Universidad del Valle de Guatemala
Natalia de León Bercián
Electrónica Digital 2
carne: 18193
Marzo 2021
sección: 20

Mini proyecto # 2

Comunicación I2C

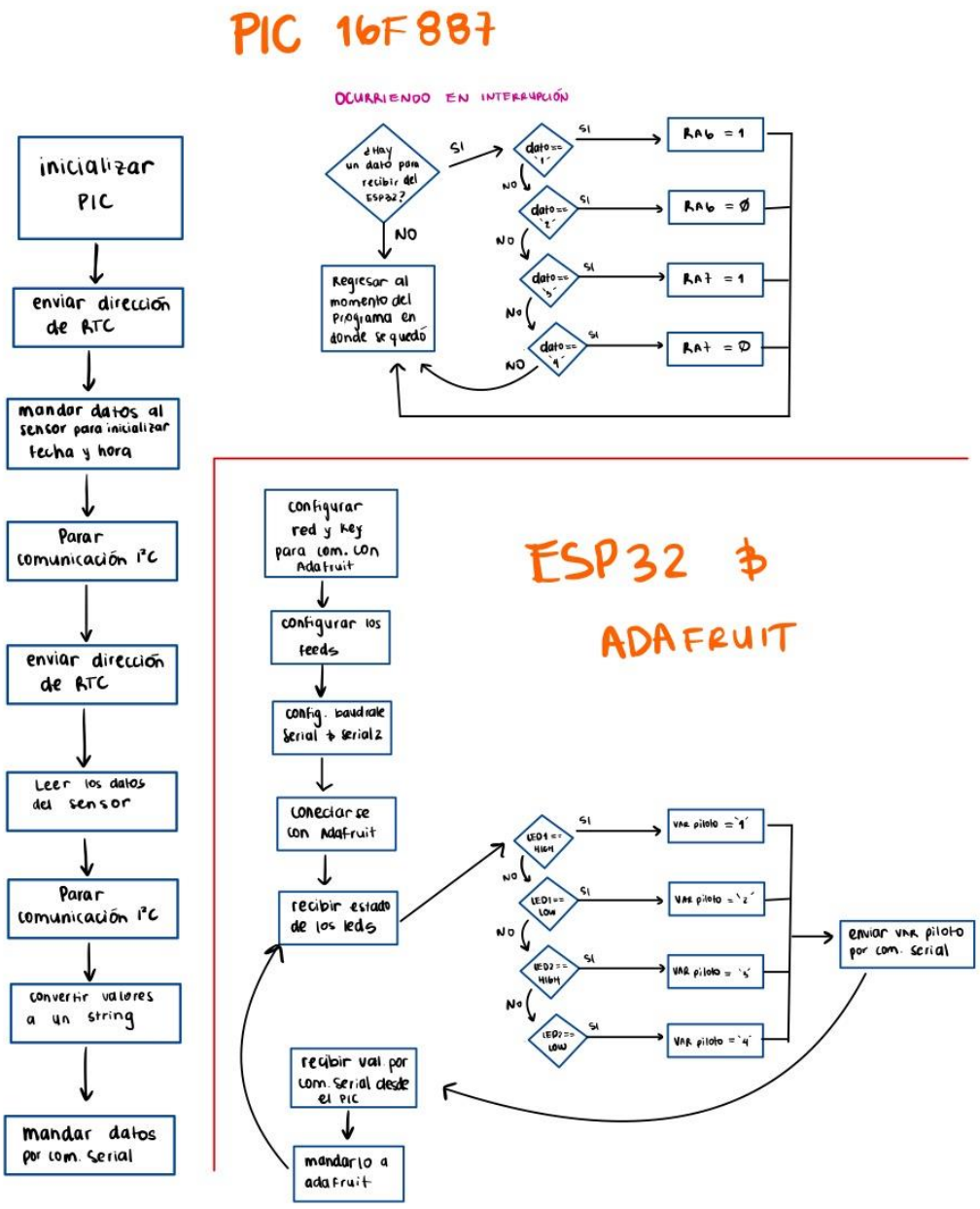
Link video

https://youtu.be/4Vr6KMc_0jg

Link Github

https://github.com/nataliadlb/LABS_REPOSITORIO.git

Diagrama de flujo



Pseudocodigo

PI16F887

```
/*
 * Código -- Mini proyecto 2
 * Author: Natalia de León Bercián
 * carné: 18193
 * Digital 2
 *
 * Created on 1 de marzo de 2021
 *
 * Las funciones relacionadas al sensor RTC fueron tomadas y luego modificadas
 * del sitio: https://simple-circuit.com/mplab-xc8-ds1307-ds3231-pic-mcu/
 * Autor: Simple Projects
 */



#define RS PORTEbits.RE0
#define RW PORTEbits.RE1
#define EN PORTEbits.RE2
#define D0 PORTDbits.RD0
#define D1 PORTDbits.RD1
#define D2 PORTDbits.RD2
#define D3 PORTDbits.RD3
#define D4 PORTDbits.RD4
#define D5 PORTDbits.RD5
#define D6 PORTDbits.RD6
#define D7 PORTDbits.RD7



//*****************************************************************************
//Librerias
//*****************************************************************************
#include <xc.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <pic16f887.h>
#include "USART.h"
#include "Oscilador.h"
#include "I2C.h"
//#include "RTC.h"


// CONFIG1
#pragma config FOSC = INTRC_NOCLKOUT// Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCLR pin function is MCLR)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)
```

```
// CONFIG2

#pragma config BOR4V = BOR40V    // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)

#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write protection off)


//************************************************************************

//Define

//************************************************************************

#define _XTAL_FREQ 8000000




//************************************************************************//

//VARIABLES                                                 //

//************************************************************************//

uint8_t  i, second, minute, hour, m_day, month, year;

char data_total[20];

char data_recive[1];

uint8_t cont;

char Time[20];

char Date[20];


//************************************************************************

//Prototipos de funciones

//************************************************************************


void setup(void);

void Write_to_RTC(void);

void Recive_from_RTC(void);

//************************************************************************

//Interrupciones

//************************************************************************


void __interrupt() ISR(void) {

  if(PIR1bits.RCIF == 1){

    data_recive = RCREG; //Recibe los datos que manda la terminal

    if (data_recive == '1'){ //auementa

      PORTAbits.RA6 = 1;

    }

    else if (data_recive == '2'){ //decrementa

      PORTAbits.RA6 = 0;

    }

    else if (data_recive == '3'){ //decrementa

      PORTAbits.RA7 = 1;

    }

    else if (data_recive == '4'){ //decrementa

      PORTAbits.RA7 = 0;

    }

    data_recive = 0;

    }

}


//************************************************************************

//Ciclo Principal

//************************************************************************


void main(void) {

  setup();
```

```
  while (1) {
    Write_to_RTC();
    Recive_from_RTC();
    Write_USART_String("TIME:"); //enviar el string con los valores de hora
    Write_USART_String("  ");
    Write_USART_String(hour_send);
    Write_USART_String(":");
    Write_USART_String(min_send);
    Write_USART_String(":");
    Write_USART_String(sec_send);
    Write_USART_String("  ");
    Write_USART_String(Date); //enviar el string con los valores de fecha
    Write_USART(13);//13 y 10 la secuencia es para dar un salto de linea
    Write_USART(10);
}

//***********************************************************************

//Funciones

//***********************************************************************



void Recive_from_RTC(void){
  // Convertir datos BCD a decimal
  second = bcd_to_decimal(second);
  minute = bcd_to_decimal(minute);
  hour   = bcd_to_decimal(hour);
  m_day  = bcd_to_decimal(m_day);
  month  = bcd_to_decimal(month);
  year   = bcd_to_decimal(year);
  // end conversion
  // uActualizar tiempo
  Time[6]  = hour   / 10 + '0';
  Time[7]  = hour   % 10 + '0';
  Time[9]  = minute / 10 + '0';
  Time[10] = minute % 10 + '0';
  Time[12] = second / 10 + '0';
  Time[13] = second % 10 + '0';

  // Actualizar fecha
  Date[6]  = m_day  / 10 + '0';
  Date[7]  = m_day  % 10 + '0';
  Date[9]  = month  / 10 + '0';
  Date[10] = month  % 10 + '0';
  Date[12] = year   / 10 + '0';
  Date[13] = year   % 10 + '0';


}

//Escribir valores iniciales al RTC, obtenido de Simple projects
void Write_to_RTC(void){
  I2C_Master_Start();        // start I2C
  I2C_Master_Write(0xD0);    // RTC chip address
  I2C_Master_Write(0);       // send register address
  I2C_Master_Write(0);              // reset seconds and start oscillator
  I2C_Master_Write(48);      // write minute value to RTC chip //y media
  I2C_Master_Write(6);      // write hour value to RTC chip
  I2C_Master_Write(1);       // write day value (not used)
  I2C_Master_Write(8);       // write date value to RTC chip
  I2C_Master_Write(3);       // write month value to RTC chip
  I2C_Master_Write(27);      // write year value to RTC chip
  I2C_Master_Stop();        // stop I2C
}
```

```
// ------ configuraciones ----- //
void setup(void) {
  //OSCCON = 0x07;
  ANSEL = 0; //RA0 y RA1 como analogico
  ANSELH = 0;
  TRISA = 0; //potenciometros, como entrada
  TRISB = 0b00000011;
  TRISCbits.TRISC6 = 0;
  TRISCbits.TRISC7 = 1;
  TRISD = 0;
  TRISE = 0;
  PORTA = 0;
  PORTB = 0;
  PORTC = 0;
  PORTD = 0;
  PORTE = 0;
  I2C_Master_Init(100000);
  USART_Init_BaudRate();
  USART_Init();
  USART_INTERRUPT();
}
```

ESP32

```
// Adafruit IO Digital Output Example
// Tutorial Link: https://learn.adafruit.com/adafruit-io-basics-digital-output
//
// Written by Todd Treece for Adafruit Industries
// Copyright (c) 2016 Adafruit Industries
// Licensed under the MIT license.
//
// All text above must be included in any redistribution.

/************************ Configuration ******************************/

// edit the config.h tab and enter your Adafruit IO credentials
// and any additional configuration needed for WiFi, cellular,
// or ethernet clients.
#include "config.h"

/************************ Example Starts Here *******************************/

#define RXD2 16
#define TXD2 17
#define IO_LOOP_DELAY 5000

unsigned long lastUpdate = 0;

String piloto  = " ";

AdafruitIO_Feed *LedPiloto1Feed = io.feed("LedPiloto1"); //DIGITAL
AdafruitIO_Feed *LedPiloto2Feed = io.feed("LedPiloto2"); //DIGITAL
AdafruitIO_Feed *ContadorFeed = io.feed("contador");

void setup() {
 Serial.begin(9600);
 Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);

 // wait for serial monitor to open
 while(! Serial);

 // connect to io.adafruit.com
 Serial.print("Connecting to Adafruit IO");
 io.connect();

 // the handleMessage function (defined below)
 // will be called whenever a message is
```

```
  // received from adafruit io.

  LedPiloto1Feed->onMessage(handleMessage1);
  LedPiloto2Feed->onMessage(handleMessage2);

  // wait for a connection
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  // we are connected
  Serial.println();
  Serial.println(io.statusText());
  LedPiloto1Feed->get();
  LedPiloto2Feed->get();

}
void loop() {
  io.run();
  if(Serial2.available()>0){
      cont = Serial2.read();
      Serial.print("sending -> ");
      Serial.println(cont);
      ContadorFeed->save(cont);
    }
//
//  if (millis() > (lastUpdate + IO_LOOP_DELAY)) {
//    // save count to the 'counter' feed on Adafruit IO
//    Serial.print("sending -> ");
//    Serial.println(cont);
//    ContadorFeed->save(cont);
//  }
//
//    // after publishing, store the current time
//    lastUpdate = millis();

    delay(3000);
}
// this function is called whenever an 'digital' feed message
// is received from Adafruit IO. it was attached to
// the 'digital' feed in the setup() function above.
void handleMessage1(AdafruitIO_Data *data) {
  Serial.println("--------");
  Serial.println("Piloto 1");
  Serial.print("received <- ");
  //Serial.println(data->value());

  if(data->toString() == "ON"){
    Serial.println("HIGH");
    piloto = "1";
    Serial2.write('1'); //49 EN ASCII
  }
  else{
    Serial.println("OFF");
    piloto = "2";
    Serial2.write('2');
  }
  Serial.println(" ");
  Serial.print("valor: ");
  Serial.println(piloto);
  Serial.println(" ");
}

void handleMessage2(AdafruitIO_Data *data) {
  Serial.println("--------");
  Serial.println("Piloto 2");
```

```
  Serial.print("received <- ");
  //Serial.println(data->value());

  if(data->toString() == "ON"){
    Serial.println("HIGH");
    piloto = "3";
    Serial2.write('3');

  }
  else{
    Serial.println("OFF");
    piloto = "4";
    Serial2.write('4');

  }
  Serial.println(" ");
  Serial.print("valor: ");
  Serial.println(piloto);
  Serial.println(" ");
}
```