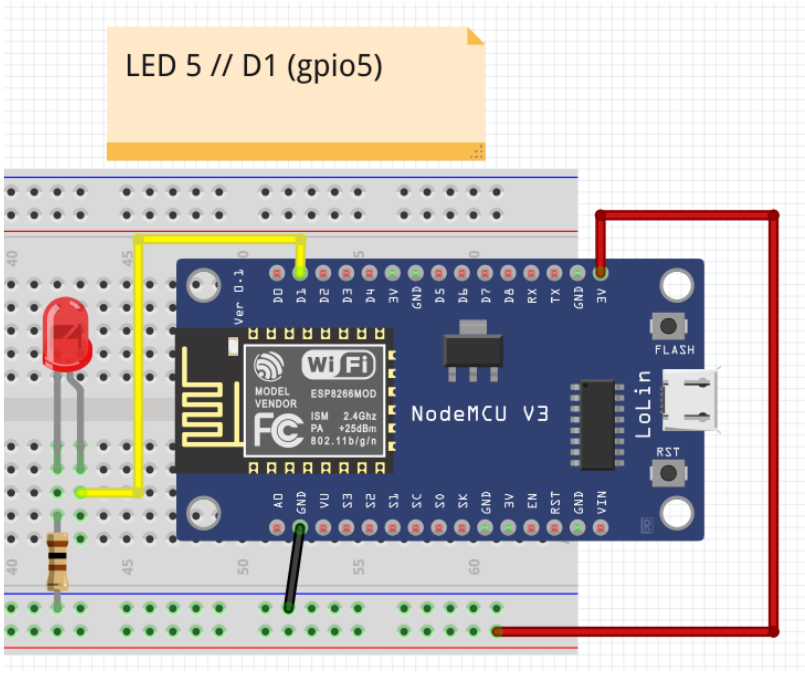
NodeMCU: proyecto "botón del pánico" (VI-C): el NodeMCU como central.

agrportfolioeducativo.blogspot.com/2019/07/nodemcu-proyecto-boton-del-panico-vi-c.html



El esquema.

Simplemente voy a ilustrar el funcionamiento. Con un simple LED conectado al pin D1 será suficiente.

Modificando el script de Google

Pensé en crear un nuevo script para la central. Pero no lo he visto necesario. Una modificación como la que sigue hace el trabajo.

}elseif(estacion.toUpperCase()==="CENTRAL"){// caso del requerimiento de la central

```
var respuesta ={};
var est="";
var valor =0;
var suma =0;
```

```
for(var i=3;i<=7;i++){
      est = sheet.getRange(1,i).getValue();
      respuesta[est]=sheet.getRange(2,i).getValue();
      suma = suma

respuesta[est];
}



   respuesta["alarma"]=suma;// si es mayor o igual a 1, se activa la alarma

      var output = HtmlService.createHtmlOutput('EMPEZAR'JSON.stringify(respuesta)

'TERMINAR');
return output;

         // Logger.log(JSON.stringify(respuesta));

 }
```

De forma simple, lee el contenido de las columnas desde la C a la G en la fila 2 y las asigna a un arreglo por clave - valor. Además crea un valor **alarma** que contendrá la suma de todos los valores. Y de la misma forma que antes, se envía ese arreglo como salida html.

El programa para el NodeMCU central.

```
/*
 Envio de datos para Google Sheets
 Hardware Utilizado: Nodemcu v1.0,
 Autor:Aurelio Gallardo Rodríguez
*/

// -- Bibliotecas auxiliares --

#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>

// -- Hardware --

// -- Variables y constantes --
constchar* ssid ="miSSID";// Rellena con el nombre de tu red WiFi
constchar* password ="miCONTRASEÑA";// Rellena con la contraseña de tu red WiFi

constchar* host ="script.google.com";// Este es el host de los scripts de google.
constint httpsPort =443;
```

```
// Huella digital del script de Google:
// D4:9E:40:F4:53:7A:04:93:38:F7:6B:4B:DC:70:02:A9:03:98:C2:DE
constchar* fingerprint ="D4 9E 40 F4 53 7A 04 93 38 F7 6B 4B DC 70 02 A9 03 98 C2
DE";
// const char* fingerprint = "46 B2 C3 44 9C 59 09 8B 01 B6 F8 BD 4C FB 00 74 91 2F EF
F6";

String googleSheetsID ="miCODIGO";// El que me da al implementar una aplicación web
en el script.

WiFiClientSecure cliente;
DynamicJsonDocument doc(1024);

int alarma =0;// detecta si se ha activado la alarma

int tiempoActivacionMinimo =3000;// Tiempo en ms de activación mínimo

#define LED 5// D1(gpio5)

// -- Setup --

void setup(){
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  connectToWiFi();
}

// -- LOOP: lectura del sensor y envío de datos según el intervalo--

void loop(){

  sendDataToGoogleSheets("CENTRAL");// se conecta a Google Spreadsheet



if(alarma>=1){
    digitalWrite(LED,true);// simplemente cuando cambie el valor de "alarma" se ilumina
    delay(tiempoActivacionMinimo);// dura encendida un tiempo
}else{
    digitalWrite(LED,false);
}



}

// -- Funciones auxiliares --

// -- Conectando a la red Wifi. Muestra la IP recibida --
```

```
void connectToWiFi(){
  Serial.println("Conectando a la red: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
while(WiFi.status()!= WL_CONNECTED){
    delay(500);
    Serial.print(".");
}
  WiFi.mode(WIFI_STA);
  Serial.println("");
  Serial.println("Conectado!");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
  delay(1000);
}


/* Función de conexión. Importante la instrucción  cliente.setInsecure(); para conectar de forma anónima
 */

void sendDataToGoogleSheets(String nombreEstacion){


String cadena="";


  Serial.print("Conectando a: ");
  Serial.println(host);

  cliente.setInsecure();

    if (!cliente.connect(host, httpsPort)) {
    Serial.println("Falla la conexión a Google Sheets -->" String(host) ": "

String(httpsPort));
return;
}


if(cliente.verify(fingerprint, host)){
    Serial.println("Certificado OK");
}
```

```
else{
   Serial.println("Debes comprobar certificado");
}

      String url = "/macros/s/" googleSheetsID "/exec?estacion="

nombreEstacion;



 Serial.print("Petición  URL ");
 Serial.println(url);

 cliente.print(String("GET ") url " HTTP/1.1\r\n"
           "Host: " host "\r\n"
           "User-Agent: BuildFailureDetectorESP8266\r\n"


"Connection: close\r\n\r\n");



// Es necesario leer las cabeceras
 Serial.println("Request enviada");
while(cliente.connected()){
String line = cliente.readStringUntil('\n');
if(line =="\r"){
    Serial.println("Cabeceras Recibidas");
    Serial.println(line);
break;
}
}

                  // lee el contenido de la respuesta y lo almacena en la variable cadena
 while (cliente.available()) {
    char c = cliente.read();
    cadena = cadena

(String) c;
}
 Serial.println("Respuesta directa del servidor");
 Serial.println(cadena);
 Serial.println("==============================");
 Serial.println();
```

```
// PROBLEMA: la respuesta, por motivos de seguridad, no es jamás el texto, o el html, sino que está codificada.
// He resuelto este problema encerrando la respuesta (JSON) entre dos palabras, EMPEZAR y TERMINAR.
// Una vez localizada la información, la extraigo con la función midString.
// Los caracteres, en codificación unicode, los transformo en sus caracteres con varias líneas de código...
// Y voalá... ¡obtiene la respuesta en formato JSON de texto!
String respuesta =midString(cadena,"EMPEZAR","TERMINAR");
char comillas =34;
  respuesta.replace("x7b","{");
  respuesta.replace("x7d","}");
  respuesta.replace("x22",String(comillas));
  respuesta.replace("\\","");// doble slash para que lo reconozca ??
// Serial.println(respuesta);



  Serial.println("Respuesta filtrada desde el servidor");
  Serial.println(respuesta);
  Serial.println("==============================");
  Serial.println();



// Usar bibliteca JSON
  deserializeJson(doc,respuesta);
  JsonObject obj = doc.as<JsonObject>();

  alarma =(int) obj[String("alarma")];// extraigo la variable alarma y la pongo en la variable del mismo valor.

     Serial.println("Respuesta extraída en variables deserializando la cadena en un objeto JSON");
  Serial.println("Estado de la alarma: "

(String) alarma);
  Serial.println("==============================");
  Serial.println();

  delay(1000);// Entre conexión y conexión 1 segundo.



  cliente.stop();//cierra la conexión
```

```
// ============================
/* Obteniendo la respuesta  */
// ============================



/*
 String line = cliente.readStringUntil('\n');
 Serial.println(line);
 if (line.startsWith("{\"state\":\"success\"")) {
  Serial.println("Éxito");
 } else {
  Serial.println("El envío falló!");
 }

 Serial.println("Respuesta:");
 Serial.println("==========");
 Serial.println(line);
 Serial.println("==========");
 Serial.println("¡Cerrando la conexión!"); */



}

/* función texto intermedio */
String midString(String str, String start, String finish){
 int locStart = str.indexOf(start);
 if (locStart==-1) return "";
 locStart += start.length();
 int locFinish = str.indexOf(finish, locStart);
 if (locFinish==-1) return "";
 return str.substring(locStart, locFinish);
}
```

El programa es muy parecido al de la estación. Inclusive más simple. Lee el valor de alarma, y si es igual o mayor que uno, activa el LED.

1.- Defino simplemente el LED como salida en D1 (GPIO 5).
2.- Defino la variable **alarma** (inicialmente a cero) y **tiempoActivacionMinimo** (inicialmente a 3000ms). La primera recogerá el estado de la alarma (si es uno o mayor se activará) y la otra variable recogerá el mínimo tiempo que permanecerá activada (independientemente si una estación desactiva la alarma seguirá activa durante ese tiempo).
3.- LOOP --> Se conecta a sendDataToGoogleSheets pasando el valor clave CENTRAL. La función recogerá y modificará el valor de la variable global alarma. Luego si alarma es uno o mayor o igual a uno, activará el LED y en caso contrario lo apagará.

4.- sendDataToGoogleSheets --> Las modificaciones realizadas permiten la activación del script de Google pasando la clave CENTRAL, la cual devuelve como respuesta el estado de cada estación y la suma de sus valores, y asigna esta última a la variable global alarma. **alarma = (int) obj[String("alarma")].**

Conclusiones.

Experimentalmente, el método funciona pero no parece muy efectivo. Recordad que en el post anterior expliqué que a veces la respuesta del servidor no es la que se espera y no puede leerse la cadena de datos desde el script de Google. Sin embargo, como cada segundo se repite la petición de la central, antes o después se activa. Si el valor **tiempoActivacionMinimo** es pequeño, incluso pueden producirse parpadeos. No sé por qué unas veces sí y otras no.

Aumentar el delay (y ponerlo en el loop y no al final de la función sendDataToGoogleSheets) puede mejorar el comportamiento. Quizás no dé respuesta correcta si se realizan peticiones muy seguidas, pero la verdad es que no sé por qué.

Tengo que investigar más. Saber si hay alguna manera de obtener una respuesta directa DESDE una hoja de cálculo de Google. Conocer otras posibles formas de intercomunicar los NodeMCU. Todo un mundo por investigar.

Posibles mejoras.

1. Evidentemente mejorar el hardware y la funcionalidad. Por ejemplo, que los leds de las estaciones y de la alarma indiquen, mediante parpadeos rápidos, si están puestos en estación, funcionando. O la central ampliarla con un twitter o un altavoz para producir la señal de alarma, etc.
2. Las mencionadas respecto a la comunicación entre los aparatos.
3. Alimentación de las estaciones. Un botón del pánico se encuentra bajo una mesa, luego debería activarse con una batería.