Trabajo de Fin de Máster

Máster Universitario en Ingeniería Industrial

Sistema de acceso electrónico para residencias

ANEXO

Autora: Cristina Triginer Coll
Directora: Rosa Rodríguez Montañés

Convocatoria: Julio 2017

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Sumari

1. Programa Principal

```
// GENERAL
#include <pic16f690.h>      // Incluye la librería del microcontrolador y sus variables
#include <xc.h>             //Incluye la librería del compilador xc8
#define _XTAL_FREQ 4000000  //Definición de la frecuencia de funcionamiento del microcontrolador, 4MHz

// Real Time Clock I2C
#define SDA_PIN RB4         //Linea de datos para el i2c
#define SCL_PIN RB6         // Linea de clock para el i2c
#define SDA_DIR TRISB4      //Dirección de los datos del i2c
#define SCL_DIR TRISB6      //Dirección del clock de i2c
#include "i2c.h"            //Libreria del manejo del i2c
#include "DS1307.h"         //Libreria del manejo del Real Time Clock
#include "UART.h"           //Librería de configuración y manejo de la EUSART

// EEPROM
#include "EEPROM.h"         //Librería del manejo de la EEPROM

// BEGIN CONFIG
#pragma config FOSC = INTRCIO   //Selección del oscilador interno
#pragma config WDTE = OFF       //Watchdog Timer desactivado
#pragma config PWRTE = OFF      //Poer-up timer desactivado
#pragma config MCLRE = OFF      //El MCLRE es una entrada digital
#pragma config CP = OFF         //Code Protection esta desactivado
#pragma config CPD = OFF        //El Data Code Protection esta desactivado
#pragma config BOREN = OFF      //El Brown out reset bit desta desactivado
#pragma config IESO = OFF       //Internal External Switchover desactivado
#pragma config FCMEN = OFF      // Fail Safe clock monitor esta desactivado
//END CONFIG


#define fila1 RC0               //Fila 1 del teclado conectado al puerto RC0
#define fila2 RC1               //Fila 2 del teclado conectado al puerto RC1
#define fila3 RC2               //Fila 3 del teclado conectado al puerto RC2
#define fila4 RC3               //Fila 4 del teclado conectado al puerto RC3
#define columna1 RA0            //Columna 1 del teclado conectado al puerto RA0
#define columna2 RA1            //Columna 2 del teclado conectado al puerto RA1
#define columna3 RA2            //Columna 3 del teclado conectado al puerto RA2
#define led_beep RC6            //Interfase del usuario con el teclado
#define led_codigo RC7          //Sensor de corriente
#define led_vermell RC4         //Motor abrir puerta
#define motor_menos RC5         //Motor cerrar puerta

char LECTURA_segons;            //variable para los segundos
char LECTURA_minuts;            //variable para los minutos
char LECTURA_hores;             //variable para las horas
char LECTURA_dia_setm;          //variable para los dias de la semana
char LECTURA_any;               //variable para el año
unsigned char DATO[12];         //vector datos recibidos via bluetooth
char i,j, index;                //variables de trabajo
char LECTURA_EEPROM;
unsigned char direccion;        //variable que recorre las filas de la eeprom
unsigned char kk, ii;           // variables de trabajo
unsigned char datoRecibido;     //Flag de dato recibido del bluetooth
```

```
unsigned char teclaraw;          //tecla leida en la exploración del teclado
unsigned char tecla;             //codigo de la tecla pulsada
unsigned char teclap;            //tecla pulsada en ASCII
unsigned char numerotecla;       //variable que cuenta el numero de teclas pulsadas
unsigned char CODIGO[4];         //vector de la clave pulsada
unsigned char teclapulsada;      //codigo de la tecla pulsada
unsigned char itaula;            //indice de la tabla de la eeprom
unsigned char verifacio;         //contador de verificación del codigo en la eeprom
unsigned char c, temp;           //variables de trabajo
unsigned char obrirporta;        //Flag para abrir la puerta
unsigned char index_beep;        //


void teclat(void);              //función de exploración del teclado
void borrarcodigo(void);        //función de reset de las variables
char comprovarhorari (void);    //función de comprobar horario
void comprovarcodi (void);      //función de comprobar codigo

void main (void)                        //PRograma principal
{
    TRISA = 0x07;                       //Configuración puerto A
    TRISC = 0x00;                       //PUERTOC as Output
    ANSEL = 0X00;                       //Configuración en modo digital
    ANSELH = 0X00;                      //Configuración en modo digital
    TRISBbits.TRISB5 = 1;               //Configuración del puerto B5 como entrada°
//  TRISCbits.TRISC3 = 0;
//  TRISCbits.TRISC5 = 0;
//  TRISAbits.TRISA5 = 0;
//  TRISAbits.TRISA4 = 0;
    UART_Init(9600);                    //Configuración de la velocidad de transmisión serie

    OPTION_REGbits.nRABPU = 0;          //Habilitación de los pull-ups
    WPUA = 0x07;                        //Configuración pull ups puerto A
    IOCA = 0x07;                        //Habilitación de las interrupciones por cambio en puerto
    RABIF=0;                            //Reset del interrupt flag
    RABIE=1;                            //Habilitación de interrupciones por cambio
    INTCONbits.PEIE = 1;                //Habilitación de interrupciones por perifericos
    PIE1bits.RCIE = 1;                  //Configuracion de interrupción por recepción en la UART
    INTCONbits.GIE = 0;                 //Habilitar interrupciones generales off

    DS1307_init();                      //Arranque del RTC
    __delay_ms(50);                     //espera para tener los datos disponibles

//***CODIGO PARA LA PUESTA EN HORA DEL RTC***
// DS1307_write(Reg_segons, 0x00);      //Introducir segundos
// DS1307_write(Reg_hores, 0x19);       //Introducir hora
// DS1307_write(Reg_minuts, 0x34);      //Introducir minutos
// DS1307_write(Reg_any,0x17);          //Introducir año



////***BORRAR EEPROM***
//while (ii<0xff){
//  escribir_eeprom(ii,0xFF);
//  ii++;
//  __delay_ms(10);
//}
```

```
//*** CODIGO PARA EL DEBUGGING DE LA LECTURA DEL RTC
//LECTURA_segons = DS1307_read(Reg_segons); // llegim les 3 mesures
//escribir_eeprom(0xF6,LECTURA_segons);
//  __delay_ms(50);
//LECTURA_minuts = DS1307_read(Reg_minuts);
//escribir_eeprom(0xF3,LECTURA_minuts);
//  __delay_ms(50);
//LECTURA_hores = DS1307_read(Reg_hores);
//escribir_eeprom(0xF0,LECTURA_hores);
//  __delay_ms(50);
//LECTURA_any=DS1307_read(Reg_any);
//escribir_eeprom(0xF8,LECTURA_any);
//  __delay_ms(50);


GIE=1;                          //Habilitación Global de las interrupciones
PORTC = 0;                      //Inicialización Puerto C a cero
led_beep=0;                     //Inicializacion de variables
led_codigo=0;                   //Inicializacion de variables
led_vermell=0;                  //Inicializacion de variables
fila1=0;fila2=0;fila3=0;fila4=0;//Inicializacion de variables
numerotecla=0;                  //Inicializacion de variables
ii=0;                           //Inicializacion de variables
i=0;                            //Inicializacion de variables
datoRecibido=0;                 //Inicializacion de variables


//***BUCLE PRINCIAPAL***
while (1)                     //mientras el contenido del parentesis sea 1 ejecutar los siguientes procedimientos
    {


while (datoRecibido)            //funcion que guarda la informacion enviada via bluetooth a la eeprom
{
kk=0;
direccion=0x00;
while(!kk)
{
    if (leer_eeprom(direccion) == 0xFF)             //si la fila de la eeprom esta vacia escribimos un 1
        {escribir_eeprom(direccion,1);
        __delay_ms(20);
            for (index=0;index<12;index++){         //escribimos los 12 bytes recibidos via bluetooth
            direccion++;                            //en la fila de la epprom "direccion"
            escribir_eeprom(direccion, DATO[index]);
            __delay_ms(20);
            kk++;}
        }
    else                                            //si la fila esta llena pasamos a la siguiente fila
    {direccion=direccion + 0x10;}
}
datoRecibido=0;                                     //ponemos a cero dato recibido
}
}
}  //fin del programa principal
```

```
//***FUNCIÓN TECLADO***
void teclat(void)
{
    tecla=0;
    fila1=0;
    fila2=1;
    fila3=1;
    fila4=1;
    if ( (PORTA & 0x07) != 7)           //Si las columnas son distintas de 1
        {                               //significa tecla pulsada
         teclaraw= ((PORTA&0x07) ^ 0x07); tecla = (teclaraw | 0x10); //Identificación
        }

    fila1=1;
    fila2=0;
    fila3=1;
    fila4=1;
    if ( (PORTA & 0x07) != 7)           //Si las columnas son distintas de 1
        {                               ///significa tecla pulsada
         teclaraw= ((PORTA&0x07) ^ 0x07); tecla = (teclaraw | 0x20); //Identificación
        }

    fila1=1;
    fila2=1;
    fila3=0;
    fila4=1;
    if ( (PORTA & 0x07) != 7)           //Si las columnas son distintas de 1
        {                               //signidica tecla pulsada
         teclaraw= ((PORTA&0x07) ^ 0x07); tecla = (teclaraw | 0x40); //Identificación
        }


    fila1=1;
    fila2=1;
    fila3=1;
    fila4=0;
    if ( (PORTA & 0x07) != 7)           //Si las columnas son distintas de 1
        {                               //significa tecla pulsada
         teclaraw= ((PORTA&0x07) ^ 0x07); tecla = (teclaraw | 0x80); //Identificación
        }

            switch (tecla){             //Asignación del codigo ASCII a la tecla identificada
            case 0x11: teclap=0x31; break;
            case 0x12: teclap=0x32; break;
            case 0x14: teclap=0x33; break;
            case 0x21: teclap=0x34; break;
            case 0x22: teclap=0x35; break;
            case 0x24: teclap=0x36; break;
            case 0x41: teclap=0x37; break;
            case 0x42: teclap=0x38; break;
            case 0x44: teclap=0x39; break;
            case 0x81: teclap=0x2A; break;
            case 0x82: teclap=0x30; break;
            case 0x84: teclap=0x23; break;
            case 0x00: teclap=0x00; break;
            }
```

```
        fila1=0;fila2=0;fila3=0;fila4=0; //Inicialización de las filas

        index_beep=100;                  //Realimentación al usuario de tecla pulsada
        while(index_beep>0)
        {
            led_beep = 1;
            __delay_ms(1);
            led_beep = 0;
            __delay_ms(1);
            index_beep--;
        }


}


//***FUNCIÓN COMPROBAR CODIGO***
void comprovarcodi (void) // función que comprueba si el codigo introducido en el teclado existe en la eeprom
{
itaula=0x00;
while ((itaula<=0xA0) & !obrirporta)            //Mientras no se encuentre codigo valido
{
if (leer_eeprom(itaula)==0x01)                  //Busqueda de la fila con codigo
{
    for (c=0;c<=3;c++)                          //comparar 4 digitos
    {
        if (CODIGO[c]== leer_eeprom(itaula+c+5))
            {
            verifacio++;
            }
        else
            {
            verifacio=0;
            break;
            }
    }

    if (verifacio==4)                           //CODIGO existente en EEPROM
        {
        if ( leer_eeprom(itaula+9) )            //hay horario asociado a CODIGO
            {
            comprovarhorari();                  //llamar a comprovarhorario
            }
        else                                    //codigo valido sin horario. Se abre puerta
            {
            obrirporta = 1;                     //Flag abrir puerta habilitado
            led_vermell=1;
            __delay_ms(1000);
            led_vermell=0;
            __delay_ms(1000);
            led_vermell=1;
            __delay_ms(1000);
            led_vermell=0;
            __delay_ms(1000);
            }
        itaula = 0xA0;
        }
    itaula= itaula+0x10;
}
else
    {
    itaula= itaula+0x10;
    }
}
borrarcodigo();                                 //llamar a borrarcodigo
}
```

```
///*** FUNCION COMPROBAR HORARIO ***
char comprovarhorari (void)     // función para comprobar si el codigo introducido esta en horario valido
{
temp=1;                                 //variable de trabajo
temp=temp << (DS1307_read(Reg_dia_setm)-1); //desplazo el 1 tantas posiciones como el dia de la semana del 1 al 7

if ((leer_eeprom(itaula+0x0A) && temp)!=0) // mientras el dia de la semana sea valido
    {
        if ((10*((DS1307_read(Reg_hores)&0x30)>>4)+(DS1307_read(Reg_hores)&0x0f))<= leer_eeprom(itaula+0x0C))
        {                                               //mientras la hora sea menor que la h de salida
            if((10*((DS1307_read(Reg_hores)&0x30)>>4)+(DS1307_read(Reg_hores)&0x0f))>= leer_eeprom(itaula+0x0B))
                {                                       //mientras la hora sea mayor que la h de entrada
                obrirporta= 1 ;             //bandera para abrir puerta habilitada
                led_vermell=1;
                __delay_ms(200);
                led_vermell=0;
                __delay_ms(200);
                led_vermell=1;
                __delay_ms(200);
                led_vermell=0;
                __delay_ms(200);
                }
            else                        //no es horario bueno, no se abre puerta
                {
                borrarcodigo();
                }
        }
        else                            //no es horario bueno, no se abre puerta
        {
        borrarcodigo();
        }
    }
else                                    //no es horario bueno, no se abre puerta
    {
        borrarcodigo();
    }
return (obrirporta);                    //devuelve el flag abrir puerta habilitado o deshabilitado
}


void borrarcodigo(void)
{
    itaula=0x00;                        //variable que recorre las posiciones de la memoria eeprom a cero
    verifacio=0;                        //contador de verificacion de codigo a cero
    CODIGO[0]=0;                        //primera tecla del vector CODIGO a cero
    CODIGO[1]=0;                        //segunda tecla del vector CODIGO a cero
    CODIGO[2]=0;                        //tercera tecla del vector CODIGO a cero
    CODIGO[3]=0;                        //cuarta tecla del vector CODIGO a cero
    obrirporta=0;                       //bandera para abrir puerta deshabilitada

//UART_Write(CODIGO[0]);
//UART_Write(CODIGO[1]);
//UART_Write(CODIGO[2]);
//UART_Write(CODIGO[3]);

}



//***SERVICIO DE INTERRUPCIONES***
//Permite obtener los 12 datos enviados por bluetooth y también obtener un codigo valido introducido en el teclado

void interrupt Interrupcion()                    //Funcion que atiende las interrupciones
{
    if(PIR1bits.RCIF)                           //siempre que la bandera del recepetor de la UART este activada
    {
        DATO[i]=UART_Read();                    //leeremos el primer dato recibido y lo colocaremos en la posción 0 del vector
        i++;
        if (i>11){i=0; datoRecibido=1;}         //Repetiremos esta acción mientras no se hayan recibido 12 datos.
        PIR1bits.RCIF=0;                        //Cuando tengamos 12 datos, activaremos la bandera DatoRecibido y pondremos a cero
                                                //la bandera del receptor de la UART.
    }
```

```
if (RABIF==1)                                   //siempre que la bandera por cambio de estado en el puerto A este activa
    {
    if ( (PORTA&0x07) != 0x07)
        {teclat();                              //lamaremos a la función teclado
         teclapulsada=teclap;                   //asignamos la tecla pulsada a la variable teclapulsada
           if (teclapulsada!=0)                 //siempre que haya una tecla pulsada
               {
               if (teclapulsada!=0x23)          //siempre que la tecla pulsada sea distina de #
                   {CODIGO[numerotecla]=teclapulsada; //colocamos el valor de la tecla pulsada en la primera posicion del vector CODIGO
                   numerotecla++;               //contador del numero de teclas pulsadas
                   }
               if (teclapulsada==0x2A)          //siempre que la tecla pulsada sea *
                   {
                   numerotecla=0;               //contador de tecla a cero
                   CODIGO[0]=0;                 //primera tecla del vector CODIGO a cero
                   CODIGO[1]=0;                 //segunda tecla del vector CODIGO a cero
                   CODIGO[2]=0;                 //tercera tecla del vector CODIGO a cero
                   CODIGO[3]=0;                 //cuarta tecla del vector CODIGO a cero
                   }
               }
        if ((numerotecla==4)& (teclapulsada==0x23)) //siempre que hayamos pulsado 5 teclas y la ultima tecla pulsada sea #
            {
            comprovarcodi();                        //llamaremos a la función comprobarcodigo
            led_codigo=1;
            __delay_ms(1000);
            led_codigo=0;
            numerotecla=0;                          //contador de tecla a cero
            }
        if ((teclapulsada==0x23) & (numerotecla!=4))//siempre que hayamos pulsado # pero el numero de teclas no sea 5
            {                                       //no se considerara un codigo
            numerotecla=0;                          //contador de tecla a cero
            CODIGO[0]=0;                            //primera tecla del vector CODIGO a cero
            CODIGO[1]=0;                            //segunda tecla del vector CODIGO a cero
            CODIGO[2]=0;                            //tercera tecla del vector CODIGO a cero
            CODIGO[3]=0;                            //cuarta tecla del vector CODIGO a cero
            }
        }


    RABIF=0;                                        //deshabilitamos la bandera por cambio de estado en el puerto A
    }

}
```

2. Librerías utilizadas

2.1. UART.h

```
char UART_Init(const long int baudrate)
{
  unsigned int x;
  x = (_XTAL_FREQ - baudrate*64)/(baudrate*64);   //SPBRG for Low Baud Rate
  if(x>255)                                       //If High Baud Rage Required
  {
    x = (_XTAL_FREQ - baudrate*16)/(baudrate*16); //SPBRG for High Baud Rate
    BRGH = 1;                                     //Setting High Baud Rate
  }
  if(x<256)
  {
    BRGH=1;
    SPBRG = 25;                                    //Writing SPBRG Register
    SYNC = 0;                                     //Setting Asynchronous Mode, ie UART
    SPEN = 1;                                     //Enables Serial Port
    CREN = 1;                                     //Enables Continuous Reception
    TXEN = 1;                                     //Enables Transmission
    return 1;                                     //Returns 1 to indicate Successful Completion
  }
  return 0;                                       //Returns 0 to indicate UART initialization failed
}

char UART_Read()
{
  while(!RCIF);
  return RCREG;
}

void UART_Write(char data)
{
  while(!TRMT);
  TXREG = data;
}


char UART_TX_Empty()
{
  return TRMT;
}


char UART_Data_Ready()
{
  return RCIF;
}

void UART_Write_Text(char *text)
{
  int i;
  for(i=0;text[i]!='\0';i++)
    UART_Write(text[i]);
}
```

```
void UART_Read_Text(char *Output, unsigned int length)
{
  unsigned int i;
  for(int i=0;i<length;i++)
  Output[i] = UART_Read();
}
```

2.2. DS1307.h

```
#define DS1307_READ_ADDR       0xD1
#define DS1307_WRITE_ADDR      0xD0

#define Reg_segons          0x00
#define Reg_minuts          0x01
#define Reg_hores           0x02
#define Reg_dia_setm        0x03
#define Reg_dia             0x04
#define Reg_mes             0x05
#define Reg_any             0x06
#define Config_Reg          0x07

void DS1307_init();
char DS1307_read( char reg);
void DS1307_write( char reg_address,  char value);


void DS1307_init()
{
   DS1307_write(Config_Reg, 0x00);// Write 0x10 to Control register to enable SQW-Out
}



 char DS1307_read( char reg)
{
   char val = 0;
   i2c_start();
   i2c_out_byte(DS1307_WRITE_ADDR);
   i2c_ack();
   i2c_out_byte(reg);
   i2c_ack();
   i2c_start();
   i2c_out_byte(DS1307_READ_ADDR);
   i2c_ack();
   val = i2c_in_byte();
   i2c_nack();
   i2c_stop();
   return(val);
}

void DS1307_write( char reg_address,  char value)
{
   i2c_start();
   i2c_out_byte(DS1307_WRITE_ADDR);
   i2c_ack();
   i2c_out_byte(reg_address);
   i2c_ack();
   i2c_out_byte(value);
   i2c_ack();
   i2c_stop();
}
```

2.3. I2c.h

```
// common i2c routines
char i2c_in_byte(void);
void i2c_out_byte(char o_byte);
void i2c_nack(void);
void i2c_ack(void);
void i2c_start(void);
void i2c_stop(void);
void i2c_high_sda(void);
void i2c_low_sda(void);
void i2c_high_scl(void);
void i2c_low_scl(void);


// Common I2C Routines
char i2c_in_byte(void)
{
   char i_byte, n;
   i2c_high_sda();
   for (n=0; n<8; n++)
   {
      i2c_high_scl();

      if (SDA_PIN)
      {
         i_byte = (i_byte << 1) | 0x01; // msbit first
      }
      else
      {
         i_byte = i_byte << 1;
      }
      i2c_low_scl();
   }
   return(i_byte);
}


 void i2c_out_byte(char o_byte)
 {
    char n;
    for(n=0; n<8; n++)
    {
       if(o_byte&0x80)
       {
          i2c_high_sda();
       }
       else
       {
          i2c_low_sda();
       }
       i2c_high_scl();
       i2c_low_scl();
       o_byte = o_byte << 1;
    }
    i2c_high_sda();
 }
```

```
void i2c_nack(void)
{
   i2c_high_sda();  // data at one
   i2c_high_scl();  // clock pulse
   i2c_low_scl();
}

void i2c_ack(void)
{
 //  i2c_low_sda(); // bring data low and clock
  // i2c_high_sda();
  // i2c_low_scl();
   i2c_high_scl();
   i2c_low_scl();
}


void i2c_start(void)
{
 //  i2c_low_scl();
   i2c_high_sda();
   i2c_high_scl();  // bring SDA low while SCL is high
   i2c_low_sda();
   i2c_low_scl();
}

void i2c_stop(void)
{
   i2c_low_scl();
   i2c_low_sda();
   i2c_high_scl();
   i2c_high_sda();  // bring SDA high while SCL is high
   // idle is SDA high and SCL high
}

void i2c_high_sda(void)
{
   // bring SDA to high impedance
   SDA_DIR = 1;
//    __delay_us(4);
}

void i2c_low_sda(void)
{
   SDA_PIN = 0;
   SDA_DIR = 0;  // output a hard logic zero
//    __delay_us(4);
}

void i2c_high_scl(void)
{
   SCL_DIR = 1;   // high impedance
 //  __delay_us(4);
}
```

```
void i2c_low_scl(void)
{
   SCL_PIN = 0;
   SCL_DIR = 0;
 //  __delay_us(4);
}
```

2.4. EEPROM.h

```
////////////////////////////////////////////////////////////////
//función para la escritura de datos en la EEPROM PIC
//es necesario darle dos parámetro, el primero será la dirección
//de la EEPROM que se quiera leer

void escribir_eeprom(unsigned char direccion,unsigned char dato){
    while(EECON1bits.WR==1);  //mientras se esté realizando alguan escritura
                              //anterior espera
    EEADR=direccion;          //se carga la dirección donde se quiere escribir
    EEDATA=dato;              //se carga el dato que se quiere escribir
    EECON1bits.EEPGD=0;       //para acceder a la memoria EEPROM PIC
    EECON1bits.WREN=1;        //habilita la escritura en la EEPROM PIC
    INTCONbits.GIE=0;         //deshabilita las interrupciones
    EECON2=0x55;              //cargar en EECON2 0x55
    EECON2=0xaa;              //cargar en EECON2 0xaa
    EECON1bits.WR=1;          //inicia la escritura en la EEPROM PIC
    INTCONbits.GIE=1;         //habilita las interrupciones
    EECON1bits.WREN=0;        //deshabilita la escritura en la EEPROM PIC
    while(EECON1bits.WR==1);  //mientras se esté realizando la escritura espera
}

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//función para la lectura de datos desde la EEPROM PIC
//es necesario darle un parámetro, este será la dirección
//de la EEPROM que se quiera leer

unsigned char leer_eeprom(unsigned char direccion){
    EEADR=direccion;      //direccion de la eeprom que se quiere leer
    EECON1bits.EEPGD=0;   //para acceder a la memoria EEPROM PIC
    EECON1bits.RD=1;      //habilita la lectura de la EEPROM PIC
    return EEDATA;        //en el registro EEDATA se encuentra el dato
                          //leido, la función retorna el dato leido
}
```