Carlos Cuellar #19275
Digital 2
Sección 10

Proyecto #2: I2C y Comunicación WIFI

Vídeo demostración: https://youtu.be/_bHFnsbYdas
Link de Repositorio: https://github.com/Cue19275/Digital2

Pseudocódigo

Pic16F887

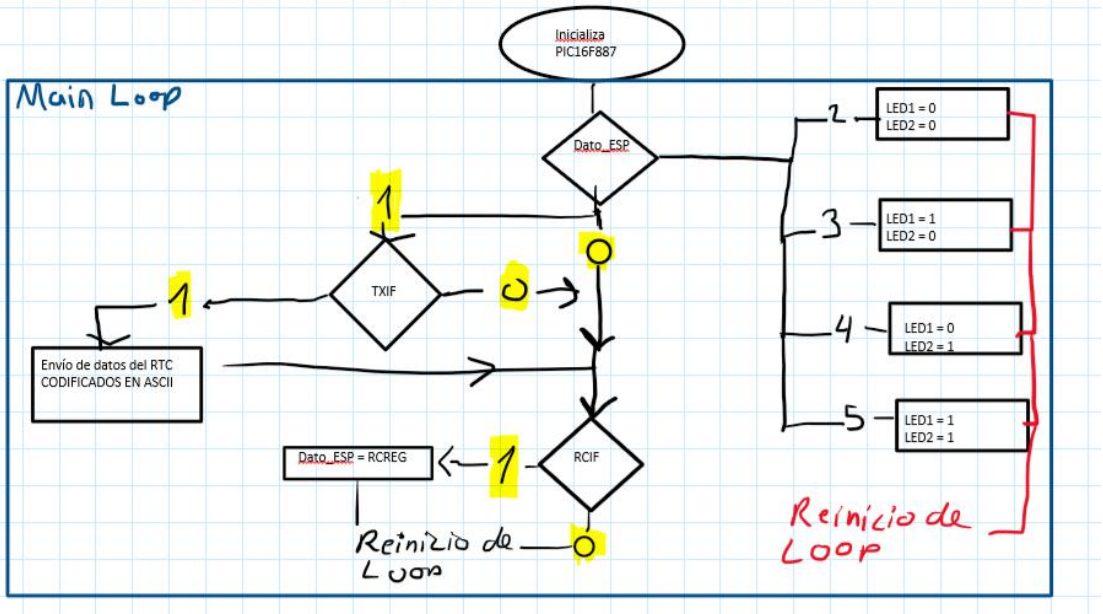
- Conexión I2C con el RTC
- Se debe de crear una rutina tanto de escritura de datos cómo de lectura
- Se debe de decodificar los datos del RTC
  BCD → ASCII
- USART para mandar y recibir datos del Esp32
- Salidas digitales para las luces piloto

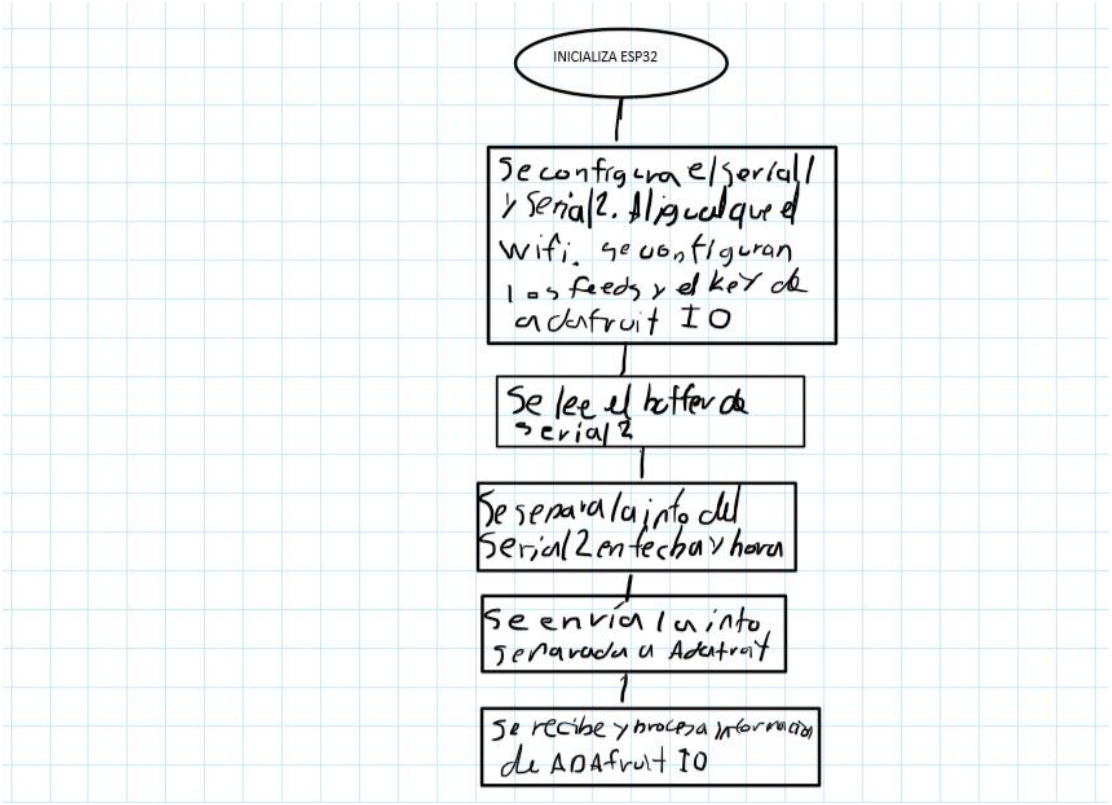
Esp32

- USART de RX y TX con el PIC
- USART con la compu para monitoreo de datos
- Wifi de recepción y envío de datos (cada 3 seg)

En cada fin de hacer 3 Feeds: uno para la hora, uno para la fecha y otro para las luces piloto

Diagrama de flujo





Código del PIC16F887:

```
//******************************************************************************
/*
 * File:   main.c
 * Author: Carlos Cuellar
 *
 * Fecha:
 */
//******************************************************************************
// Importación de Librerías
//******************************************************************************

#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ 4000000

#include "Osc.h"
#include "I2C.h"
//#include "ADCLIB.h"
//#include "LCD8BIT.h"
#include "usart.h"
#include "ASCII_NUM.h"
//#include "SPI.h"

//******************************************************************************
// Palabra de configuración
//******************************************************************************
// 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 digital input, MCLR internally tied to VDD)
#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)



//*****************************************************************************
// Variables
//*****************************************************************************
uint8_t var = 0;
uint8_t seg = 0;
uint8_t segD = 0;
uint8_t segU = 0;
uint8_t min = 0;
uint8_t minD = 0;
uint8_t minU = 0;
uint8_t hora = 0;
uint8_t horaD = 0;
uint8_t horaU = 0;
uint8_t dia = 0;
uint8_t diaD = 0;
uint8_t diaU = 0;
uint8_t mes = 0;
uint8_t mesD = 0;
uint8_t mesU = 0;
uint8_t ano = 0;
uint8_t anoD = 0;
uint8_t anoU = 0;
uint8_t basura = 0;
uint8_t DecenasH = 0;
uint8_t toggleTX = 0;
uint8_t dato_esp = 0;
//*****************************************************************************
// Prototipos de funciones
//*****************************************************************************
void Setup(void);
void escribir_tiempo(void);
void recibir_tiempo(void);
void convertirBCD(void);
```

```
void envio(void);

//*****************************************************************************
// Vector de Interrupción
//*****************************************************************************

void __interrupt() ISR(void) {
      if(dato_esp == 1){
      if(PIR1bits.TXIF == 1){
           PIE1bits.TXIE = 0;
           toggleTX++;
           envio();
    }
     }
    if (INTCONbits.TMR0IF == 1){
           INTCONbits.TMR0IF = 0;
           TMR0 = 100;
           var++;
           if(var == 2){
                 PIE1bits.TXIE=1;
                 var = 0;
           }
    }
    if (PIR1bits.RCIF == 1){
           dato_esp = RCREG;
    }
}

//*****************************************************************************
// Main Loop
//*****************************************************************************
void main(void) {
      initOsc(20);
      Setup();
      USARTconf();
      //escribir_tiempo();
      while(1){

              recibir_tiempo();
           __delay_ms(200);
           convertirBCD();

           if (dato_esp == 1){
                 PORTBbits.RB7 = 1;
                 PIE1bits.TXIE = 1;
           }
           if (dato_esp == 0){
                 PORTBbits.RB7 = 0;
                 PIE1bits.TXIE = 0;
           }
           if(dato_esp == 2){
                PORTBbits.RB0 = 0;
                PORTBbits.RB1 = 0;
           }
           if(dato_esp == 3){
                PORTBbits.RB0 = 1;
                PORTBbits.RB1 = 0;
```

```
            }
            if(dato_esp == 4){
                PORTBbits.RB0 = 0;
                PORTBbits.RB1 = 1;
            }
            if(dato_esp == 5){
                PORTBbits.RB0 = 1;
                PORTBbits.RB1 = 1;
            }
        }
}
//*****************************************************************************
// Vector de Interrupción
//*****************************************************************************
void Setup(void) {
    //CONFIG I&0
    PORTA = 0; //POT
    PORTB = 0;
    TRISB = 0;
    TRISC = 0;
    ANSEL = 0;
    ANSELH = 0;

    INTCONbits.TMR0IF = 0;
    OPTION_REG = 0b11010111;
    INTCONbits.GIE = 1;
    INTCONbits.PEIE = 1;
    INTCONbits.T0IE = 1;
    TMR0 = 250;
    PIE1bits.TXIE = 1;
    PIE1bits.RCIE = 1;

    I2C_Master_Init(100000);

}
//*****************************************************************************
// ESC FECHA
//*****************************************************************************
void escribir_tiempo(void){
    I2C_Master_Start();//Inicializo
    I2C_Master_Write(0xD0);//Escribo D0, es la direccion par meter datos al rtc
    I2C_Master_Write(0);//Escribo un 00, cursor
    I2C_Master_Write(0b00000000);//ESCRIBO SEGUNDOS
    I2C_Master_Write(0x35);//ESCRIBO MIN
    I2C_Master_Write(0x01);//ESCRIBO HORAS
    I2C_Master_Write(1);//1 PARA IGNORAR DIA
    I2C_Master_Write(0x1);//Meto día
    I2C_Master_Write(0x03);//Meto mes
    I2C_Master_Write(0x21);//Meto año
    I2C_Master_Stop();//finalizo comunicacion

}
//*****************************************************************************
// Leer Tiempo
//*****************************************************************************
void recibir_tiempo(void){
    I2C_Master_Start();//Inicializo
```

```
	I2C_Master_Write(0xD0);//Escribo D0, es la direccion par meter datos al rtc
	I2C_Master_Write(0);//Escribo un 00, cursor

	I2C_Master_Start();//Inicializo
	I2C_Master_Write(0xD1);//Escribo D1 para decir que voy a leer
	seg = I2C_Master_Read(1);
	min = I2C_Master_Read(1);
	hora = I2C_Master_Read(1);
	basura = I2C_Master_Read(1);
	dia = I2C_Master_Read(1);
	mes = I2C_Master_Read(1);
	ano = I2C_Master_Read(0);
	I2C_Master_Stop();//finalizo comunicacion

}
//*****************************************************************************
// Convertir en ASCII
//*****************************************************************************
void convertirBCD(void){
	segU = num_ascii(seg & 0b00001111);
	segD = num_ascii((seg & 0b11110000)>>4);
	minU = num_ascii(min & 0b00001111);
	minD = num_ascii((min & 0b11110000)>>4);
	horaU = num_ascii((hora & 0b00001111));
	horaD = num_ascii((hora & 0b00110000)>>4);
	diaU = num_ascii(dia & 0b00001111);
	diaD = num_ascii((dia & 0b11110000)>>4);
	mesU = num_ascii(mes & 0b00001111);
	mesD = num_ascii((mes & 0b11110000)>>4);
	anoU = num_ascii(ano & 0b00001111);
	anoD = num_ascii((ano & 0b11110000)>>4);
}
//*****************************************************************************
// Envio
//*****************************************************************************
void envio (void){
	switch(toggleTX){
		case 1:
			TXREG = diaD;
			break;
		case 2:
			TXREG = diaU;
			break;
		case 3:
			TXREG = 47;
			break;
		case 4:
			TXREG = mesD;
			break;
		case 5:
			TXREG = mesU;
			break;
		case 6:
			TXREG = 47;
			break;
		case 7:
			TXREG = anoD;
```

```
                break;
            case 8:
                TXREG = anoU;
                break;
            case 9:
                TXREG = 0x20;
                break;
            case 10:
                TXREG = horaD;
                break;
            case 11:
                TXREG = horaU;
                break;
            case 12:
                TXREG = 58;
                break;
            case 13:
                TXREG = minD;
                break;
            case 14:
                TXREG = minU;
                break;
            case 15:
                TXREG = 58;
                break;
            case 16:
                TXREG = segD;
                break;
            case 17:
                TXREG = segU;
                break;
            case 18:
                TXREG = 10;
                toggleTX = 0;
                break;


        }
}
```

Código del ESP32

```
   int cont = 0;
   int cont2 = 0;
   int sw = 0;
   String lectura;
   char tiempo[17];
   char tiempo2[17];
   String fecha;
   String basura;
   String hora;
   String fecha2;
   char sep [] = " ";
   char *ptr;
   char *ptr2;
   int bandLED = 0;
   #include "config.h"

   AdafruitIO_Feed *serialm = io.feed("serialm");
   AdafruitIO_Feed *serialh = io.feed("serialh");
   AdafruitIO_Feed *sld = io.feed("sld");

void setup(){
   Serial.begin(9600); //Inicio la comunicación serial con la compu, este seria el Serial 0 de mi ESP
                       //El único argumento de la función es el baudrate que quiero usar
   Serial2.begin(9600, SERIAL_8N1, 16, 17); //Estoy activando el pin 16 y 17 para usar su función serial
                                            //Los argumentos son baudrate, protocolo y
pines.
  io.connect();
   sld->onMessage(handleMessage);
   // wait for a connection
   while(io.status() < AIO_CONNECTED) {
      Serial.print(".");
      delay(500);
}
sld->get();
}

void loop(){
   io.run();
   Serial2.write(1);
   if(Serial2.available()>0){  //Aquí digo que cuando el buffer del puerto serial tenga almacenada
alguna cantidad de bytes
                               //Se procede a hacer la lectura del puerto
      /*Serial.print(char(Serial2.read())); //Se leen los bytes del puerto serial en forma de char, luego
se imprimen en el monitor

      tiempo[cont]= Serial2.read();
      cont++;*/


      Serial2.readBytesUntil(10, tiempo, 17);
  /*fecha.concat(tiempo[0]);
  fecha.concat(tiempo[1]);
  fecha.concat(tiempo[2]);
  fecha.concat(tiempo[3]);
  fecha.concat(tiempo[4]);
  fecha.concat(tiempo[5]);
```

```
  fecha.concat(tiempo[6]);
  fecha.concat(tiempo[7]);*/

        Serial2.write(0);



    }


    //sscanf(tiempo, "%s %s", basura, hora);
    ptr = strtok(tiempo, sep);
    fecha2 = ptr;
    ptr2 = strtok(NULL, sep);
    hora = ptr2;
    Serial.println(fecha2);
    Serial.println(hora);
    Serial.println(bandLED);
    Serial2.write(bandLED);

if (hora != NULL){
        serialm->save(hora);
}
if (fecha2 != NULL){
        serialh->save(fecha2);
}
        delay(4000);


}
void handleMessage(AdafruitIO_Data *data) {

    sw = data->toInt();
    switch (sw){
        case 0:
        bandLED = 2;
        break;
        case 1:
        bandLED = 3;
        break;
        case 2:
        bandLED = 4;
        break;
        case 3:
        bandLED = 5;
        break;
    }

}
```