UNIVERSIDAD DEL VALLE DE GUATEMALA
PROCESAMIENTO DE SEÑALES
ELECTRÓNICA Y MECATRÓNICA
Mini Proyecto 2
Raúl Aguilar
17581

Pseudocodigo

PIC 16F887

Librerías:

• I2C

• OSC

• USART

Puertos:

- ANSEL = 0; // Puerto A
- TRISA = 0; // Puerto a OUT
- PORTA = 0; // Puerto A entrada apagado
- ANSELH = 0; // Puerto B
- TRISB = 0; // Puerto B
- PORTB = 0; // Puerto B apagado
- TRISC = 0; // Puerto C OUT
- PORTC = 0; // Puerto C apagado
- TRISCbits.TRISC6 = 0; // TX salida

- TRISD = 0; // Puerto D salida
- PORTD = 0; // Puerto D apagados
- TRISE = 0; // Puerto E salida
- PORTE = 0; // Puerto E apagado
- TRISEbits.TRISE0 = 0; // TX salida
- TRISEbits.TRISE3 = 0; // TX salida
- InitOSC(7);

Programa principal:

Empieza la comunicación por I2C. Se lee el dato del sensor y se guarda en una variable.

- I2C_M_Start();
- sensor = I2C_M_Read (0);
- I2C_M_Stop();I2C_M_Read;

- Se manda el dato pro USART
- sprintf(buffer, "%d \r\n",sensor);
- USART_SendString(buffer);

Se hace la configuración del USART:

- void USART_config(void)
- USART_lib_config(); // USART
- void escribir_char (uint8_t valor)
- TXREG = valor; // Se envia Byte a TXREG
- while (PIR1bits.TXIF == 0); // Espera a que se haya enviado dato
- void USART_SendString(const char *str)

- while(*str!='\0')
- escribir_char(*str);
- str++;
- char leer_char(void)
- if (RCSTAbits.OERR ==0)

- CREN = 0; // Apagar modulo para apagar error
- NOP();
- CREN = 1; // Enciende una vez no haya error

- return (RCREG); // Se envia valor a RCREG

UNIVERSIDAD DEL VALLE DE GUATEMALA
PROCESAMIENTO DE SEÑALES
ELECTRÓNICA Y MECATRÓNICA
Mini Proyecto 2
Raúl Aguilar
17581

ESP32

```
include "config.h"

#define RXD2 16

#define TXD2 17

#define LED_PIN 2

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

// this int will hold the current count for our sketch

int count = 0;

int DATA =0;

int LED1 = 1;

int contador = 0;

int Temp = 0;

// set up the 'counter' feed

//AdafruitIO_Feed *counter = io.feed("counter");

//AdafruitIO_Feed *ContadorFeed = io.feed("Contador");

AdafruitIO_Feed *EstadoFeed = io.feed("Temperatura");

void setup() {

 // start the serial connection
```

UNIVERSIDAD DEL VALLE DE GUATEMALA
PROCESAMIENTO DE SEÑALES
ELECTRÓNICA Y MECATRÓNICA
Mini Proyecto 2
Raúl Aguilar
17581

```
 Serial.begin(115200);

 Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);


 // wait for serial monitor to open

 while(! Serial);


 Serial.print("Connecting to Adafruit IO");


 // connect to io.adafruit.com

 io.connect();


 // wait for a connection

 while(io.status() < AIO_CONNECTED) {

  Serial.print(".");

  Serial.println(io.statusText());

    delay(500);

 }


 // we are connected

 Serial.println();

 Serial.println(io.statusText());


}


void loop() {
```

UNIVERSIDAD DEL VALLE DE GUATEMALA
PROCESAMIENTO DE SEÑALES
ELECTRÓNICA Y MECATRÓNICA
Mini Proyecto 2
Raúl Aguilar
17581


```
 // 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();
 //DATAFeed->save(DATA);
 //DATA++;

 // save count to the 'counter' feed on Adafruit IO
 //Serial.print("sending -> ");
 //Serial.println(contador);
  delay(3000);
  while (Serial2.available()){
  Temp = Serial2.read();
  }
  EstadoFeed->save(Temp);
  delay(3000);
 // increment the count by 1
 //count++;

 // Adafruit IO is rate limited for publishing, so a delay is required in
 // between feed->save events. In this example, we will wait three seconds
 // (1000 milliseconds == 1 second) during each loop.
 delay(3000);

}
```