

Registrador de vibraciones v1.0

Manual de desarrollo



Contenido

1.	Programación del módulo Wi-Fi.....	3
1.1	Requerimientos previos.....	3
1.2	Creación de la interfaz web.....	6
1.3	Programación del módulo	7
2.	Configuración del lector de tarjetas SD	10
3.	Programación del DSC.....	12
3.1	Funciones del DSC.....	12
3.2	Mapa de conexión con otros elementos del dispositivo	13
3.3	Estructura del código fuente.....	13

1. Programación del módulo Wi-Fi

El módulo ESP8266 es el elemento encargado de dar conectividad web al dispositivo. Originalmente viene precargado con un firmware que lo hace funcionar mediante comandos AT lo cual resulta conveniente si se desea configurara manualmente, pero dificulta su uso si se desea conectarlo a un microcontrolador.

Para facilitar el desarrollo de este proyecto el firmware del módulo ESP fue reescrito empleando el IDE de Arduino, esto permitió conectarlo a los servidores de la empresa Adafruit para crear un dispositivo de internet de las cosas capaz de ser activado en forma remota a través de una interfaz web.

1.1 Requerimientos previos

Materiales empleados

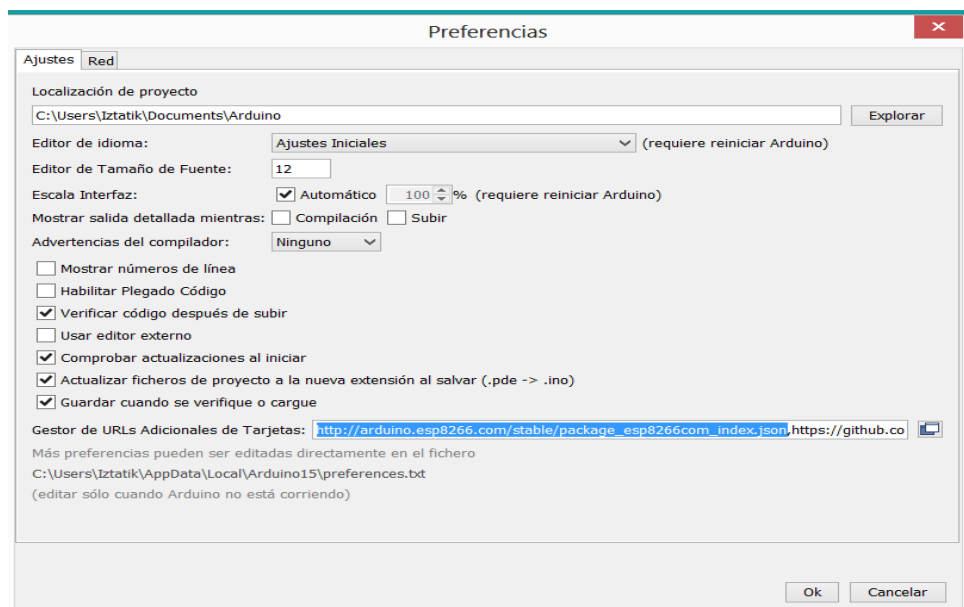
- Esp8266-01
- Interfaz USB-serial FTDI232
- Arduino IDE 1.6.9

Instalación del Plug-in ESP8266 en el Arduino IDE

Instalar este plug-in en el IDE de Arduino permitirá programar el módulo wi-fi como una tarjeta arduino. A continuación se muestran los pasos para lograr esta tarea.

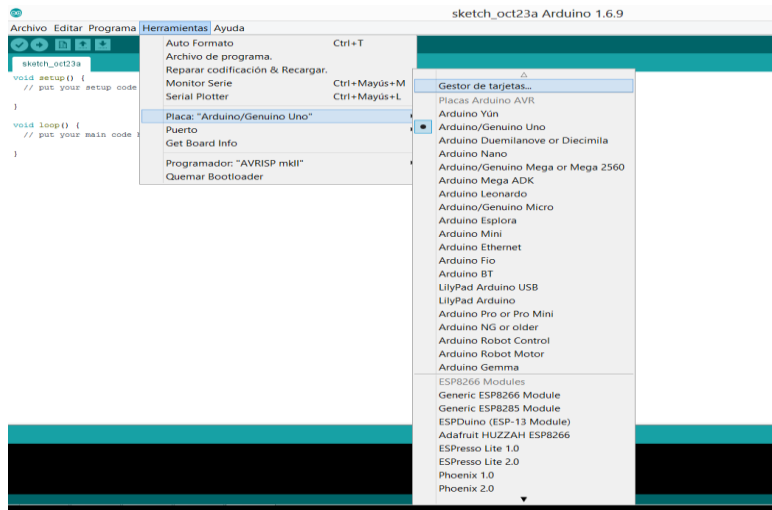
- a) Abrir el IDE de arduino e ir a Archivo/Preferencias. En el recuadro de Gestor de URL's adicionales de Tarjetas escribir la siguiente liga:

http://arduino.esp8266.com/package_esp8266com_index.json

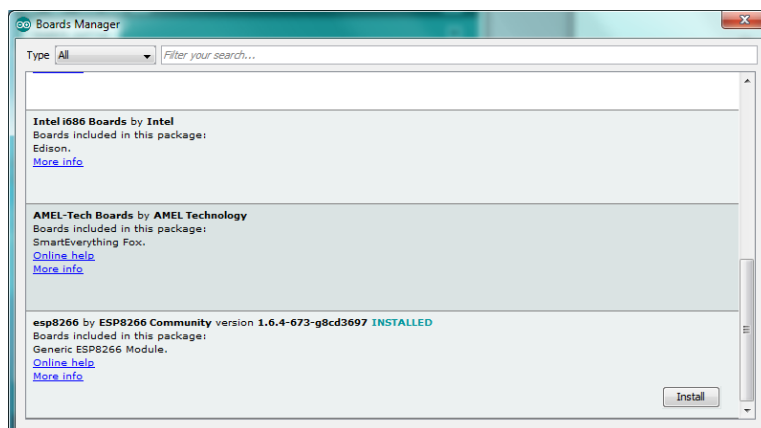


Registrador de vibraciones v1.0

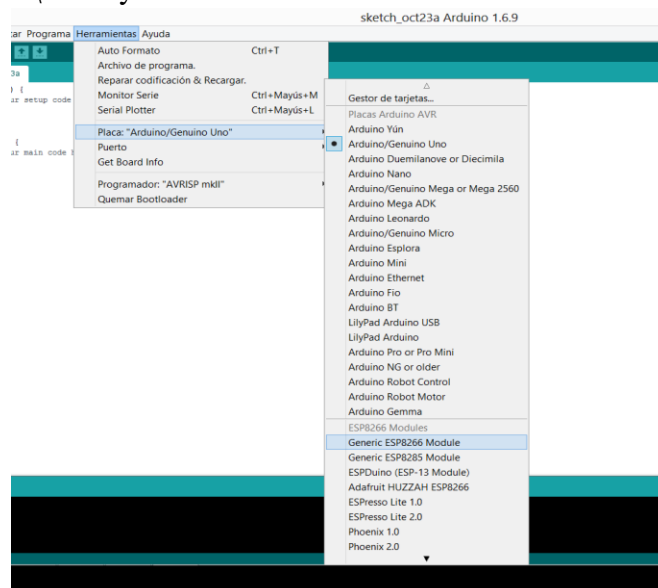
b) Dar click en OK e ir a \Herramientas\Placa\Gestor de tarjetas.



c) Buscar la opción de ESP8266 Community versión e instalarla.

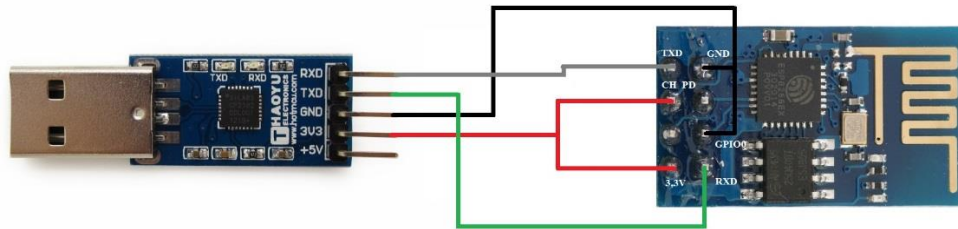


d) Ir a \Herramientas\Placa y seleccionar ESP8266 Generic Module



Registrador de vibraciones v1.0

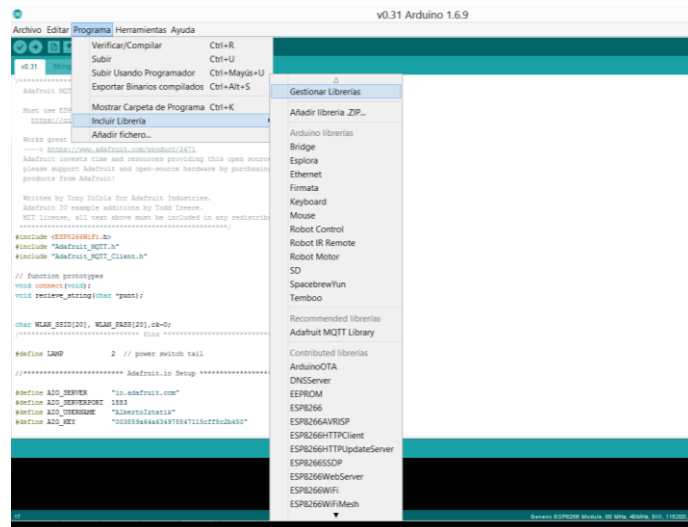
e) Conectar la interfaz FTDI al módulo Wi-fi de la siguiente manera:



Instalación de la librería MQTT de Adafruit en el Arduino IDE

Esta librería permite la conexión del módulo Wi-fi a los servidores de la empresa Adafruit.

1) Diríjase al menú de Programa/Incluir Librería/ Gestor de Librerías



2) En el filtro de búsqueda escriba Adafruit. Busque, seleccione y de click en instalar a la opción de Adafruit MQTT Library.

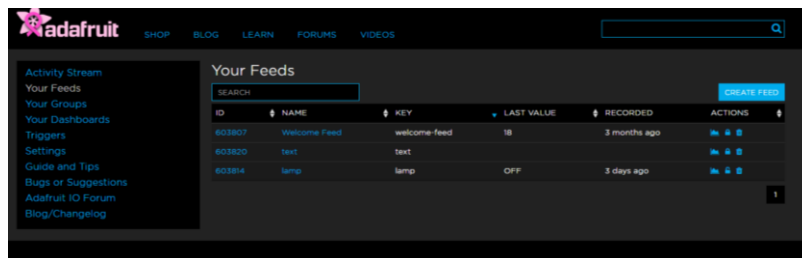


1.2 Creación de la interfaz web

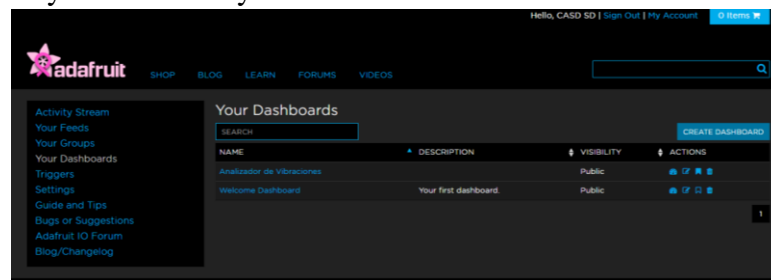
Para la creación de la interfaz web es necesario tener un perfil o crear uno en la página oficial de Adafruit: https://accounts.adafruit.com/users/sign_in

Una vez creada la cuenta deben seguirse los siguientes pasos:

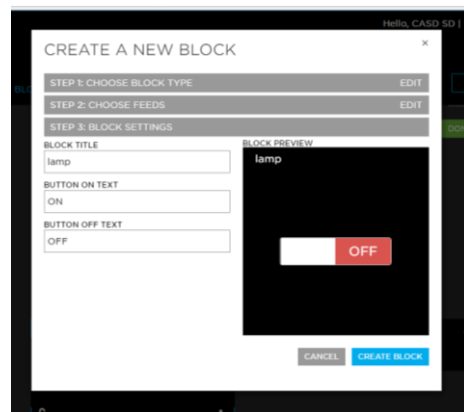
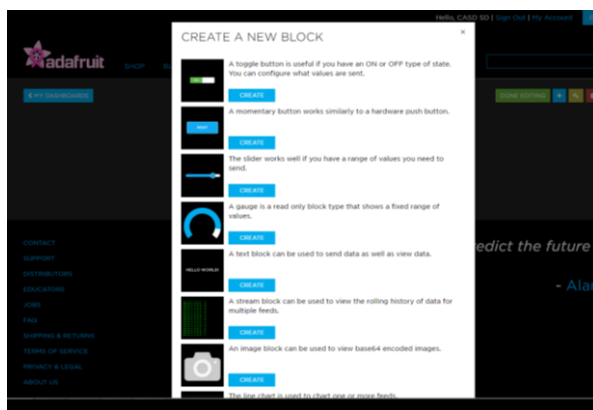
- 1) Ingresar a la página <https://io.adafruit.com/> e iniciar sesión. Ir a la sección MyFeeds y crear un Feed con el nombre “lamp”. Se ha escogido este nombre con la finalidad de coincidir con el nombre planteado en el ejemplo base y la programación explicada en la sección 1.3.



- 2) Ir a la sección My dashboards y seleccionar Create Dashboard.



- 3) Seleccionar las opciones edit dashboard y créate block. Crear un elemento tipo botón y un elemento tipo cuadro de texto, ambos bajo el Group/Feed de “lamp”. Debe cuidarse que el campo de button on text sea llenado con la palabra “ON” y el button off text sea llenado con la palabra “OFF”



- 4) El resultado será una interfaz como la que se muestra a continuación:



Para mayor información seguir la documentación disponible en la siguiente liga:

<https://learn.adafruit.com/adafruit-io-basics-feeds/overview>

1.3 Programación del módulo

Para la programación del módulo se tomó como base la idea fundamental del envío de una señal para el inicio o término de la grabación de datos del dispositivo. Para ello se partió de un ejemplo de código de la empresa Adafruit, el cual enciende y apaga una lámpara de forma remota a través de una interfaz web. El ejemplo puede ser descargado de la siguiente liga:

https://github.com/adafruit/adafruit-io-basics/tree/master/esp8266/digital_out

Teniendo este ejemplo como base se hicieron las siguientes modificaciones:

- 1) El ejemplo original contempla que se programe el nombre y clave de la red wlan de forma anticipada, no obstante esto se traduce en la necesidad de reprogramar el módulo cada vez que se quiera cambiar de red. Para evitar esto, se declararon estos valores como variables las cuales son definidas mediante una lectura serial cada vez que se enciende el módulo.

```
*****
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

// function prototypes
void connect(void);
void recieve_string(char *punt);

char WLAN_SSID[20], WLAN_PASS[20], ck=0;
/***** Pins *****/

#define LAMP 2 // power switch tail
```

Definicion de la función para la recepción de datos

Definicion de las variables

- 2) Se creó una función encargada de recibir los datos de las variables WLAN_SSID y WLAN_PASS cada vez que se enciende el módulo. Debe notarse que el código fuente de esta función se encuentra en una pestaña aparte de la función principal.

Registrador de vibraciones v1.0

```
v0.31 String_recieve
void recieve_string(char *punt){

    char buffa;
    do{while (!Serial.available());}while(Serial.read() !='-').

    while (1){
        while (!Serial.available());
        buffa=Serial.read();
        if (buffa == '-') {*punt=0x00;return;}
        *punt=buffa;
        punt++;
    }
    return;
}
```

- 3) En la sección de definiciones se introdujeron el nombre de usuario y las claves AIO generadas al crear la interfaz web.

```
char WLAN_SSID[20], WLAN_PASS[20], ck=0;
/***** Pins *****/

#define LAMP          2 // power switch tail

/***** Adafruit.io Setup *****/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "AlbertoIztatik"
#define AIO_KEY          "003859a64a634978847115cff8c2b450"
```

Nombre de usuario (puntero a AIO_USERNAME)

Llave AIO (puntero a AIO_KEY)

- 4) La función void setup fue modificada para iniciar la conexión wifi hasta asegurar la recepción vía serial y la definición de las variables del nombre y clave de la red wlan. También se modificó la velocidad de transmisión serial a 57600 baudios. En la función Serial.begin.

```
void setup() {

    // set power switch tail pin as an output
    pinMode(LAMP, OUTPUT);
    digitalWrite(LAMP, 0);

    Serial.begin(57600);
    delay(10);
    recieve_string(WLAN_SSID);
    recieve_string(WLAN_PASS);
    Serial.println(WLAN_SSID);
    Serial.println(WLAN_PASS);

    WiFi.begin(WLAN_SSID,WLAN_PASS);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(F("."));
    }
    Serial.print(F("+"));
    // listen for events on the lamp feed
    mqtt.subscribe(#lamp);

    // connect to adafruit io
    connect();

}
```

Recepción serial del nombre y clave de la red wlan (puntero a recieve_string(WLAN_SSID);)


Inicio de la comunicación wifi (puntero a WiFi.begin(WLAN_SSID,WLAN_PASS);)

- 5) Se modificó la función void loop ara que al recibir la señal de encendido “ON” de la interfaz web, el módulo ESP envié de forma serial la letra “s” indicando el inicio de

Registrador de vibraciones v1.0

una nueva grabación de datos. De igual manera, al recibir una señal “OFF” el módulo ESP enviará de forma serial la letra “e” indicando la finalización de la grabación de datos. *El DSC conectado al módulo ESP recibirá estos datos por su puerto serial y ejecutará las operaciones mencionadas.*

```
void loop() {  
  
  Adafruit_MQTT_Subscribe *subscription;  
  
  // ping adafruit io a few times to make sure we remain connected  
  if(! mqtt.ping(3)) {  
    // reconnect to adafruit io  
    if(! mqtt.connected())  
      connect();  
  }  
  
  // this is our 'wait for incoming subscription packets' busy subloop  
  while (subscription = mqtt.readSubscription(1000)) {  
  
    // we only care about the lamp events  
    if (subscription == slamp) {  
  
      // convert mqtt ascii payload to int  
      char *value = (char *)lamp.lastread;  
  
      // Apply message to lamp  
      String message = String(value);  
      message.trim();  
      if (message == "ON") {digitalWrite(LAMP, HIGH);Serial.print("s");}  
      if (message == "OFF") {digitalWrite(LAMP, LOW);Serial.print("e");}  
    }  
  }  
}
```



Envío de datos de forma serial

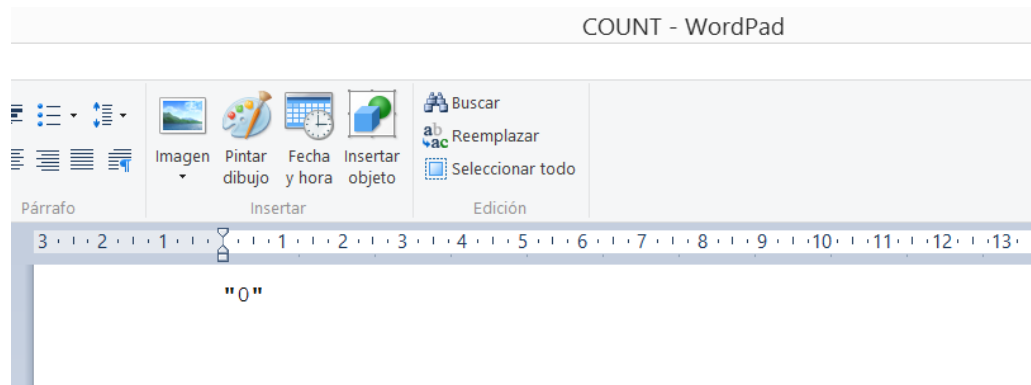
- 6) A lo largo de todo el código se eliminaron las impresiones seriales por el puerto serie que estaban definidas a excepción de las que muestra que el estado de conexión esta en proceso (.) y la que muestra una conexión exitosa (+).

Para mayor información seguir la documentación disponible en la siguiente liga:
<https://learn.adafruit.com/home-automation-in-the-cloud-with-the-esp8266-and-adafruit-io/introduction>

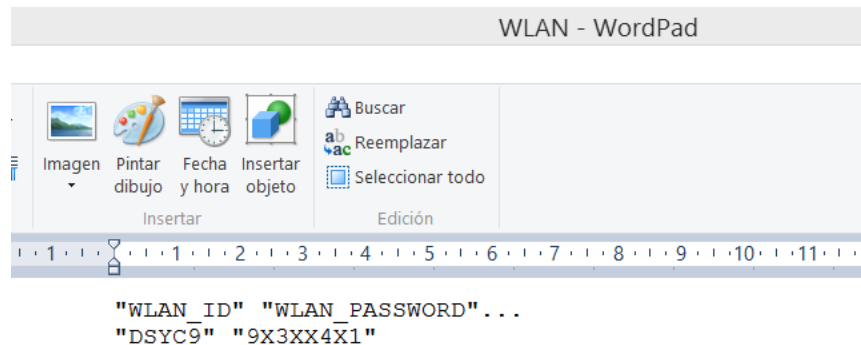
2. Configuración del lector de tarjetas SD

EL lector de tarjetas SD Open Log se configura mediante tres archivos de texto. A continuación se detalla la función de los mismos:

- 1) **Archivo COUNT.-** Lleva un conteo de los archivos de grabación generados, permitiendo que el DSC conozca la cantidad de archivos generados hasta el momento y pueda generar el archivo siguiente en la lista.



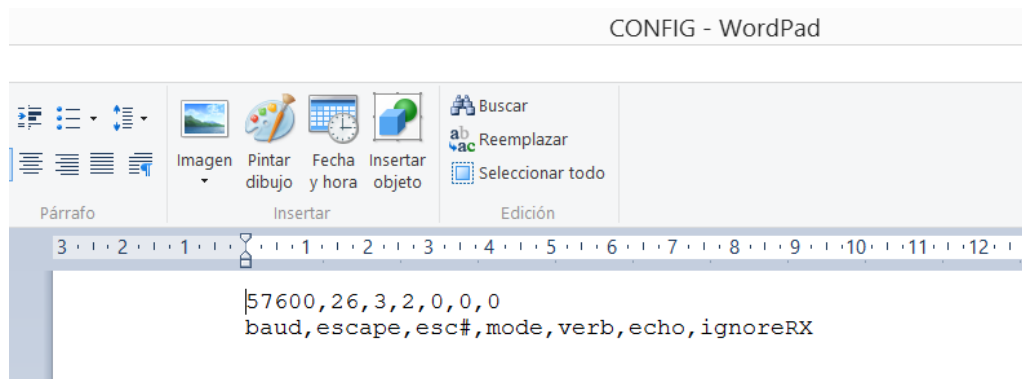
- 2) **Archivo WLAN.-** contiene los datos de la red de área local a la cual se conectará el dispositivo, el DSC leerá este archivo para obtener esa información y posteriormente enviarla al módulo ESP para que este realice la conexión.



- 3) **Archivo CONFIG.-** Contiene los parámetros de configuración básica para la comunicación serial de la tarjeta. Estos son en orden:
 - Baudrate.- Velocidad de la conexión. 57600 para esta aplicación.
 - Escape.- Valor en ASCII del carácter que permite poner al Open Log en modo comando. Ctrl+z para esta aplicación.

Registrador de vibraciones v1.0

- Esc#.- número de veces que debe leerse el carácter de escape para poner al Open Log en modo comando. 3 para esta aplicación.
- Mode.- Modo con el que inicia el Open log al encenderse en cada ocasión. Modo comando para esta aplicación (valor 0).
- Verb.- Responde con un carácter cada que se produce un error al leerse un comando no reconocido. Apagado para esta aplicación (valor 0).
- Echo.- Responde con un carácter cada que se lee un dato por el puerto serial. Apagado para esta aplicación (valor 0).
- Ignore RX.- Ignora la línea serial de RX por un periodo cada que se enciende el Open Log. Apagado para esta aplicación (valor 0).



3. Programación del DSC

Herramientas empleadas:

- MPLABX IDE V3.10
- C16 Compiler V1.25
- Programador Pickit3
- dsPic30f3014

3.1 Funciones del DSC

El DSC (controlador digital de señales) es el elemento de control principal del dispositivo. Como tal, es el encargado de interactuar con los otros elementos y gestionar sus tareas. A continuación se enumeran las funciones del DSC respecto a todos los elementos:

Lector de tarjetas SD Open Log:

- **Creación y finalización de un nuevo archivo de grabación.-** Al presionar el botón físico o virtual (interfaz web) el dispositivo leerá el contenido del fichero COUNT de la SD, incrementará en un valor este conteo, describirá dicho fichero con el valor incrementado y creará un nuevo fichero con el nombre del valor del conteo incrementado donde grabará los datos obtenidos del acelerómetro.
- **Lectura del archivo de configuración de la red wlan.-** Cada vez que se enciende el dispositivo, el DSC leerá vía serial los datos necesarios para la conexión a una red de área local del fichero WLAN en la tarjeta SD, posteriormente enviará estos datos al módulo wifi para que este pueda la conexión.

Módulo wifi ESP8266

- **Envío de los datos de la red wlan.-** El DSC lee estos datos del fichero WLAN de la tarjeta SD y posteriormente los envía vía serial al módulo ESP para que este entable la conexión.
- **Recepción de los datos de activación.-** Cada vez que se presiona el botón de la interfaz web, el DSC recibe vía serial las instrucciones para la creación o finalización de un archivo de grabación.

Reloj en tiempo Real

- **Lectura del RTC.-** El DSC lee el valor del reloj en tiempo real cada que se inicia o finaliza un archivo de grabación para escribir la hora en la que este fue iniciado y terminado.

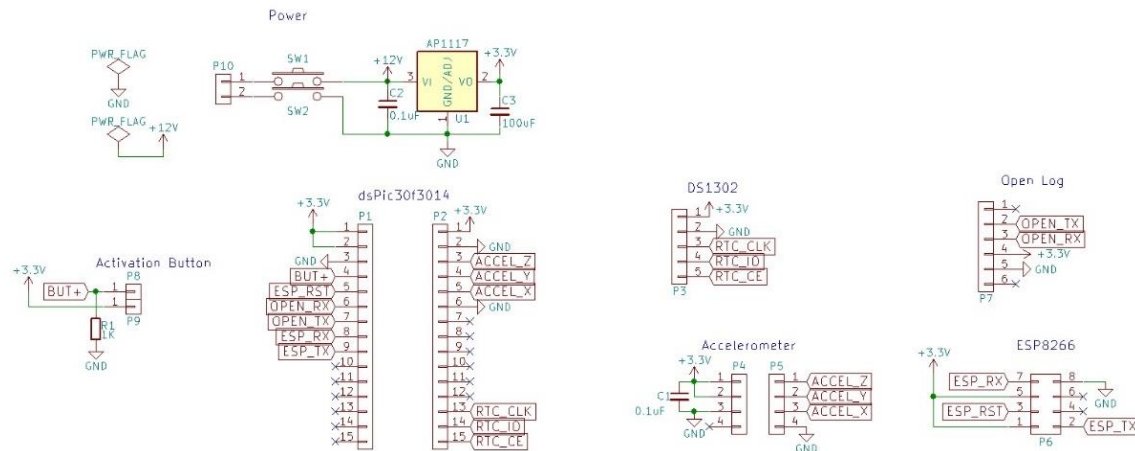
Botón del dispositivo.- El DSC detectará que se ha presionado dicho botón e iniciará o detendrá la grabación de datos.

Registrador de vibraciones v1.0

Gestión de energía de la batería.- El DSC determinará el nivel de carga de las pilas y encenderá el led rojo cuando estas estén totalmente descargadas.

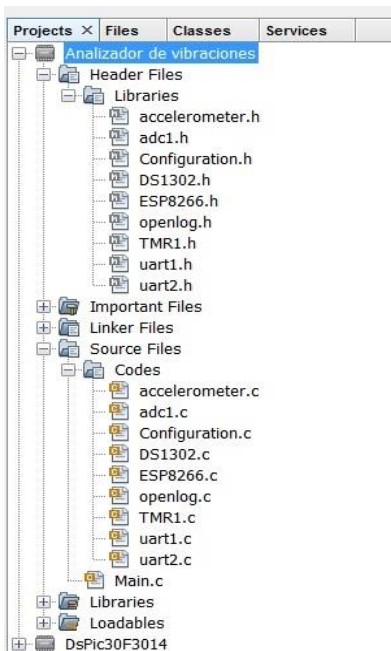
3.2 Mapa de conexión con otros elementos del dispositivo

El DSC y en general todos los elementos del dispositivo se encuentran conectados entre sí como muestra el siguiente diagrama esquemático:



3.3 Estructura del código fuente

El código fuente ha sido organizado en distintos ficheros para hacerlo lo más modular posible, dichos ficheros contienen un conjunto de funciones que al ser invocadas realizan funciones específicas del sistema. A continuación se detalla cada uno de estos ficheros y las funciones que contiene:



Archivos de cabecera:

- a) **accelerometer.h.-** definición de variables y funciones del archivo accelerometer.c.
- b) **adc.h.-** definición de variables y funciones del archivo adc.c.
- c) **Configuration.h.-** definición de la frecuencia de reloj del sistema, inclusión de las librerías a emplear.
- d) **DS1302.h.-** definición de variables y funciones del archivo DS1302.c
- e) **ESP8266.h.-** definición de variables y funciones del archivo ESP8266.c.
- f) **openlog.h.-** definición de variables y funciones del archivo openlog.c
- g) **TMR1.h.-** Definición de la frecuencia de las interrupciones para las lecturas del ADC.
- h) **uart1.h.-** Frecuencia y funciones para la comunicación serie con el Open Log
- i) **uart2.h.-** Frecuencia y funciones para la comunicación serie con el ESP.

Archivos de código fuente:

- a) **accelerometer.c:**
 - **ACCEL_get ()**.- Regresa la lectura de cualquier eje del acelerómetro convertida a un valor de voltaje.
- b) **adc.c:**
 - **ADC1_initialize()**.- configura el ADC para su funcionamiento
 - **ADC1_getdata ()**.- realiza “manualmente” el proceso de muestreo y conversión de un dato del ADC.
 - **BATT_get ()**.-Regresa la lectura del voltaje de la batería convertido a un valor de voltaje.
- c) **Configuration.c.-** Contiene la definición de los fuses del DSC.
 - **SYSTEM_initialize ()**.- invoca todas las funciones de inicialización de cada función y gestiona el tiempo de retardo para que se ejecuten.
- d) **DS1302.h:**
 - **DS1302_initialize ()**.- Inicializa el DS1302.
 - **DS1302_write ()**.- Hace una lectura en los registros del DS1302.
 - **DS1302_read ()**.- Realiza una lectura de los registros del DS1302.

- **DS1302_bcd2bin ()**.- Convierte un dato de BCD a binario.
- **DS1302_rgtclean ()**.- Determina el registro del RTC con el que se trabajará.

e) ESP8266.c:

- **initilize_ESP8266 ()**.- Inicializa el ESP poniendo su pin de reset a 0.
- **wlan_ESP8266 ()**.- Envía el ESP de forma serial los datos necesarios para la conexión a la red wlan.

f) openlog.c:

- **OPENLOG_initilize ()**.- Lee los archivos WLAN y COUNT de la SD e invoca la función wlan_ESP8266 ()
- **OPENLOG_get_string ()**.- Recibe una cadena de caracteres del puerto serial.
- **OPENLOG_parser ()**.- Función que ejecuta un algoritmo de búsqueda en el string recibido para extraer la información útil.
- **OPENLOG_completewrite ()**.- Escribe en una sola invocación las lecturas obtenidas del acelerómetro.
- **OPENLOG_comandmode ()**.- Hace que el Open Log entre en modo comando.
- **OPENLOG_newfile ()**.- crea un nuevo fichero de grabación en el Open Log.
- **OPENLOG_printtime ()**.- Obtiene la hora del RTC y la imprime en un archivo en uso del Open Log.

g) TMR1.c:

- **TMR1_initialize ()**.- Configura el timer 1 para realizar interrupciones cierto periodo.

h) uart1.c:

- **UART1_initialize ()**.- Configura la comunicación serial.
- **UART1_send_string ()**.- Envía una cadena de caracteres por el puerto.

i) uart2.c:

- **UART2_initialize ()**.- Configura la comunicación serial.
- **UART2_send_string ()**.- Envía una cadena de caracteres por el puerto.
- **UART2_get_string ()**.- Recibe una cadena de caracteres por el puerto serial.
- **_U2RXInterrupt ()**.- Rutina de interrupción que se genera al recibir un dato por el puerto.

Función principal main

La función main ejecuta el siguiente algoritmo:

1. Invoca la función SYSTEM_initialize () para inicializar todos los elementos.
2. Define las variables de la función.
3. Define el pin del botón como entrada y los pines de los leds como salidas.
4. Enciende el led verde indicando que el dispositivo está listo para iniciar una grabación, apaga el led rojo de la batería.
5. En el bucle infinito espera a que se produzca algún evento que inicie la grabación (presionar el botón o recepción serial).
6. Si se inicia la grabación se espera un tiempo para evitar rebotes del botón, se enciende el led amarillo y se genera un nuevo archivo.
7. Repetidamente se leen los datos de los ejes del acelerómetro y la batería, los primeros se imprimen en el archivo creado y el segundo se compara con un valor umbral que prende o apaga el led rojo.
8. Si se presiona de nuevo algún botón se apaga el led amarillo y se regresa al estado descrito en el inciso 5.

```
int main(void) {
    SYSTEM_initialize();
    unsigned int accel_dataX, accel_dataY, accel_dataZ, batt_stat ;
    unsigned int counter=0;

    TRISFbits.TRISF0=1;
    TRISCbits.TRISC13=0;
    TRISCbits.TRISC14=0;
    TRISDbits.TRISD8 =0;

    LED_on=1;
    LED_batt = 0;

    while (1){

        if(button&&!function_flag){function_flag =1;file_flag=1;__delay_ms(500);}
        if(file_flag==1){file_flag=0;OPENLOG_newfile();}

        while (TMR1_FLAG && function_flag){

            LED_active =1;
            accel_dataX = ACCEL_get(x);
            accel_dataY = ACCEL_get(y);
            accel_dataZ = ACCEL_get(z);
            batt_stat = BATT_get();
            OPENLOG_completewrite(counter++,accel_dataX,accel_dataY,accel_dataZ);
            if (batt_stat<=2100) LED_batt=1; // aprox 6.10v de las pilas
            else LED_batt=0;
            TMR1_FLAG=0;
        }

        if(PORTFbits.RF0&&function_flag){function_flag =0;file_flag=2;__delay_ms(1000);}
        if(file_flag==2) {OPENLOG_printtime();file_flag=0;LED_active =0;counter=0;}

    }

    return 0;
}
```