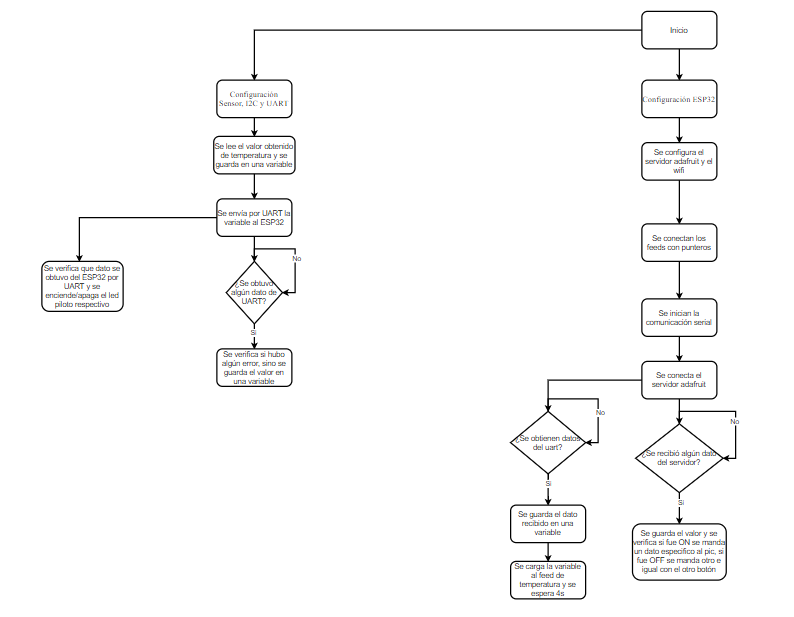
Universidad del Valle de Guatemala Cristopher René Sagastume 18640  
Electrónica Digital 2  
Sección 30 26/03/2021

**Diagrama General**



\*\* se adjunta también un PDF con el diagrama para mejor visualización

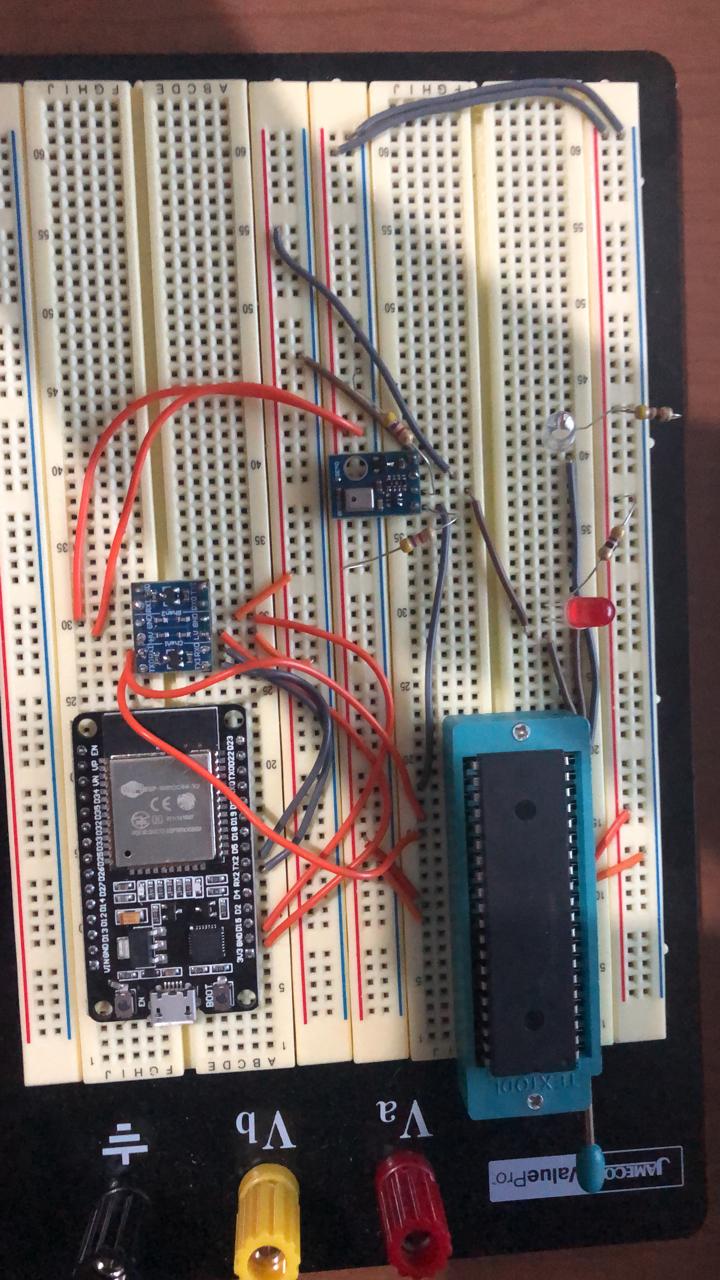
**Pseudocódigo ESP32**

1. Configuración Adafruit y wifi
2. Se configuran los feed con punteros
3. Se inicia la comunicación serial con el UART2 del ESP32
4. Se conecta el servidor Adafruit
5. Se verifica si se obtuvo algún cambio en los botones del servidor **(esto se hace en paralelo a todo)**
6. Si se obtuvo algún cambio se guarda en el puntero para cargarlo al feed
7. También en esa misma cargada de valor al feed se verifica que fue exactamente lo que se obtuvo y dependiendo lo que se obtuvo se envía un dato al pic por UART
8. Se verifica si existe algún dato sin leer **(esto se hace en paralelo a todo)**
9. Si hubiera dato se guarda en una variable
10. Luego esa variable se envía al puntero para guardarlo al feed de temperatura.

**Pseudocódigo PIC**

1. Configuramos pines,Master I2C y UART
2. Leemos el valor obtenido del sensor por I2C
3. Enviamos el valor obtenido por UART al ESP32
4. verificamos si el valor obtenido es “+”
   1. si es + encendemos el pin RD0 (led rojo)
   2. si es – apagamos el pin RD0(rojo)
5. verificamos si el valor obtenido es “/”
   1. si es / encendemos el pin RD1(led blanco)
   2. si es & apagamos el pin RD1(led blanco)
6. en paralelo a todo esto se tiene la interrupción
7. verificamos si hubo interrupción de recepción UART
   1. si la hubo vemos si hubo error de overrun
   2. sino guardamos el valor de RCREG en la variable de recibido

**Esquemático Físico**

****

**Código comentado**

**Main**

/\*

\* File: Proyecto 2.c

\* Author: Cristopher Sagastume 18640

\*

\* Created on 20 de marzo de 2021, 08:46 AM

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// PALABRA DE CONFIGURACIÓN

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// CONFIG1

#pragma config FOSC = INTRC\_CLKOUT// Oscillator Selection bits (INTOSC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)

#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 enabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is enabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is enabled)

#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3/PGM pin has PGM function, low voltage programming enabled)

// 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)

// #pragma config statements should precede project file includes.

#include <stdio.h>

#include <xc.h>

#include<stdint.h>

#include "I2C.h"

#include "UART.h"

#include "sensor.h"

#define \_XTAL\_FREQ 8000000

uint8\_t Temp=0;

uint8\_t recibido=0;

void setup(void);

void \_\_interrupt() ISR(void);

void main(void) {

//configuración de pines, comunicación I2C y comunicación UART

setup();

Config\_USARTT();

recibir();

sensor\_init();

while(1){

//se lee la temperatura del sensor

Temp=read(TEMPERATURE\_ADDRESS);

//se envia la temperatura por uart al esp32

enviar(Temp);

//si el dato recibido de la uart es "+" se enciende un led RD0

if (recibido == 0x2B){

PORTDbits.RD0=1;

}

//si el dato recibido de la uart es "-" se apaga un led RD0

else if(recibido == 0x2D){

PORTDbits.RD0=0;

}

//si el dato recibido de la uart es "/" se apaga un led RD1

if (recibido == 0x2F){

PORTDbits.RD1=1;

}

//si el dato recibido de la uart es "&" se apaga un led RD1

else if (recibido ==0x26){

PORTDbits.RD1=0;

}

}

}

void setup(void) {

TRISD = 0b00000000;// puerto D como salida

TRISC = 0b10000000; //activamos el RX como entrada

PORTC = 0;//limpiamos puertos

PORTD = 0;

}

void \_\_interrupt() ISR(void) {

if (PIR1bits.RCIF == 1){//verificamos si fue interrupcion de la recepción USART

if (RCSTAbits.OERR == 1) {//verificamos si hubo algún error de overrun

RCSTAbits.CREN = 0;

\_\_delay\_us(300);

} else {

recibido = RCREG; //guardamos el valor recibido en una variable

}

}

}

**Sensor**

/\*

\* File: sensor.c

\* Author: SAGASTUME

\*

\* Created on 26 de marzo de 2021, 08:12 PM

\*/

#include <xc.h>

#include "I2C.h"

#include "sensor.h"

//se inicia la comunicación i2c con el sensor

void sensor\_init() {

I2C\_init(100000); //inicio a 100Kb/s

}

//se lee los datos del sensor

float read(char regAddress) {

signed int MSB, LSB;

//se inicia la comunicación i2c

I2C\_start();

//se escribe la dirección del sensor y se le indica que se le escribirá

I2C\_write(SENSOR\_ADDRESS + 0); // 0 = Write

//se le indica la dirección que le se utilizará

I2C\_write(regAddress);

//se reinicia la comunicación

I2C\_restart();

//se escribe la dirección del sensor y se le indica que se leera

I2C\_write(SENSOR\_ADDRESS + 1); // 1 = Read

//se guarda el valor de la lectura del i2c en 2 variables

MSB = I2C\_read(ACK);

LSB = I2C\_read(NACK);

//se detiene la comunicación

I2C\_stop();

//se convierte el valor recibido a float

return toFloat((MSB << 8) + LSB);

}

//------------------------------------------------------------------------------

float toFloat(signed int tempr) {

float result = (float) (tempr >> 8);//Se descarta el byte menos significativo

return (result);

}

//------------------------------------------------------------------------------

**I2C**

/\*

\* File: I2C.c

\* Author: Chisanga

\* Editor: SAGASTUME

\* Created on 06 May 2020, 11:33 AM

\*/

#include <xc.h>

#define \_XTAL\_FREQ 8000000

#include "I2C.h"

//------------------------------------------------------------------------------

#define ACK 0

#define NACK 1

//--------------Function Purpose: Configure I2C module--------------------------

void I2C\_init(unsigned long speed){

SCK\_dir = 1; // SCK pins input

SDA\_dir = 1; // Make SDA and

SSPADD = ((\_XTAL\_FREQ/4000)/speed) - 1; //velocidad de baudrate

SSPSTAT = 0x80; // Slew Rate control is disabled

SSPCON = 0x28; // Select and enable I2C in master mode

}

//---------------Function Purpose: I2C\_Start sends start bit sequence-----------

void I2C\_start(void){

SEN = 1; // Send start bit

while(!SSPIF); // Wait for it to complete

SSPIF = 0; // Clear the flag bit

}

//---------------Function Purpose: I2C\_ReStart sends start bit sequence---------

void I2C\_restart(void){

RSEN = 1; // Send Restart bit

while(!SSPIF); // Wait for it to complete

SSPIF = 0; // Clear the flag bit

}

//---------------Function : I2C\_Stop sends stop bit sequence--------------------

void I2C\_stop(void){

PEN = 1; // Send stop bit

while(!SSPIF); // Wait for it to complete

SSPIF = 0; // Clear the flag bit

}

//---------------Function : Send ACK/NACK bit sequence--------------------------

void I2C\_ack(char acknowledge){

ACKDT = acknowledge; // 0 means ACK, 1 means NACK

ACKEN = 1; // Send ACKDT value

while(!SSPIF); // Wait for it to complete

SSPIF = 0; // Clear the flag bit

}

//---------------Function Purpose: I2C\_Write\_Byte transfers one byte------------

unsigned char I2C\_write(unsigned char data){

SSPBUF = data; // Send Byte value

while(!SSPIF); // Wait for it to complete

SSPIF = 0; // Clear the flag bit

return ACKSTAT; // Return ACK/NACK from slave

}

//----------------Function Purpose: I2C\_Read\_Byte reads one byte----------------

unsigned char I2C\_read(char acknowledge){

RCEN = 1; // Enable reception of 8 bits

while(!SSPIF); // Wait for it to complete

SSPIF = 0; // Clear the flag bit

I2C\_ack(acknowledge);

return SSPBUF; // Return received byte

}

**Uart**

/\*

\* File: UART.c

\* Author: SAGASTUME

\*

\* Created on 9 de febrero de 2021, 12:41 AM

\*/

#include <xc.h>

#define \_XTAL\_FREQ 8000000

void Config\_USARTT(void) {

TXSTAbits.SYNC = 0; //modo asincrono

TXSTAbits.TXEN = 1; //activamos la transmisión

TXSTAbits.BRGH = 0; //velocidad baja de baud rate (velocidad de trabajo de la comuniación)

BAUDCTLbits.BRG16 = 1; //geneardor de 16 bits de baud rate

SPBRG = 25; // tener un baudrate a 19230

RCSTAbits.SPEN = 1; //se activa la comunación del RX/TX

INTCONbits.GIE = 1; //se activan la interrupción global

return;

}

void enviar(char \*valor) {

TXREG = valor[0];

while (TRMT == 0) {//bucle si TRMT sigue lleno

}

TXREG = valor[1];//si TRMT esta vacio le cargamos otro valor al TXREG->TRMT

while (TRMT == 0) {

}

TXREG = valor[2];

while (TRMT == 0) {

}

TXREG = valor[3];

while (TRMT == 0) {

}

TXREG = 0x20;

while (TRMT == 0) {

}

}

void recibir(void) {

RCSTAbits.CREN = 1; //activamos el recibidor contunio asincrono

RCSTAbits.FERR = 0; //apagamos el error de frame bit

PIE1bits.RCIE = 1; //ecendemos la interrupción de recepción de la comunicación USART

RCSTAbits.OERR = 0; //apagamos el overrun error

}

**ESP32**

//autor: SAGASTUME 18640

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Configuration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//se importa la libreria de configuración y se define los pines de RXD2 y TXD2 del esp32

#include "config.h"

#define RXD2 16

#define TXD2 17

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

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

#define IO\_LOOP\_DELAY 5000

unsigned long lastUpdate = 0;

// set up the 'counter' feed

AdafruitIO\_Feed \*LED = io.feed("LED1");

AdafruitIO\_Feed \*LED2 = io.feed("LED2");

AdafruitIO\_Feed \*temperatura = io.feed("temp");

void setup() {

// start the serial connection

Serial.begin(115200);

Serial2.begin(19200, 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();

// 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.

LED->onMessage(handleMessage);

LED2->onMessage(handleMessage2);

// wait for a connection

while (io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

// we are connected

Serial.println();

Serial.println(io.statusText());

LED->get();

LED2->get();

}

void loop() {

// 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();

if (millis() > (lastUpdate + IO\_LOOP\_DELAY)) {

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

if (Serial2.available()) {

temp=Serial2.read();

Serial.print("sending -> ");

Serial.println(temp);

temperatura->save(temp);

delay(4000);

}

}

}

void handleMessage(AdafruitIO\_Data \*data) {

Serial.print("received <- ");

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

if (data->toString() == "ON") {

Serial.println("ON");

Serial2.write("+");

} else if (data->toString() == "OFF") {

Serial.println("OFF");

Serial2.write("-");

}

delay(2000);

}

void handleMessage2(AdafruitIO\_Data \*data) {

Serial.print("received <- ");

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

if (data->toString() == "ON") {

Serial.println("/");

Serial2.write("/");

} else if (data->toString() == "OFF") {

Serial.println("&");

Serial2.write("&");

}

delay(2000);

}

**Link del repositorio**

<https://github.com/sag18640/Electronica_Digital_2.git>