0 = open)
      bit_set(Alarm,0); //Define que hay una ventana abierta
      if (state == 0x20 && before_w1 == 1){ //Modo out i s'obre finestra (no cuenta
      que estubiera abierta): alarma
         output_high(PIN_D2);
      }
      else{ //normal mode -> close air and heat
         before_w1 = 0;
      }
   }

   if (input(PIN_C7) == 0){
      bit_set(Alarm,0); //Define que hay una ventana abierta
      if (state == 0x20 && before_w2 == 1){ //Modo out i s'obre finestra (no cuenta
      que estubiera abierta): alarma
         output_high(PIN_D2);
      }
      else{ //normal mode -> close air and heat
         before_w2 = 0;
      }
   }
   else before_w2 = 1;

   if (input(PIN_C7) == 1 && input(PIN_C6) == 1 && state == 0x10) bit_clear(Alarm,0);

   rtos_yield();
}


////// CONTROL CALIDAD DEL AIRE //////////////
#task (rate = 5s, max = 20ms)
void control_air(void){
   float air_q;
   set_adc_channel(3);
   air_q = read_adc();
   air_q = air_q*5.0/1023.0;  //Escalar a 0-5V
   if(air_q>3)
      bit_set(Alarm,2);
   else
      bit_clear(alarm,2);
}

////// CONTROL SENSOR DE PRESENCIA //////
#task(rate = 1s, max = 10ms)
void control_presencia(void);
int cnt1 = 0, cnt2 = 0, cnt3 = 0;
void control_presencia(void){ //Si se deja de detectar i modo auto, apaga la luz
1(se enciende con interrupcion)
   if (input(PIN_B4) == 0 && LC1 == 1 && input_state(PIN_D0) == 1){
      cnt1++;
      if (cnt1 >= 10){
         output_low(PIN_D0);
         cnt1 = 0;
      }
   }
   if (input(PIN_B7) == 0 && LC2 == 1 && input_state(PIN_D3) == 1){
      cnt2++;
      if (cnt2 >= 10){
         output_low(PIN_D3);
         cnt2 = 0;
      }
   }
   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 SENSIBLITY = 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)/SENSIBLITY;
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
            (in any mode)
278               output_high(PIN_D0);
279            }
280      }
```

```
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;
```

```
            if (state == 0x10){ //Normal mode && Welcome home
               rtos_enable(control_toldo);
               mover_persiana(0); //Subir persiana
            }
            else if(state == 0x20){ //OUT MODE
               mover_toldo(0); //Subir toldo
               mover_persiana(8); //Bajar persiana
               rtos_disable(control_toldo);
               output_low(PIN_D0);  //Cerrar luces
               output_low(PIN_D3);
               output_low(PIN_B3);
            }
         }
         else if (data_in == 0xBB){ //Define new temp desired
            if (status == 2) temp_d = incoming*1.0;
            else{
               temp_d = temp_d + incoming/10.0 ;
               data_in = 0;
            }
         }
         else if(data_in == 0x68){ //Hour for high price
            output_low(PIN_D6);  //Apagar lavaplatos
            output_low(PIN_D7); //Apagar lavadora
            data_in = 0;
         }
         else if (data_in == 0x67){ //Low price hour
            output_high(PIN_D6); // Activar lavaplatos
            output_high(PIN_D7); //Activar lavadora
            data_in = 0;
         }
         else if(data_in == 0xc6){
            LC1 = incoming;
            if (LC1 == 2){
               output_high(PIN_D0);
            }
            else if (LC1 == 3){
               output_low(PIN_D0);
            }
            data_in = 0;
         }
         else if(data_in == 0xc5){
            ToC = incoming;
            update_ToC();
            data_in = 0;
         }
         else if(data_in == 0xc7){
            BlindsC = incoming;
            update_BlindsC();
            data_in = 0;
         }
         else if(data_in == 0xc4){
            AirC = incoming;
            data_in = 0;
            if (AirC == 3) output_low(PIN_C1); //OFF manual
            else if (AirC == 2) output_high(PIN_C1); //ON manual
         }
         else if(data_in == 0xc3){
            HeatC = incoming;
            data_in = 0;
            if (HeatC == 3) output_low(PIN_C2); //OFF manual
            else if (HeatC == 2) output_high(PIN_C2); //ON manual
         }
         else if(data_in == 0xc2){
            LC2 = incoming;
            if (LC2 == 2){
               output_high(PIN_D3);
            }
            else if (LC2 == 3){
               output_low(PIN_D3);
            }
            data_in = 0;
         }
         else if(data_in == 0xc1){
```

```
            LC3 = incoming;
            if (LC3 == 2){
               output_high(PIN_B3);
            }
            else if (LC3 == 3){
               output_low(PIN_B3);
            }
            data_in = 0;
         }
         else if(data_in == 0x2b){ //Blinds up
            mover_persiana(0); //Subir persiana
         }
         else if(data_in == 0x2c){ //Blinds down
            mover_persiana(8); //Bajar persiana
         }
         else if(data_in == 0x2d){
            output_low(PIN_E0); //Parar persiana
            output_low(PIN_E1);
         }

      }
      if(status == 0x80) //Master is requesting data
      {
         if(data_in == 0x14){ //Enviar T y P
            temp_h=temp_s; //Valor entero para enviar por i2c
            temp_l=(temp_s-temp_h)*10; //Decimales para enviar por i2c
            i2c_write(TOTAL,temp_h);  //Send temp sensor
            i2c_write(TOTAL,temp_l);
            int l_state = 0; //3 bits of light' states
            l_state = input_state(PIN_D0)<<2;
            l_state = l_state + (input_state(PIN_D3)<<1);
            l_state = l_state + input_state(PIN_B3);
            i2c_write(TOTAL,l_state);

            int16 Pint;
            Pint = P; //Cojer los valores enteros de P p.e: 1234 W
            i2c_write(TOTAL,Pint&0xF);
            i2c_write(TOTAL,Pint>>4&0xF);
            i2c_write(TOTAL,Pint>>8&0xF);

         }
         if (data_in == 0x15) //Enviar Alarma
            i2c_write(TOTAL,alarm);
      }

   }


   void main() {
      bit_set(TRISC,4); //C3 / SCL input
      bit_set(TRISC,5); //C4 / SDA input
      bit_set(TRISC,6); //V1 input
      bit_set(TRISC,7); //V2 input
      set_tris_d(0xFF); //Port_D as output

      bit_set(TRISB,2); //MD3 input
      bit_clear(TRISB,3);//LLum 3 output
      bit_set(TRISB,4);
      bit_set(TRISB,5);
      bit_set(TRISB,6);
      bit_set(TRISB,7);

      clear_interrupt(int_ssp);
      disable_interrupts(int_ssp);
      enable_interrupts(int_ssp);
      enable_interrupts(int_rb);
      enable_interrupts(int_ext2);
      enable_interrupts(global);

      bit_clear(TRISE,0); //output for blinds up
      bit_clear(TRISE,1); //output for blinds down

      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
```