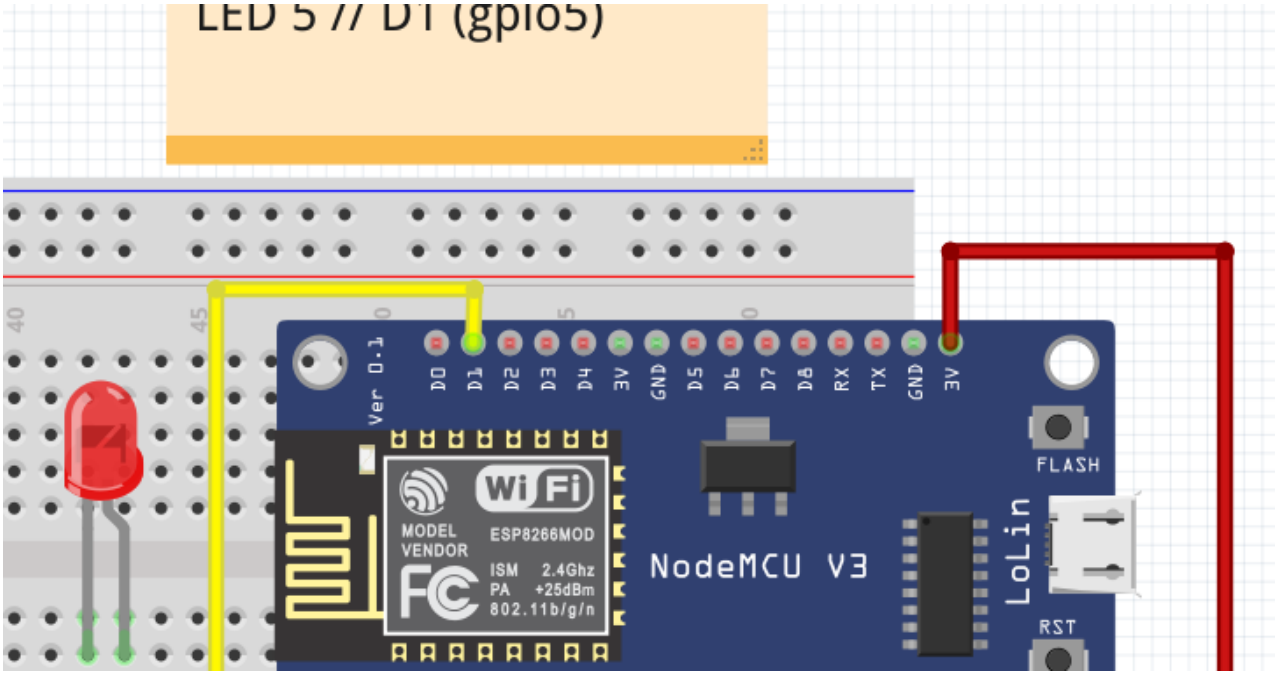
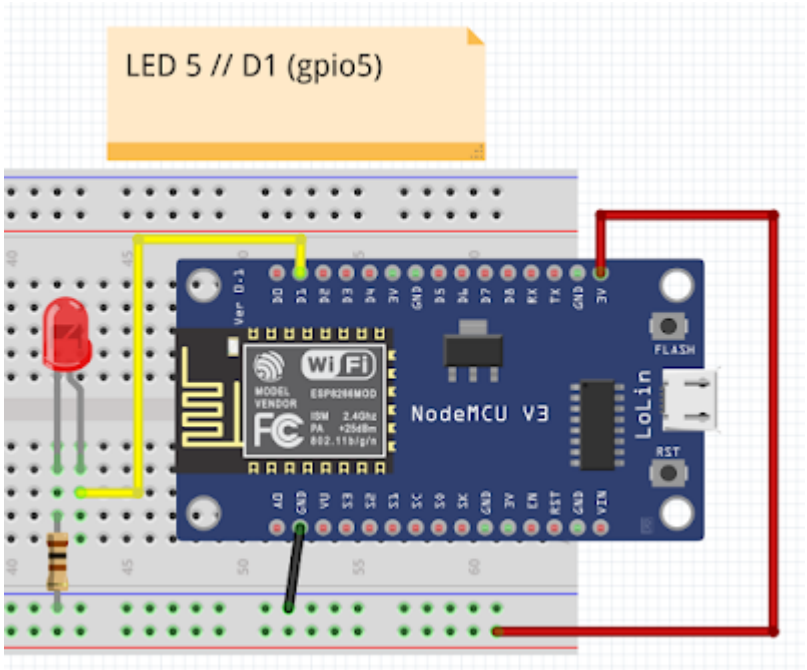
NodeMCU: comunicación con Firebase. La central lee datos (VII-B).

agrportfolioeducativo.blogspot.com/2019/07/nodemcu-comunicacion-con-firebase-la_13.html



Montaje

Pues exactamente igual a la central de antes



El programa

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

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

#define FIREBASE_HOST "[elquesea].firebaseio.com"
#define FIREBASE_AUTH "sdfsgdlkm3485j3dfu992349322njfewrtet" //inventado
#define WIFI_SSID "miSSID"
#define WIFI_PASSWORD "miCONTRASEÑA"

//Define FirebaseESP8266 data object
FirebaseData bd;

String path ="/Estaciones";// path a FireBase
char*estaciones[]={"A","B","C","D","E"};// Estaciones que voy a controlar

int i=1;// contador general

int activo=0;// Alarma

#define LED 5// D1(gpio5)

void setup()
{

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



// 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");



}

// Bucle principal
void loop(){

  activo=0;// reinicializo la variable

    for(i=0;i<=4;i++) {
    delay(1);
    activo=activo+rFB("/"+(String) estaciones[i]);
  }


    digitalWrite(LED,(activo>=1));// activo el LED si es mayor o igual a 1.



}

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



int valor=0;
```

```
  if (Firebase.getInt(bd, pathestacion)) {
    if (bd.dataType() == "int") {
      Serial.println("Dato leído: "+ path+estacion+ " --> "

+(String) bd.intData());
      valor = bd.intData();// retorna el valor
}
}else{
// Si no existe el dato de la estación, salta el error
// Pero el error se muestra en pantalla (bd.errorReason()) Y REINICIALIZA LA UNIDAD,
lo cual no quiero.
    Serial.println("Estoy dando un error... ojo");
// Serial.println(bd.errorReason());
}



return valor;

  }
```

= = = = = = = = =

Explico simplemente qué cambia respecto al programa de la entrada anterior.

1. Defino un array tipo string con las cinco estaciones que puede leer la central: **char *estaciones[]= {"A","B","C","D","E"}**. Evidentemente esto es ampliable y modificable.
2. Defino la variable **activo**. Será cero si no se activa ninguna estación, y uno o más de uno si se activa una o más estaciones.
3. Solamente defino la salida del led de activación **#define LED 5** y **pinMode(LED, OUTPUT).**
4. SETUP --> Las instrucciones de conexión son las mismas, tanto para la conexión WiFi como para la de Firebase.
5. **rFB(String estacion) -**-> La función de lectura es idéntica a la de la entrada anterior. Solo cambia que devuelve cero si no es capaz de encontrar un dato, evitando ejecutar .errorReason(), y puedo pasar como variable la estacion que quiero leer.
6. LOOP --> El programa es sencillo. Empieza siempre con activo=0 y lee desde un bucle todas las estaciones (que deben tener valores 0 o 1). Suma sus valores, de forma que si una está activa, activo valdrá uno y 3 si hay tres activas. Actuará sobre el led encendiéndolo si activo es un número mayor o igual que uno.