

# Automatización de Vivienda IoT con PSoC y ESP8266

Resumen: Los sistemas encargados del control remoto y monitoreo del hogar cada vez son más comunes, van desde dispositivos alarmas hasta servicios de gestión energética de electrodomésticos. Es por esto que se presenta el diseño de un sistema para el control de iluminación, programación de encendido de electrométricos y monitoreo del hogar vía internet con Blink.

Palabras clave: Domótica, Blink, Modulo Wifi, Programación de tareas, RTC, Notificaciones.

## 1. INTRODUCCIÓN

En los sistemas de automatización de vivienda es vital contar con conexión a internet, para que permita un control remoto. El control y monitoreo de dispositivos actualmente es conocido como internet de las cosas, sin embargo el reto es grande, puesto que se requiere que los dispositivos cuenten con módulos de comunicación capaces de enviar datos a través de internet, lo cual representa una alta complejidad. Adicionalmente es necesario la conexión con un servidor, que permitan un constante flujo de datos, el debido manejo de los mismos y la correcta visualización en una aplicación web.

Por estas razones surge Blink como una herramienta que permita la conexión a internet y a aplicaciones web, con gran facilidad y con gran soporte para los dispositivos con tarjeta de comunicación Wi-Fi del mercado. Es por esto que se realizara una aplicación con Blink que se conecte a un dispositivo muy común en el mercado como lo es el ESP8266, el cual tiene soporte para el envío y recepción de datos con la aplicación para Smartphone Blink. A su vez el ESP8266 se comunicara con el PSoC 5LP, para que este controle los periféricos necesarios que permitan el monitoreo de temperatura, la detección de personas en el hogar y el control de electrodomésticos e iluminación.

## 2. FORMULACIÓN DEL PROBLEMA

En este problema entonces se deberá ddiseñar e implementar un sistema de domótica que permita encender o apagar por lo menos cuatro dispositivos “electrónicos” (los cuales serán simulados con LED’s), controlar el sistema de iluminación en una casa (Los cuales serán representados dos bombillos DC de 12 V) y recibir información de por lo menos tres sensores (movimiento, temperatura y magnético de puerta o ventana). Adicionalmente se tendrán sensores encargados de detectar movimiento en la casa o un incendio. Todos los elementos del sistema podrán ser controlados remota o localmente. Para el control remoto se hará uso de un módulo Wi-Fi ESP8266 y una aplicación desarrollada en Blynk para smartphone.

El encendido o apagado tendrá además la opción de activarlos por tiempo, es decir, haciendo uso de un reloj de tiempo real (RTC) conectado al PSoC se podrá determinar a qué hora enciende y/o apaga un dispositivo en particular. Para el sistema de iluminación se tendrá la opción de encender o apagar el bombillo o configurar el nivel de iluminación en cada uno de ellos. Los sensores enviaran por lo menos dos tipos de alarma al usuario cuando se disparen, un mensaje al Smartphone.

## 3. DISEÑO Y MODELO DE SOLUCIÓN

### *1)* Aplicación

La aplicación usada para la comunicación móvil fue Blynk la cual tiene una interface

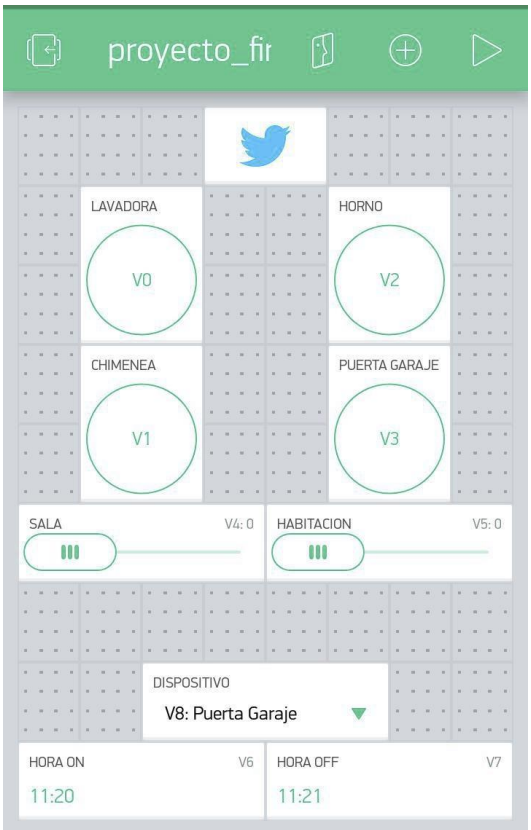



Figura 1. Interface de la aplicación de Blink.

### *2) Comunicación entre psoc y Blink*

La comunicación entre la aplicación y el psoc se realizo por el modulo y se programo por medio de Arduino.

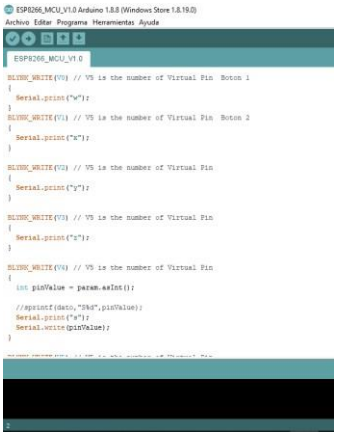



Figura.2. Comunicación entre Blink y Psoc

La comunicación entre la aplicación y el psoc se realizo por medio de etiquetas, el código de Arduino:

```
void loop()
{
  Blynk.run();
  if(Serial.available()>0)
  {
    char in_serial=Serial.read();
    //Serial.print(in_serial);
    if(banderat){
      sprintf(tweets,"Alerta N%d de Temperatura",in_serial);
      Blynk.tweet(tweets);
      banderat=false;

    }
     if(banderam){
      sprintf(tweets,"Alerta N%d de Movimiento",in_serial);
      Blynk.tweet(tweets);
      banderam=false;
     }
     if (banderaw)
     {
      if(in_serial=='1')
      {
        Blynk.virtualWrite(V0,HIGH);
      }else{
        Blynk.virtualWrite(V0,LOW);
        }
       //BLYNK_CONNECTED();
      banderaw=false;
     }
     if(banderax)
     {
      if(in_serial=='1')
      {
        Blynk.virtualWrite(V1,HIGH);
      }else{
        Blynk.virtualWrite(V1,LOW);
        }
       //BLYNK_CONNECTED();
      banderax=false;
     }
     if(banderay)
     {
      if(in_serial=='1')
      {
        Blynk.virtualWrite(V2,HIGH);
      }else{
        Blynk.virtualWrite(V2,LOW);
        }
       //BLYNK_CONNECTED();
      banderay=false;
     }
     if(banderaz)
     {
     if(in_serial=='1')
      {
        Blynk.virtualWrite(V3,HIGH);
      }else{
        Blynk.virtualWrite(V3,LOW);
        }
     if(bandraq)
     {
      Blynk.virtualWrite(V4,in_serial);
       bandraq=false;
     }
////////////////banderas de estados;

    if (in_serial=='t')
    {
      banderat=true;
    }else if(in_serial=='m'){
      banderam=true;

    }else if(in_serial=='W'){
      banderaw=true;
    }
     else if(in_serial=='X'){
      banderax=true;
     }
     else if(in_serial=='Y'){
      banderay=true;
        }
     else if(in_serial=='Z'){
       banderaz=true;
        }else if(in_serial=='Q'){
       bandraq=true;
        }
  }
```

## 3) *Modulo ESP8266*

El módulo de comunicaciones para el protocolo wifi fue el módulo MCU.V.1.0 ESP8266 con el fin de lograr la comunicación entre la aplicación de Blink comunicándolo de forma serial

Figura3. Modulo ESP8266 MCU

Los comandos se realizaron por medio de los llamados de las funciones que da la librería de Blink para su correcto funcionamiento y utilizando caracteres de control para comunicarse.

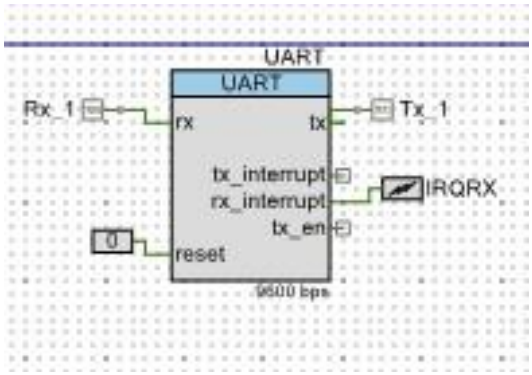



Figura4. Bloque de comunicación UART.

## 4) *Módulo RTC*

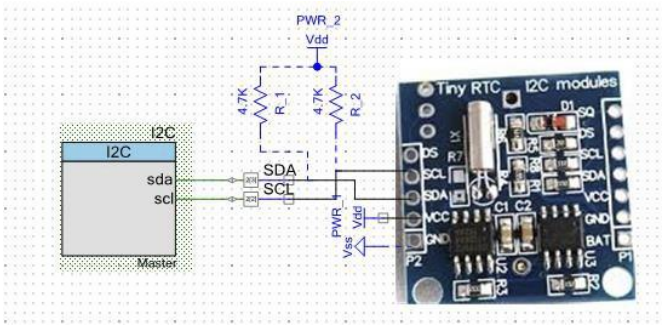



Figura 5- Conexión RTC

**Figure 4. Data Write—Slave Receiver Mode**

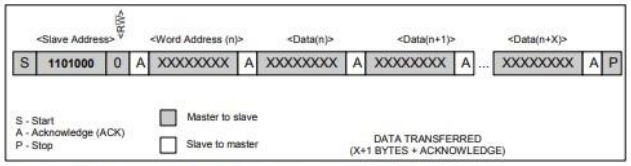



**Figure 5. Data Read—Slave Transmitter Mode**

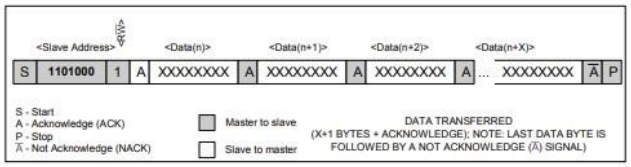



Figura 6 - Formato de lectura y escritura de datos RTC

El código para la configuración y visualización del RTC

```
void DS_begintx (void){
        do{
            //Espera mienstras el esclavo le responde
        }while(I2C_MasterSendStart(DS1307_dir, I2C_WRITE_XFER_MODE)!=I2C_MSTR_NO_ERROR);
}

void DS_init(void){
    DS_begintx();
    I2C_MasterWriteByte(direccion_de_registro_control);
    I2C_MasterWriteByte(registro_control);
    I2C_MasterSendStop();
}

void DS_set_data(){
    uint8 i;

    for(i=0;i<=7;i++){
        DS_begintx();
        I2C_MasterWriteByte(i);     // Escribe la posicion
        I2C_MasterWriteByte(ds.datos[i]);           // Escribe el dato correspondiente
        I2C_MasterSendStop();
    }
}

void DS_get_data(){
    uint8 i;
    for(i=0;i<=7;i++){
        DS_begintx();
        I2C_MasterWriteByte(i);//Pone direccion de memoria que quiere leer
        I2C_MasterSendRestart(DS1307_dir, I2C_READ_XFER_MODE); // Re transmite para obtener datos
        ds.datos[i]=I2C_MasterReadByte(I2C_NAK_DATA);
        I2C_MasterSendStop();
    }
}
```

## 5) *Circuito de pwm*

El circuito implementado en el desarrollo del proyecto para el funcionamiento de los bombillos de 12V:

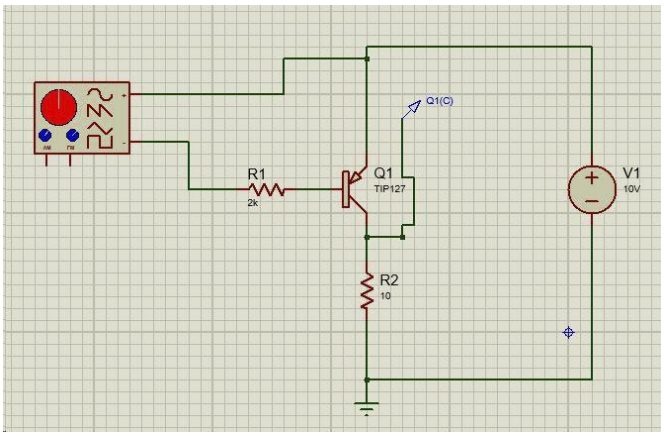



Figura 7. circuito de conmutación (dimer) de bombillos.

## 6) *Pwm*

La configuración de bloque del pwm para el funcionamiento del dimer de los bombillos.

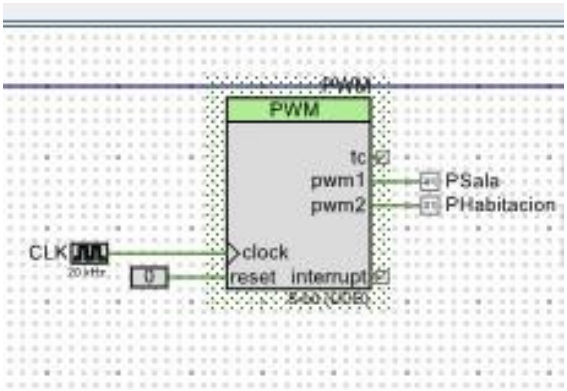



Figura 8. Bloque de PWM

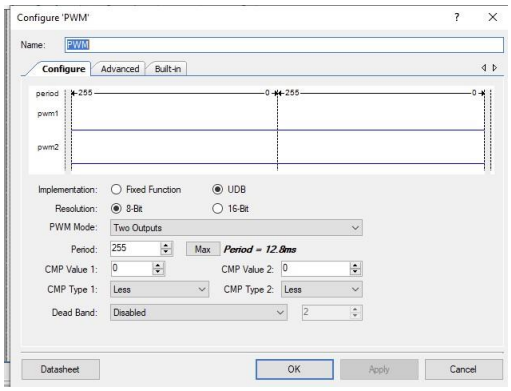



Figura 9. Configuración PWM

## 7) *Adc*

Para la obtención del muestreo de la temperatura se utilizo el adc

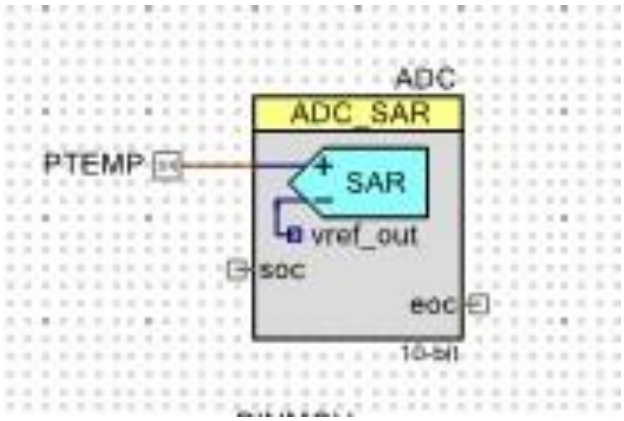



Figura10. Bloque del ADC

El código usado para el muestreo

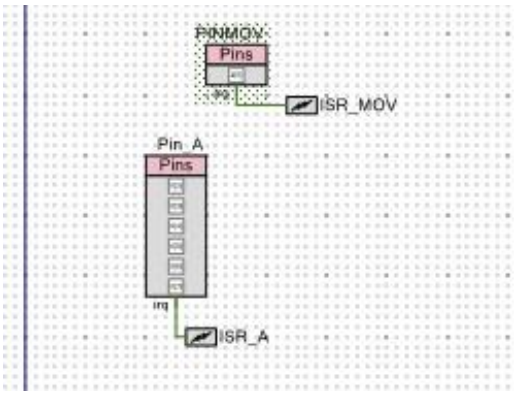
```
void muestreo(){
    //Temperatura
    uint16 temp=0;
    aux[0]=aux[1];//Actualiza dato pasado
    ADC_StartConvert();
    ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
    temp=ADC_GetResult16();
    aux[1]=ADC_CountsTo_mVolts(temp);
    if((2500>aux[0])&&(aux[1]>=2500)){//Nueva medida es mayor a 2500 y la anterior era menor 2500
        char t[1];
        temp2=temp2+1;//Aumenta el numero de alertas
        LCD_Position(0,0);
        LCD_PrintString("Warning T #");
        LCD_PrintNumber(temp2);
        UART_PutString("t");
        t[0]=(0xFF&temp2);
        UART_PutString(t);

    }
    temp=EEPROM_ReadByte(0);//Actualiza aca ya que en la interupcion no se puede
    if(temp2!=temp){
        if(temp2>temp){
            EEPROM_WriteByte(temp2,0);
        }else{
            temp2=temp;
        }
```



8) Interrupciones

Las interrupciones usadas fueron para los pulsadores y para el sensor de movimiento




El código de interrupciones del pinmov

```
}

CY_ISR(InterrupISR1){
    char m[1];
        temp2=temp2+1;//Aumenta el numero de alertas
        LCD_Position(0,0);
        LCD_PrintString("Warning M #");
        LCD_PrintNumber(temp2);
        UART_PutString("m");
        m[0]=(0xFF&temp2);
        UART_PutString(m);
        PINMOV_ClearInterrupt();
}

```

El código para las interrupciones de los interruptores

```
CY_ISR(InterrupA){
switch(Pin_A_Read()) {
        CyDelay(300);
        case 0b00111110:{
            PINA_Write(~PINA_Read());
            UART_PutChar('W');
            if (PINA_Read()== 1)
            {
                UART_PutChar('1');
            }else{
                UART_PutChar('0');
            }
             break;
        }
        case 0b00111101:{
            PINC_Write(~PINC_Read());
            UART_PutChar('Y');
            if (PINC_Read()== 1)
            {
                UART_PutChar('1');
            }else{
                UART_PutChar('0');
            }
            break;
        }
        case 0b00111011:{
            PIND_Write(~PIND_Read());
            UART_PutChar('Z');
            if (PIND_Read()== 1)
            {
                UART_PutChar('1');
            }else{
                UART_PutChar('0');
            }
            break;
        }
        case 0b00110111:{
            PINB_Write(~PINB_Read());
            UART_PutChar('X');
            if (PINB_Read()== 1)
            {
                UART_PutChar('1');
            }else{
                UART_PutChar('0');
            }

            break;
        }
        case 0b00101111:{

            if(dato<=90)
            {
                dato=dato+10;
            }

            UART_PutChar('Q');
            char dimmer=((255*dato)/100);
            UART_PutChar(dato);
            PWM_WriteCompare2(dimmer);
            break;
        }
        case 0b00011111:{

            if(dato>=10)
            {
              dato=dato-10;
            }

            UART_PutChar('Q');
            char dimmer=((255*dato)/100);
            UART_PutChar(dato);
            PWM_WriteCompare2(dimmer);
            break;
        }default:{
            break;
        }
```



# 5. CONCLUSIONES

- Con el uso del protocolo de comunicación UART se pueden generar diversas aplicaciones que contemplen transferencia de datos entre múltiples elementos electrónicos.
- Con la ayuda de la aplicación de la playstore se logra la creación rápida de un aplicativo.
- Se aprendió el uso de las funciones de la librería de Arduino para el uso del Blink.

# ANEXOS

## Código de Arduino:

```
/*************************************************
*************
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define BLYNK_MAX_SENDBYTES 256 //alargar el maxiomo del mensaje
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
// char auth[] = "bc5aa1f8e8954a4992fbce283d802d15";//nicolas
  char auth[] = "cbde13b152d3432da53ad5585f9eb550";//jefer

// Your WiFi credentials.
// Set password to "" for open networks.
//char ssid[] = "JAPEREZ";
//char pass[] = "26071967";
char ssid[] = "jefer";
char pass[] = "holiwi1234";
//char ssid[]="JEFERSSON";
//char pass[]="sebastian94102901147";

char bandera[2];
int alert=0;
char tweets[256];
char dato[12];

bool bandraq=false;
bool banderaw=false;
bool banderax=false;
bool banderay=false;
bool banderaz=false;
bool banderat=false;
bool banderam=false;

//BLYNK_CONNECTED(){
  //Blynk.syncVirtual(V0);
  //Blynk.syncVirtual(V1);
  //Blynk.syncVirtual(V2);
  //Blynk.syncVirtual(V3);
//}

BLYNK_WRITE(V0) // V5 is the number of Virtual Pin Boton 1
{
  Serial.print("w");
}
BLYNK_WRITE(V1) // V5 is the number of Virtual Pin Boton 2
{
  Serial.print("x");
}

BLYNK_WRITE(V2) // V5 is the number of Virtual Pin
{
  Serial.print("y");
}

BLYNK_WRITE(V3) // V5 is the number of Virtual Pin
{
  Serial.print("z");
}

BLYNK_WRITE(V4) // V5 is the number of Virtual Pin
{
  int pinValue = param.asInt();

  //sprintf(dato,"S%d",pinValue);
  Serial.print("s");
  Serial.write(pinValue);
}

BLYNK_WRITE(V5) // V5 is the number of Virtual Pin
{
  int pinValue = param.asInt();
  Serial.print("h");
  Serial.write(pinValue);
}

BLYNK_WRITE(V6) // V5 is the number of Virtual Pin  Sala
{
 int hora, minuto;
  int pinValue = param.asInt();
  hora=char(pinValue/3600);
  minuto=char((pinValue%3600)/60);
  Serial.print("i");
  Serial.write(hora);//Envia parte alta
   Serial.write(minuto);//Envia parte baja
}

BLYNK_WRITE(V7) // V5 is the number of Virtual Pin
{
  int hora, minuto;
  int pinValue = param.asInt();
  hora=char(pinValue/3600);
  minuto=char((pinValue%3600)/60);
  Serial.print("f");
  Serial.write(hora);//Envia parte alta
```



```
  Serial.write(minuto);//Envia parte baja
}

BLYNK_WRITE(V8) // V5 is the number of Virtual Pin
{
 int pinValue = param.asInt();
 Serial.print("m");
 Serial.write(pinValue);
}

void BlYNK_CONECTED_PSOC(int Puerto,int Hora,int
Minuto)
{
 int tiempo=(Hora*3600)+(Minuto*60);
 Blynk.virtualWrite(Puerto,tiempo);
}

void setup()
{
  // Debug console
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  Serial.print("*");
  delay(50);
  //BLYNK_CONNECTED();
}

void loop()
{
  Blynk.run();
  if(Serial.available()>0)
  {
    char in_serial=Serial.read();
    //Serial.print(in_serial);
    if(banderat){
      sprintf(tweets,"Alerta N%d de Temperatura",in_serial);
      Blynk.tweet(tweets);
      banderat=false;

    }
     if(banderam){
      sprintf(tweets,"Alerta N%d de Movimiento",in_serial);
      Blynk.tweet(tweets);
      banderam=false;
     }
     if (banderaw)
     {
      if(in_serial=='1')
      {
        Blynk.virtualWrite(V0,HIGH);
      }else{
        Blynk.virtualWrite(V0,LOW);
        }
       //BLYNK_CONNECTED();
       banderaw=false;
     }
     if(banderax)
     {
      if(in_serial=='1')
      {
        Blynk.virtualWrite(V1,HIGH);
       }else{
        Blynk.virtualWrite(V1,LOW);
        }
       //BLYNK_CONNECTED();
       banderax=false;
     }
     if(banderay)
     {
      if(in_serial=='1')
      {
        Blynk.virtualWrite(V2,HIGH);
       }else{
        Blynk.virtualWrite(V2,LOW);
        }
       //BLYNK_CONNECTED();
       banderay=false;
     }
     if(banderaz)
     {
     if(in_serial=='1')
      {
        Blynk.virtualWrite(V3,HIGH);
       }else{
        Blynk.virtualWrite(V3,LOW);
        }
       //BLYNK_CONNECTED();
       banderaz=false;
     }
     if(bandraq)
     {
      Blynk.virtualWrite(V4,in_serial);
       bandraq=false;
     }
////////////////banderas de estados;

    if (in_serial=='t')
    {
      banderat=true;
    }else if(in_serial=='m'){
      banderam=true;

    }else if(in_serial=='W'){
      banderaw=true;
    }
     else if(in_serial=='X'){
      banderax=true;
     }
     else if(in_serial=='Y'){
      banderay=true;
       }
     else if(in_serial=='Z'){
       banderaz=true;
        }else if(in_serial=='Q'){
       bandraq=true;
        }
  }

}
```



## Codigo de psoc:

```
/*
 *
========================================
 *
 *


========================================
*/
#include "project.h"
#include <stdio.h>
#include <stdbool.h>

#define DS1307_dir 0x68
#define ds1307_dir_memory 0x00
#define direccion_de_registro_control
0x07
#define registro_control 0b10010001

volatile char dato=0;
volatile bool bandera = false;
volatile bool banderaM = false;
volatile bool banderaI = false;
volatile bool banderaF = false;
volatile bool banderaS = false;
volatile bool banderaH = false;
volatile bool cont=false;
volatile bool item1[4]={0,0,0,0};
volatile bool item2[4]={0,0,0,0};
volatile bool item3[4]={0,0,0,0};
volatile bool item4[4]={0,0,0,0};
volatile uint16 temp2=0;
volatile char item=0;


typedef union
{struct{
  char sec;
  char  min;
  char hour;
  char weekDay;
  char date;
  char month;
  char  year;
};
char datos[8];
}rtc_t;

rtc_t ds;

volatile char wHoraInicio[3];
volatile char xHoraInicio[3];
volatile char yHoraInicio[3];
volatile char zHoraInicio[3];

volatile char wHoraFin[3];
volatile char xHoraFin[3];
volatile char yHoraFin[3];
volatile char zHoraFin[3];

uint16 aux[2]={0,0};
```

```
void DS_begintx (void){
        do{
                //Espera mienstras el
esclavo le responde

}while(I2C_MasterSendStart(DS1307_dir,
I2C_WRITE_XFER_MODE)!=I2C_MSTR_NO_ERROR);
}

void DS_init(void){
    DS_begintx();

I2C_MasterWriteByte(direccion_de_registro
_control);

I2C_MasterWriteByte(registro_control);
    I2C_MasterSendStop();
}

void DS_set_data(){
    uint8 i;

    for(i=0;i<=7;i++){
        DS_begintx();
        I2C_MasterWriteByte(i);     //
Escribe la posicion
        I2C_MasterWriteByte(ds.datos[i]);
// Escribe el dato correspondiente
        I2C_MasterSendStop();
    }
}

void DS_get_data(){
        uint8 i;
    for(i=0;i<=7;i++){
        DS_begintx();
        I2C_MasterWriteByte(i);//Pone
direccion de memoria que quiere leer
        I2C_MasterSendRestart(DS1307_dir,
I2C_READ_XFER_MODE); // Re transmite para
obtener datos

ds.datos[i]=I2C_MasterReadByte(I2C_NAK_DA
TA);
        I2C_MasterSendStop();
    }

}

CY_ISR(InterrupRx){
    char dato;
    dato=UART_GetChar();//recibe el dato
del bluetooth
    if(bandera==true){
        switch (dato){
            case 'w':{
                PINA_Write(~PINA_Read());
                dato=0;
                break;
            }
```



```
            case 'x':{
                PINB_Write(~PINB_Read());
                dato=0;
                break;
            }
            case 'y':{
                PINC_Write(~PINC_Read());
                dato=0;
                break;
            }
            case 'z':{
                PIND_Write(~PIND_Read());
                dato=0;
                break;
            }
            case 's':{
                LCD_Position(0,0);
                LCD_PrintString("Hall
%");
                banderaS=true;
                break;
            }
            case 'h':{
                LCD_Position(0,0);
                LCD_PrintString("Room
%");
                banderaH=true;
                break;
            }
            case 'i':{
                LCD_Position(0,0);
                LCD_PrintString("ON
");
                banderaI=true;
                break;
            }
            case 'f':{
                LCD_Position(0,0);
                LCD_PrintString("OFF
");
                banderaF=true;
                break;
            }
            case 'm':{
                LCD_Position(0,0);
                LCD_PrintString("Select:
");
                banderaM=true;
                break;
            }
            default:
            {
                if (banderaS==true){
                    char
dimmer=((255*dato)/100);
                    LCD_Position(0,6);

LCD_PrintNumber(dato);

PWM_WriteCompare2(dimmer);
                    banderaS=false;
                }
                if (banderaH==true){
                    char
dimmer=((255*dato)/100);
                    LCD_Position(0,6);

LCD_PrintNumber(dato);

PWM_WriteCompare1(dimmer);
                    banderaH=false;
                }
                if (banderaI==true){
                    if(cont==true){
                        if(dato>9){//Solo
para impresion

LCD_PrintNumber(dato);
                        }else{

LCD_PrintNumber(0);

LCD_PrintNumber(dato);
                        }
                        banderaI=false;
                        cont=false;
                        //Transpaso de
datos
                        char a;

a=((dato/10)<<4)+dato%10;
                        switch(item){
                            case 1:{

wHoraInicio[1]=a;

if(dato==0){//Condicion de reset

item1[1]=false;
                                }
                                else{

item1[1]=true;
                                }
                                break;
                            }
                            case 2:{

xHoraInicio[1]=a;

if(dato==0){//Condicion de reset

item2[1]=false;
                                }
                                else{

item2[1]=true;
                                }
                                break;
                            }
                            case 3:{
```



```
yHoraInicio[1]=a;

if(dato==0){//Condicion de reset

item3[1]=false;
                                    }
                                    else{

item3[1]=true;
                                    }
                                    break;
                                }
                                case 4:{

zHoraInicio[1]=a;

if(dato==0){//Condicion de reset

item4[1]=false;
                                    }
                                    else{

item4[1]=true;
                                    }
                                    break;
                                }
                                default:{
                                    break;}
                            }

                        }else{

LCD_Position(0,5);

LCD_PrintNumber(dato);
                            LCD_PutChar(':');
                            cont=true;
                            //Poner datos
                            char a;

a=((dato/10)<<4)+dato%10;
                            switch(item){
                                case 1:{

wHoraInicio[2]=a;

if(dato==0){//Condicion de reset

item1[0]=false;
                                    }
                                    else{

item1[0]=true;
                                    }
                                    break;
                                }
                                case 2:{


xHoraInicio[2]=a;

if(dato==0){//Condicion de reset

item2[0]=false;
                                    }
                                    else{

item2[0]=true;
                                    }
                                    break;
                                }
                                case 3:{

yHoraInicio[2]=a;

if(dato==0){//Condicion de reset

item3[0]=false;
                                    }
                                    else{

item3[0]=true;
                                    }
                                    break;
                                }
                                case 4:{

zHoraInicio[2]=a;

if(dato==0){//Condicion de reset

item4[0]=false;
                                    }
                                    else{

item4[0]=true;
                                    }
                                    break;
                                }
                                default:{
                                    break;}
                            }

                        }

                    }
                    if (banderaF==true){
                        if(cont==true){
                            if(dato>9){

LCD_PrintNumber(dato);
                            }else{

LCD_PrintNumber(0);

LCD_PrintNumber(dato);
                            }
                            banderaF=false;
                            cont=false;
                            //Transpaso de
datos
```



```
                        char a;
a=((dato/10)<<4)+dato%10;
                        switch(item){
                            case 1:{
wHoraFin[1]=a;
if(dato==0){//Condicion de reset
item1[3]=false;
                                }
                                else{
item1[3]=true;
                                }
                                break;
                            }
                            case 2:{
xHoraFin[1]=a;
if(dato==0){//Condicion de reset
item2[1]=false;
                                }
                                else{
item2[1]=true;
                                }
                                break;
                            }
                            case 3:{
yHoraFin[1]=a;
if(dato==0){//Condicion de reset
item3[3]=false;
                                }
                                else{
item3[3]=true;
                                }
                                break;
                            }
                            case 4:{
zHoraFin[1]=a;
if(dato==0){//Condicion de reset
item4[3]=false;
                                }
                                else{
item4[3]=true;
                                }
                                break;
                            }
                            default:{
                                break;}
                            }
                        }else{
LCD_Position(0,5);
LCD_PrintNumber(dato);
                            LCD_PutChar(':');
                            cont=true;
                            //Poner datos
                            char a;
a=((dato/10)<<4)+dato%10;
                            switch(item){
                                case 1:{
wHoraFin[2]=a;
if(dato==0){//Condicion de reset
item1[2]=false;
                                }
                                else{
item1[2]=true;
                                }
                                break;
                            }
                            case 2:{
xHoraFin[2]=a;
if(dato==0){//Condicion de reset
item2[2]=false;
                                }
                                else{
item2[2]=true;
                                }
                                break;
                            }
                            case 3:{
yHoraFin[2]=a;
if(dato==0){//Condicion de reset
item3[2]=false;
                                }
                                else{
item3[2]=true;
                                }
                                break;
                            }
                            case 4:{
zHoraFin[2]=a;
if(dato==0){//Condicion de reset
```



```
item4[2]=false;
                                    }
                                    else{

item4[2]=true;
                                    }
                                    break;

                                }
                                default:{
                                    break;}
                            }

                        }
                    }
                    if (banderaM==true){
                        LCD_Position(0,8);

LCD_PrintNumber(dato);// muestra item
                        item=dato;
                        banderaM=false;
                    }
                    break;
                }
    }
    }else{
        if(dato=='*'){
            LCD_Position(0,0);

LCD_PrintString("Connect");//Se detecta
la ultima linea del inicio
            bandera=true;
        }

    }
}

void reloj(){

        LCD_Position(1,4);

LCD_PrintNumber(0x03&(ds.hour>>4));

LCD_PrintNumber((0b00001111)&ds.hour);
        LCD_PutChar(':');
        LCD_PrintNumber(ds.min>>4);

LCD_PrintNumber((0b00001111)&ds.min);
        LCD_PutChar(':');
        LCD_PrintNumber(ds.sec>>4);

LCD_PrintNumber((0b00001111)&ds.sec);
}

void  muestreo(){
    //Temperatura
    uint16 temp=0;
    aux[0]=aux[1];//Actualiza dato pasado
    ADC_StartConvert();
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
    temp=ADC_GetResult16();
    aux[1]=ADC_CountsTo_mVolts(temp);

if((2500>aux[0])&&(aux[1]>=2500)){//Nueva
medida es mayor a 2500 y la anterior era
menor 2500 ->50°
        char t[1];
        temp2=temp2+1;//Aumenta el numero
de alertas
        LCD_Position(0,0);
        LCD_PrintString("Warning T #");
        LCD_PrintNumber(temp2);
        UART_PutString("t");
        t[0]=(0xFF&temp2);
        UART_PutString(t);

    }
    temp=EEPROM_ReadByte(0);//Actualiza
aca ya que en la interupcion no se puede
    if(temp2!=temp){
        if(temp2>temp){
            EEPROM_WriteByte(temp2,0);
        }else{
            temp2=temp;
        }
    }

}

void comparacion(){
    char a[3]={};
    if(item1[0]||item1[1]){
//        LCD_Position(0,0);
//        LCD_PrintString("Puede
imprimir");

if((wHoraInicio[2]==((ds.hour)&(0b0011111
1)))&(wHoraInicio[1]==ds.min)&(ds.sec==0x
0)){
            PINA_Write(1);
            sprintf(a,"w%d",1);
            UART_PutString(a);
        }
    }
    if(item1[2]||item1[3]){

if((wHoraFin[2]==((ds.hour)&(0b00111111))
)&(wHoraFin[1]==ds.min)&(ds.sec==0x0)){
            PINA_Write(0);

            sprintf(a,"w%d",0);
            UART_PutString(a);
        }
    }
    if(item2[0]||item2[1]){

if((xHoraInicio[2]==((ds.hour)&(0b0011111
1)))&(xHoraInicio[1]==ds.min)&(ds.sec==0x
0)){
```



```
                PINB_Write(1);

                sprintf(a,"x%d",1);
                UART_PutString(a);
            }
        }
        if(item2[2]||item2[3]){

if((xHoraFin[2]==((ds.hour)&(0b00111111))
)&(xHoraFin[1]==ds.min)&(ds.sec==0x0)){
                PINB_Write(0);

                sprintf(a,"x%d",0);
                UART_PutString(a);
            }
        }
        if(item3[0]||item3[1]){

if((yHoraInicio[2]==((ds.hour)&(0b0011111
1)))&(yHoraInicio[1]==ds.min)&(ds.sec==0x
0)){
                PINC_Write(1);

                sprintf(a,"y%d",1);
                UART_PutString(a);
            }
        }
        if(item3[2]||item3[3]){

if((yHoraFin[2]==((ds.hour)&(0b00111111))
)&(yHoraFin[1]==ds.min)&(ds.sec==0x0)){
            PINC_Write(0);

                sprintf(a,"y%d",0);
                UART_PutString(a);
            }
        }
        if(item4[0]||item4[1]){

if((zHoraInicio[2]==((ds.hour)&(0b0011111
1)))&(zHoraInicio[1]==ds.min)&(ds.sec==0x
0)){
            PIND_Write(1);

                sprintf(a,"z%d",1);
                UART_PutString(a);
            }
        }
        if(item4[2]||item4[3]){

if((zHoraFin[2]==((ds.hour)&(0b00111111))
)&(zHoraFin[1]==ds.min)&(ds.sec==0x0)){
            PIND_Write(0);

                sprintf(a,"z%d",0);
                UART_PutString(a);
            }
        }

}
```

```
CY_ISR(InterrupISR1){
     char m[1];
          temp2=temp2+1;//Aumenta el numero
de alertas
          LCD_Position(0,0);
          LCD_PrintString("Warning M #");
          LCD_PrintNumber(temp2);
          UART_PutString("m");
          m[0]=(0xFF&temp2);
          UART_PutString(m);
          PINMOV_ClearInterrupt();
}

CY_ISR(InterrupA){
switch(Pin_A_Read()) {
         CyDelay(300);
          case 0b00111110:{
              PINA_Write(~PINA_Read());
              UART_PutChar('W');
              if (PINA_Read()== 1)
              {
                  UART_PutChar('1');
              }else{
                  UART_PutChar('0');
              }
               break;
          }
          case 0b00111101:{
              PINC_Write(~PINC_Read());
              UART_PutChar('Y');
              if (PINC_Read()== 1)
              {
                  UART_PutChar('1');
              }else{
                  UART_PutChar('0');
              }
              break;
          }
          case 0b00111011:{
              PIND_Write(~PIND_Read());
              UART_PutChar('Z');
              if (PIND_Read()== 1)
              {
                  UART_PutChar('1');
              }else{
                  UART_PutChar('0');
              }
              break;
          }
          case 0b00110111:{
              PINB_Write(~PINB_Read());
              UART_PutChar('X');
              if (PINB_Read()== 1)
              {
                  UART_PutChar('1');
              }else{
                  UART_PutChar('0');
              }

              break;
          }
```



```
        case 0b00101111:{

            if(dato<=90)
            {
                dato=dato+10;
            }

            UART_PutChar('Q');
            char dimmer=((255*dato)/100);
            UART_PutChar(dato);
            PWM_WriteCompare2(dimmer);
            break;
        }
        case 0b00011111:{

                if(dato>=10)
                {
                  dato=dato-10;
                }

            UART_PutChar('Q');
            char   dimmer=((255*dato)/100);
            UART_PutChar(dato);
            PWM_WriteCompare2(dimmer);
            break;
        }default:{
            break;
        }
    }
   Pin_A_ClearInterrupt();
}


int   main(void)
{
    CyGlobalIntEnable; /* Enable global
interrupts. */
    IRQRX_StartEx(InterrupRx);
    ISR_A_StartEx(InterrupA);

    UART_Start();
    LCD_Start();
    I2C_Start();
    PWM_Start();
    ADC_Start();
//    ds.sec =  0x00; //
//    ds.min =  0x48;//
//    ds.hour = 0b00001001;//Formato 24
horas bit 6 en 0 - 16 horas
//    ds.date = 0x03; // dia 2
//    ds.month = 0x03;//marzo
//    ds.year = 0x19; // 2019
//    ds.weekDay = 6; // Sunday: 6th day
of week considering monday as first day.;
//    DS_init();//Configura
//    DS_set_data();
    PINA_Write(0);
    PINB_Write(0);
    PINC_Write(0);
    PIND_Write(0);
    PWM_WriteCompare1(0);
    PWM_WriteCompare2(0);
    EEPROM_Start();
    EEPROM_WriteByte(0,0);//Comentar en
la presentacion


    for(;;)
    {
        reloj();
        DS_get_data();
        comparacion();
        muestreo();
        CyDelay(500);
//        LCD_Position(0,0);
//        LCD_PrintString("holiwi");
    }
}
```