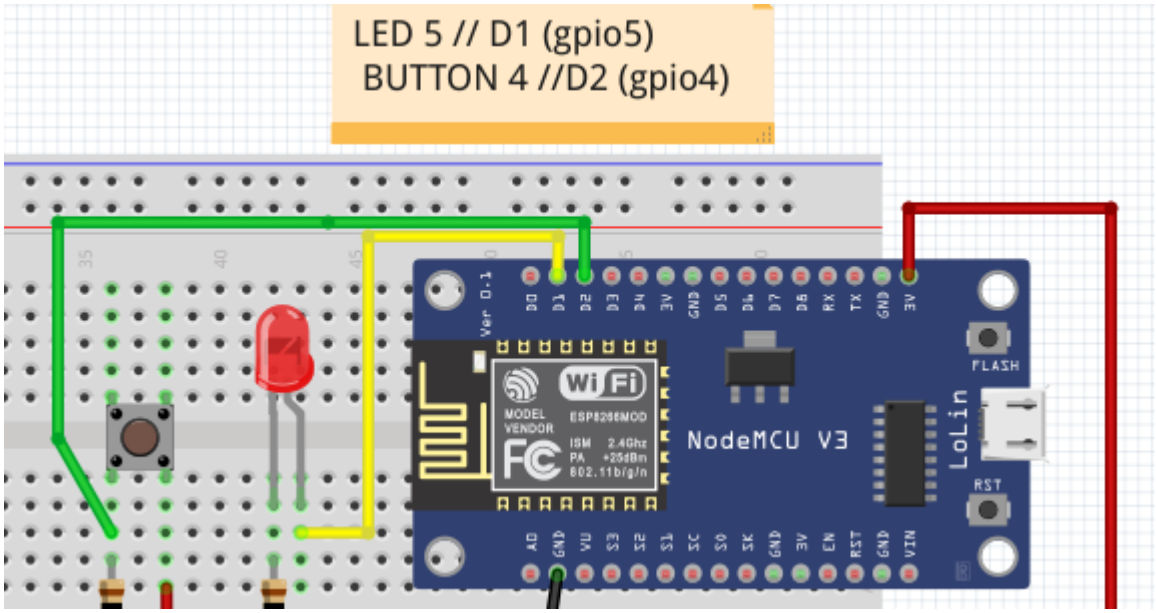
NodeMCU: comunicación con Firebase. La estación escribe datos (VII-A).

agrportfolioeducativo.blogspot.com/2019/07/nodemcu-comunicacion-con-firebase-la.html



Montaje

No cambiamos el Hardware de la estación ni de la central. Exactamente lo mismo.

El programa

```
/*
 * Created by Aurelio Gallardo Rodríguez
 * Based on: K. Suwatchai (Mobizt)
 *
 * Email: aurelio@seritium.es
 *
 * ESTACION. Escritura
 *
 * Julio - 2019
 *
*/

//FirebaseESP8266.h must be included before ESP8266WiFi.h
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
```

```
#define FIREBASE_HOST "[loquesea].firebaseio.com"
#define FIREBASE_AUTH "klf4fdgHjkdfgk4lldfglk4pmvnsflksdfg"
#define WIFI_SSID "miSSID"
#define WIFI_PASSWORD "miCONTRASEÑA"

//Define FirebaseESP8266 data object
FirebaseData bd;

String path ="/Estaciones";// path a FireBase
String estacion ="/A";// Estacion que voy a controlar

int i=1;// contador general

#define LED 5// D1(gpio5)
#define BUTTON 4//D2(gpio4)

int estado=0;
int estadoAnterior=0;
int estadoLuz=0;

void setup()
{

  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT);



// conectando a la WIFI
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Conectando a la WiFi");
while(WiFi.status()!= WL_CONNECTED)
{
    Serial.print(".");
    delay(300);
}
  Serial.println();
  Serial.print("Conectado con IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();



// Conectando a la bd real Time Firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
```

```
//Set database read timeout to 1 minute (max 15 minutes)
  Firebase.setReadTimeout(bd, 1000*60);
//tiny, small, medium, large and unlimited.
//Size and its write timeout e.g. tiny (1s), small (10s), medium (30s) and large (60s).
  Firebase.setwriteSizeLimit(bd, "unlimited");



  wFB(0);// Escribe un cero en la estación al arrancar



}

// Bucle principal
void loop(){

  estado=digitalRead(BUTTON);
  if (estado!=estadoAnterior){
    // el estadoAnterior ahora es el estado
    estadoAnterior=estado;
    // si estado es alto, se pasa de bajo a alto
    if (estado) {
      estadoLuz=!estadoLuz; // El estado de la luz cambia
      // sendDataToGoogleSheets(estacion,estadoLuz); // envío a googleSheet esa
información
      wFB(estadoLuz); // en vez de enviarlo a Google Spreadsheet, se lo envío a Real Time
BD Firebase
      Serial.println("Estado del LED: "

+ String(estadoLuz));//Manda al monitor serie ese estado.
      delay(1);
}
}



  digitalWrite(LED,estadoLuz);
// digitalWrite(LED,rFB()); // Escribe el estado en un LED, PERO leyéndolo de la BD.
}



// Función que escribe un dato en Firebase.
// Además, actualiza el dato si tiene la misma ruta, no genera un error.
void wFB(int dato){//Write Data in FB

  if (Firebase.setInt(bd,path+estacion,dato)) {
     Serial.println("PASSED");
```

```
    Serial.println("PATH: " bd.dataPath());
    Serial.println("TYPE: " bd.dataType());
    Serial.println("ETag: " bd.ETag());
    Serial.print("VALUE: ");
    if (bd.dataType() == "int")
      Serial.println(bd.intData());
    else if (bd.dataType() == "float")
      Serial.println(bd.floatData(), 5);
    else if (bd.dataType() == "double")
      printf("%.9lf\n", bd.doubleData());
    else if (bd.dataType() == "boolean")
      Serial.println(bd.boolData() == 1 ? "true" : "false");
    else if (bd.dataType() == "string")
      Serial.println(bd.stringData());
    else if (bd.dataType() == "json")
      Serial.println(bd.jsonData());
    Serial.println("----------------------------------");
    Serial.println();
  } else {
    Serial.println("FAILED");
    Serial.println("REASON: "

bd.errorReason());
    Serial.println("----------------------------------");
    Serial.println();
}
}



// Función de lectura
int rFB(){// lee el dato de la entrada correspondiente

  if (Firebase.getInt(bd, path

estacion)){

    if (bd.dataType() == "int") {
      Serial.println("Dato leído: "

+ (String) bd.intData());
return bd.intData();
}

                } else {
    Serial.println(bd.errorReason());
  }
}
```

= = = = = = = = = =

1. El código está desarrollado a partir del ejemplo básico de la biblioteca FirebaseESP8266. Además cargamos la biblioteca ESP8266WiFi en la cabecera del programa. Para programas más elaborados quizás es interesante la lectura detallada del README de la biblioteca FirebaseESP8266.
2. Defino cuatro parámetros: FIREBASE_HOST, FIREBASE_AUTH, WIFI_SSID y WIFI_PASSWORD. En el post anterior expliqué dónde conseguir los dos primeros, los que me permiten conectarme a FIREBASE y los otros dos ya los hemos visto con anterioridad.
3. Creamos un objeto de base de datos de la clase FirebaseData: **FirebaseData bd**.
4. Defino después varias variables: **path**, ruta dentro de Firebase a los campos de la base de datos, el nombre de la estación que estoy manejando "**A**", un contador general,las definiciones de los pines y de las distintas variables de estado, que ya vimos en entradas anteriores.
5. SETUP --> inicializamos la conexión monitor serie, definimos los pines como entradas y salidas e iniciamos el proceso de conexión a la red Wifi, muy similar a programas anteriores. Posteriormente conectamos con la base de datos Firebase: Firebase.begin conecta a la base de datos con su token de autentificación, Firebase.reconnectWiFi es una función que viene en el ejemplo, opcional según README, y que supongo que comprobará que estamos conectados a la WiFi. Las siguientes dos instrucciones: **Firebase.setReadTimeout(bd, 1000 * 60)** y **Firebase.setwriteSizeLimit(bd, "unlimited")** no vienen explicadas en el README de la biblioteca FirebaseESP8266. Me imagino que la primera es el tiempo de espera límite cuando se realiza una petición de lectura y la otra función tiene que ver con las peticiones de escritura y también el tiempo de espera, según los comentarios en el ejemplo. El setup acaba llamando a la función de escritura en Firebase con un cero. Lógico, si reinicio la estación, esta debe tener como valor inicial cero.
6. LOOP --> las instrucciones son casi análogas a la de la entrada anterior. Reconocen el cambio de estado del botón (flanco de subida) y cambian el estado del LED. Este estado se envía ahora, no a una hoja de cálculo de Google, sino a la función **wFB(estadoLuz)**, que escribirá el dato en la base de datos Firebase. También ponen el estado en el LED, y, comentado, aparece la forma de hacerlo leyendo el dato desde la base de datos: **rFB()**.
7. **wFB(int dato)** --> esta función escribe un entero en la base de datos. La función que lo hace es **Firebase.setInt(bd,path+estacion,dato)** devuelve **true** si consigue hacerlo. El parámetro bd es la base de datos, path+estacion es la ruta al campo donde se guarda el dato "/Estacion/A" y dato es el dato que hemos pasado a la función. Si efectivamente graba el dato, se imprime información complementaria en el monitor serie: tipo, ruta, Etag, y el valor. Si no es capaz de grabar el dato, lanza un mensaje de error con el método **.errorReason()**.

8. **rFB() -->** esta función, no imprescindible, pero que se añade al programa, lee un dato en la ruta path+estacion (o sea, "/Estacion/A"). Simplemente se comprueba si se produce la lectura con **Firebase.getInt(bd, path+estacion),** y, si existe, se comprueba si es un entero y se devuelve con **return bd.intData().** Asimismo si no existiera, se lanza el método **.errorReason()**.

NOTA: si no existe el campo, la función de escritura lo crea. Si volvemos a escribir el mismo campo con otro valor, se actualiza. Sin embargo, si intentamos leer un campo que no existe, se lanza un error. El método .errorReason() muestra las causas de no haberlo conseguido, pero también reinicia el módulo NodeMCU.