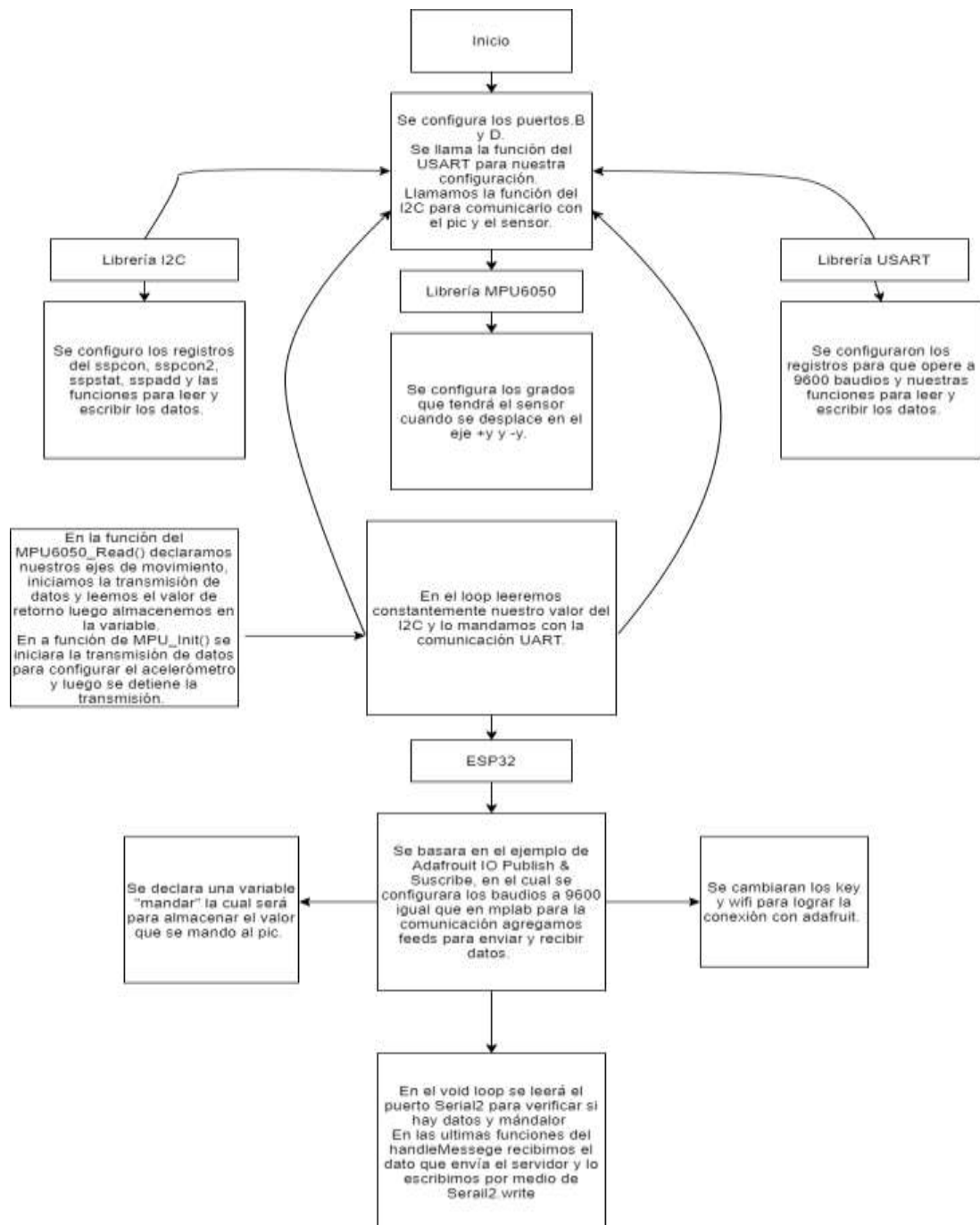


MiniProyecto#2

Diagrama de flujo:



Link de github:

https://github.com/Helder1121/Labsdigitaldos/tree/main/Mini_proyectosdos

Link de youtube:

<https://www.youtube.com/watch?v=c1iKabxvNJ0>

Link de Adafruit:

<https://io.adafruit.com/Helder1131/dashboards>

Progra comentada:

Principal:

```
//*****  
  
// Palabra de configuración  
  
//*****  
  
// CONFIG1  
  
#pragma config FOSC = INTRC_NOCLKOUT // Oscillator Selection bits (INTOSCIO oscillator: I/O  
function on RA6/OSC2/CLKOUT pin, I/O function on 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 = ON // Internal External Switchover bit (Internal/External Switchover  
mode is enabled)  
  
#pragma config FCMEN = ON // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is  
enabled)  
  
#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 = BOR21V // Brown-out Reset Selection bit (Brown-out Reset set to 2.1V)

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

//*****

// Importación de librerías

//*****

#include <xc.h>

#include <stdio.h>

//#include "config.h"

#include "USART.h"

#include "MPU.h"

//*****

// Variables

//*****

#define _XTAL_FREQ 8000000

//*****

// Ciclo principal

//*****

void main(void){

    UART_TX_Init();

    TRISD2 = 0; // LED Indicador

    TRISB = 0; // Leds conectadas para demostrar la variable ay

    ANSELH = 0;

```

```

MPU6050_Init();

//*****

// Loop principal

//*****

while(1)
{
    RD2 = ~RD2; // Blink LED verde

    MPU6050_Read();

    __delay_ms(50);
}

return;
}

//Se hizo con oscilador externo al inicio pero no era tan estable y mejor se
//hizo con uno interno.

```

Librerias:

```

/*
* File:  USART.c
* Author: betov
*
* Created on 22 de febrero de 2021, 07:53 AM
*/

//Extraido de:

//https://deepbluembedded.com/mpu6050-with-microchip-pic-accelerometer-gyroscope-
interfacing-with-pic/

//y combinacion del proyecto pasado que tambien era de:

////Extraido de https://electrosome.com/

#include <xc.h>

#include "USART.h"

```

```
//-----[ UART Routines ]-----
```

```
//-----
```

```
void UART_TX_Init(void)
```

```
{
```

```
    BRGH = 0; // Set For High-Speed Baud Rate
```

```
    SPBRG = 12; // Set The Baud Rate To Be 9600 bps
```

```
    //--[ Enable The Ascynchronous Serial Port ]--
```

```
    SYNC = 0;
```

```
    SPEN = 1;
```

```
    //--[ Set The RX-TX Pins to be in UART mode (not io) ]--
```

```
    TX_D = 1;
```

```
    RX_D = 1;
```

```
    TXEN = 1; // Enable UART Transmission
```

```
}
```

```
void UART_Write(unsigned char data)
```

```
{
```

```
    while(!TRMT);
```

```
    TXREG = data;
```

```
}
```

```
void UART_Write_String(char* buf)
```

```
{
```

```
    int i=0;
```

```
    while(buf[i] != '\0')
```

```
        UART_Write(buf[i++]);
```

```

}

/*
 * File: MPU.c
 * Author: betov
 *
 * Created on 9 de marzo de 2021, 01:08 AM
 */

//Extrado de: https://deepbluembedded.com/mpu6050-with-microchip-pic-accelerometer-gyroscope-interfacing-with-pic/

#include <xc.h>

#include "I2c.h"

#include "MPU.h"

#include "USART.h" // for debugging serial terminal

#include <stdio.h>

//-----[ MPU6050 Routines ]-----
//-----

void MPU6050_Init()
{
    // Power-Up Delay & I2C_Init
    __delay_ms(100);
    I2C_Master_Init();

    // Setting The Sample (Data) Rate
    I2C_Start(0xD0);
    I2C_Master_Write(SMPLRT_DIV);
    I2C_Master_Write(0x07);
    I2C_Master_Stop();

```

```
// Setting The Clock Source
```

```
I2C_Start(0xD0);
```

```
I2C_Master_Write(PWR_MGMT_1);
```

```
I2C_Master_Write(0x01);
```

```
I2C_Master_Stop();
```

```
// Configure The DLPF
```

```
I2C_Start(0xD0);
```

```
I2C_Master_Write(CONFIG);
```

```
I2C_Master_Write(0x00);
```

```
I2C_Master_Stop();
```

```
// Configure The ACCEL (FSR= +-2g)
```

```
I2C_Start(0xD0);
```

```
I2C_Master_Write(ACCEL_CONFIG);
```

```
I2C_Master_Write(0x00);
```

```
I2C_Master_Stop();
```

```
// Configure The GYRO (FSR= +-2000d/s)
```

```
I2C_Start(0xD0);
```

```
I2C_Master_Write(GYRO_CONFIG);
```

```
I2C_Master_Write(0x18);
```

```
I2C_Master_Stop();
```

```
// Enable Data Ready Interrupts
```

```
I2C_Start(0xD0);
```

```
I2C_Master_Write(INT_ENABLE);
```

```
I2C_Master_Write(0x01);
```

```

    I2C_Master_Stop();
}

void MPU6050_Read()
{
    char buffer[40];
    int Ax,Ay,Az,T,Gx,Gy,Gz;
    // Prepare For Reading, Starting From ACCEL_XOUT_H
    I2C_Start(0xD0);
    I2C_Master_Write(ACCEL_XOUT_H);
    I2C_Master_Stop();
    I2C_Start(0xD1);
    Ax = ((int)I2C_Read(0)<<8) | (int)I2C_Read(0);
    Ay = ((int)I2C_Read(0)<<8) | (int)I2C_Read(0);
    Az = ((int)I2C_Read(0)<<8) | (int)I2C_Read(0);
    T = ((int)I2C_Read(0)<<8) | (int)I2C_Read(0);
    Gx = ((int)I2C_Read(0)<<8) | (int)I2C_Read(0);
    Gy = ((int)I2C_Read(0)<<8) | (int)I2C_Read(0);
    Gz = ((int)I2C_Read(0)<<8) | (int)I2C_Read(1);
    I2C_Master_Stop();

    PORTB = (Ay+16384)/128;//Conversion para los datos de la variable ay del
    //acelerometro

    sprintf(buffer," Ay = %d  ",Ay);
    UART_Write_String(PORTB);
}

/*
* File: I2c.c

```



```
* Author: betov
```

```
*
```

```
* Created on 9 de marzo de 2021, 01:09 AM
```

```
*/
```

```
//Extraido de: https://deepbluembedded.com/mpu6050-with-microchip-pic-accelerometer-gyroscope-interfacing-with-pic/
```

```
#include <xc.h>
```

```
#include "I2c.h"
```

```
//-----[ I2C Routines ]-----
```

```
//-----
```

```
void I2C_Master_Init()
```

```
{
```

```
    SSPCON = 0x28;
```

```
    SSPCON2 = 0x00;
```

```
    SSPSTAT = 0x00;
```

```
    SSPADD = ((_XTAL_FREQ/4)/I2C_BaudRate) - 1;
```

```
    SCL_D = 1;
```

```
    SDA_D = 1;
```

```
}
```

```
void I2C_Master_Wait()
```

```
{
```

```
    while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));
```

```
}
```

```
void I2C_Master_Start()
```

```
{
```

```
    I2C_Master_Wait();  
    SEN = 1;  
}
```

```
void I2C_Start(char add)  
{  
    I2C_Master_Wait();  
    SEN = 1;  
    I2C_Master_Write(add);  
}
```

```
void I2C_Master_RepeatedStart()  
{  
    I2C_Master_Wait();  
    RSEN = 1;  
}
```

```
void I2C_Master_Stop()  
{  
    I2C_Master_Wait();  
    PEN = 1;  
}
```

```
void I2C_ACK(void)  
{  
    ACKDT = 0;                // 0 -> ACK  
    ACKEN = 1;                // Send ACK  
    while(ACKEN);  
}
```

```
void I2C_NACK(void)
```

```
{  
    ACKDT = 1;           // 1 -> NACK  
    ACKEN = 1;           // Send NACK  
    while(ACKEN);  
}
```

```
unsigned char I2C_Master_Write(unsigned char data)
```

```
{  
    I2C_Master_Wait();  
    SSPBUF = data;  
    while(!SSPIF); // Wait Until Completion  
    SSPIF = 0;  
    return ACKSTAT;  
}
```

```
unsigned char I2C_Read_Byte(void)
```

```
{  
    //---[ Receive & Return A Byte ]---  
    I2C_Master_Wait();  
    RCEN = 1;           // Enable & Start Reception  
    while(!SSPIF);     // Wait Until Completion  
    SSPIF = 0;           // Clear The Interrupt Flag Bit  
    I2C_Master_Wait();  
    return SSPBUF;      // Return The Received Byte  
}
```

```
unsigned char I2C_Read(unsigned char ACK_NACK)
```

```

{
  //---[ Receive & Return A Byte & Send ACK or NACK ]---
  unsigned char Data;

  RCEN = 1;

  while(!BF);

  Data = SSPBUF;

  if(ACK_NACK==0)
    I2C_ACK();

  else
    I2C_NACK();

  while(!SSPIF);

  SSPIF=0;

  return Data;
}

```

Porgra de arduino:

```

// Adafruit IO Publish & Subscribe Example
//
// Adafruit invests time and resources providing this open source code.
// Please support Adafruit and open source hardware by purchasing
// products from Adafruit!
//
// 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 *****/

// this int will hold the current count for our sketch
int mandar = 0;
int LED2 = 0;
#define LED_PIN 2
#define RXD2 16
#define TXD2 17
// Track time of last published messages and limit feed->save events to once
// every IO_LOOP_DELAY milliseconds.
//
// Because this sketch is publishing AND subscribing, we can't use a long
// delay() function call in the main loop since that would prevent io.run()
// from being called often enough to receive all incoming messages.
//
// Instead, we can use the millis() function to get the current time in
// milliseconds and avoid publishing until IO_LOOP_DELAY milliseconds have
// passed.
#define IO_LOOP_DELAY 5000
unsigned long lastUpdate = 0;

// set up the 'counter' feed
AdafruitIO_Feed *mandarFeed = io.feed("mandar");

```

```
AdafruitIO_Feed *recibirFeed = io.feed("recibir");  
AdafruitIO_Feed *LEDREDFeed = io.feed("LEDRED");
```

```
void setup() {  
  pinMode(LED_PIN, OUTPUT);  
  pinMode(22, OUTPUT);  
  pinMode(23, OUTPUT);  
  // start the serial connection  
  Serial.begin(9600);  
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);  
  delay(1000);  
  
  // wait for serial monitor to open  
  while (! Serial);  
  
  Serial.print("Connecting to Adafruit IO");  
  
  // connect to io.adafruit.com  
  io.connect();  
  
  // set up a message handler for the count feed.  
  // the handleMessage function (defined below)  
  // will be called whenever a message is  
  // received from adafruit io.  
  recibirFeed->onMessage(handleMessage);  
  LEDREDFeed->onMessage(handleMessage2);  
  
  // wait for a connection  
  while (io.status() < AIO_CONNECTED) {
```

```
Serial.print(".");  
delay(500);  
}
```

```
// we are connected  
Serial.println();  
Serial.println(io.statusText());  
recibirFeed->get();  
LEDREDFeed->get();  
  
}
```

```
void loop() {  
  if(Serial.available()){  
    Serial.write("-");  
    Serial2.write(Serial.read());  
    //LED  
  
  }  
  if(Serial2.available()> 0){  
    //Serial.write(Serial2.read());  
    mandar = Serial2.read();  
    Serial.println(Serial2.read());  
  
  }
```

```
// io.run(); is required for all sketches.  
// it should always be present at the top of your loop  
// function. it keeps the client connected to
```

```
// io.adafruit.com, and processes any incoming data.
```

```
io.run();
```

```
//enviar=random(0,100);
```

```
if (millis() > (lastUpdate + IO_LOOP_DELAY)) {
```

```
    // save count to the 'counter' feed on Adafruit IO
```

```
    Serial.print("Valor de PIC -> ");
```

```
    Serial.println(mandar); //Valor que se manda del pic
```

```
    mandarFeed->save(mandar);
```

```
    // increment the count by 1
```

```
    //count++;
```

```
    // after publishing, store the current time
```

```
    lastUpdate = millis();
```

```
}
```

```
delay(3000);
```

```
}
```

```
// this function is called whenever a 'counter' message
```

```
// is received from Adafruit IO. it was attached to
```

```
// the counter feed in the setup() function above.
```

```
void handleMessage(AdafruitIO_Data *data) {
```

```
    Serial.print("received <- ");
```

```
    Serial.println(data->value());
```

```
    if (data->toString() == "ON") { //enciende el LED azul
```



```

    digitalWrite(LED_PIN, HIGH);
}
if (data->toString() == "OFF") { //apaga el LED azul
    digitalWrite(LED_PIN, LOW);
}
}

void handleMessage2(AdafruitIO_Data *data) {

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

    if(data->toString() == "ON"){ //enciende el LED azul
        digitalWrite(22, HIGH);
        Serial.print("LEDRED");
    }

    if(data->toString() == "OFF"){ //apaga el LED azul
        digitalWrite(22, LOW);
        Serial.print("LEDRED");
    }
}

```