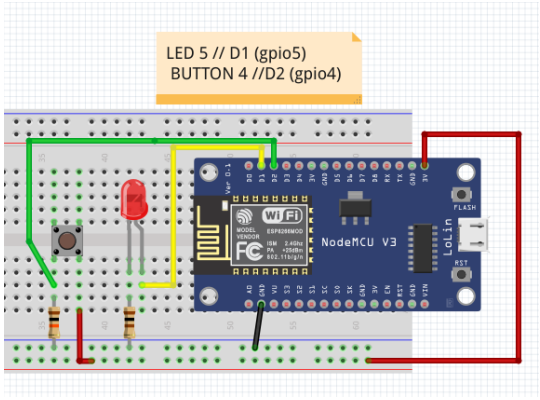
NodeMCU: proyecto "botón del pánico" (VI-B): el NodeMCU como estación.

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

Conexionado del NodeMCU



Programación

```
/*
 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 ="CODIGO";// El que me da al implementar una aplicación web en el script.
// Script EscribeDatos --> Código.gs --> De la hoja "DatosIntermedios"

WiFiClientSecure cliente;
DynamicJsonDocument doc(1024);

String estacion="A";
int activo=0;

#define LED 5// D1(gpio5)
#define BUTTON 4//D2(gpio4)
int estado=0;
int estadoAnterior=0;
int estadoLuz=0;

// -- Setup --

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

```
  pinMode(BUTTON, INPUT);
  connectToWiFi();
}

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

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
      Serial.println("Estado del LED: "

String(estadoLuz));//Manda al monitor serie ese estado.
      delay(10);
}
}
  digitalWrite(LED,estadoLuz);// Escribe el estado en un LED.
}



// -- 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, int Activacion){



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 "&activo="

(String) Activacion;



  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>();



String exito = obj[String("exito")];
String comentario = obj[String("comentario")];

      Serial.println("Respuesta extraída en variables deserializando la cadena en un objeto JSON");
  Serial.println("Ha tenido éxito: "exito);
  Serial.println("Envía el comentario: "

comentario);
  Serial.println("==============================");
  Serial.println();



  cliente.stop();//cierra 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);
}
```

= = = = = = = = = =

1.- El código empieza definiendo tres bibliotecas: una **ESP8266WiFi.h** con la que podremos usar el código en nuestro NodeMCU, otra **WiFiClientSecure.h** para conectarnos a una red Wifi y poder hacer transacciones seguras a través del protocolo https y otra **ArduinoJson.h** con la que podremos trabajar con objetos JSON.

2.- Declaramos después constantes de conexión: nuestra ssid, la contraseña, el host al que conectaremos, el puerto, la huella SHA1 y el código del script de google. Todo esto lo explicamos en la entrada NodeMCU: mando datos de un sensor a Google Hoja de cálculo (IV).

3.- Definimos después dos objetos: **cliente** de la clase WiFiClientSecure y **doc** de la clase DynamicJsonDocument. También las variables **estacion** (valor "A" pero admite hasta la "E"), la variable **activo** (inicialmente a cero), defino las variables **LED** y **BUTTON** en los GPIO 5 y 4 (D1 y D2 respectivamente en el NodeMCU), y las variables **estado**, **estadoAnterior** y **estadoLuz**, necesarias para que el pulsador actúe por flanco de subida. Todas inicialmente a cero.

4.- En el SETUP empiezo la conexión Serial, defino los pines del LED como salida y del BUTTON como entrada y llamo a la función conectar a la Wifi, exactamente igual que en post anteriores.

5.- LOOP --> lee el botón (digitalRead) y se almacena en la variable estado. Si la variable **estado** y **estadoAnterior** son distintas es que se ha producido un cambio. Si además **estado** es verdadero (o sea he pulsado el botón) entonces cambio el valor de **estadoLuz (**de 0 a 1, o de 1 a 0), y llamo a la función que manda el dato a la hoja de cálculo de google. Fuera de los ifs, el LED adquiere el estado de **estadoLuz** (digitalWrite). Además, una vez se produce un cambio de estado, se actualiza la variable **estadoAnterior**.

6.- Lo más destacable de la función **sendDataToGoogleSheets** es que recibe dos parámetros: el nombre de la estación y el estado, activo o no. Es prácticamente idéntica a la que vimos en el apartado NodeMCU: mando datos de un sensor a Google Hoja de cálculo (IV) pero con alguna diferencia importante. Primero, la url es distinta y se construye con esta línea: "**String url = "/macros/s/" + googleSheetsID + "/exec?estacion=" + nombreEstacion + "&activo=" + (String) Activacion;**". Segundo, una vez recibidas las cabeceras en la variable **cadena** almaceno toda la información que devuelve el servidor, y aquí viene la dificultad.

¿Qué hace el script de Google? Tanto la clase ContentService, capaz de enviar salidas de texto de diferentes tipos, como la clase HtmlOutput que genera un código HTML, no entregan la información directamente, sino que lo hacen a través de un redirecionamiento (*Due to security considerations, scripts cannot directly return content to a browser. Instead, they must serve the content from a different URL. This is why the URL in the browser for output created via this web app will be different than the original script URL*). Esto puede observarse bien analizando la URL de la respuesta en un navegador.

Por lo tanto nuestra variable cadena no contiene el objeto JSON de retorno que necesitaba. Contiene un código que no entiendo bien, pero que en definitiva, hace un redireccionamiento. Intenté conectarme, a su vez, al script redireccionado pero la respuesta es un nuevo

redireccionamiento. En definitiva ¿cómo obtener la información y no el código?

Puede que haya otras maneras de hacerlo. Quizás publicando el contenido de la hoja de cálculo y accediendo a sus datos a través de un JSON, o quizás de otra forma. Yo voy a explicar lo que conseguí en el código que os muestro.

La respuesta tiene esta forma:

<!doctype html>
<html>
<head>
<meta name="chromevox" content-script="no">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons"><link rel="stylesheet" href="/static/macros/client/css/2091290193-mae_html_css_ltr.css">
<script type="text/javascript" src="/static/macros/client/js/2221758526-warden_bin_i18n_warden__es.js"></script>
</head>
<body role="main">
<table id="warning-bar-table" class="full_size" cellspacing="0" cellpadding="0"><tr><td>
</td></tr><tr><td style="height: 100%"><iframe id="sandboxFrame" allow="accelerometer *; ambient-light-sensor *; autoplay *; camera *; encrypted-media *; fullscreen *; geolocation *; gyroscope *; magnetometer *; microphone *; midi *; payment *; picture-in-picture *; speaker *; usb *; vibrate *; vr *" sandbox="allow-forms allow-modals allow-popups allow-popups-to-escape-sandbox allow-same-origin allow-scripts allow-top-navigation">
</iframe>
</td></tr></table><script type="text/javascript">
(function() {
var el = document.getElementById('sandboxFrame');
el.onload = function() {
goog.script.init("\x7b\x22functionNames\x22:\x5b\x22doGet\x22,\x22construyeJSON\x22,\x22myFunction\x22\x5d,\x22sandboxMode\x2byqpps5hk2m4544dc3ivedt2u4gyxvbqn6phxi-0lu-script.googleusercontent.com\x22,\x22clientSideProperties\x22:\x7b\x22google.script.sandbox.mode\x22:\x22IFRAME_SANDBOX\x22,\x**A activo: 0\\\x22\x7dTERMINAR**\x22,\x22ncc\x22:\x22\x7b\\\x22awhs\\\x22:true\x7d\x22\x7d", "", undefined, true , false , false , "false", "https:\/\/n-2byqpps5hk2m4544dc3ivedt2u4gyxvbqn6phxi-0lu-script.googleusercontent.com");}
el.src = 'https:\/\/n-2byqpps5hk2m4544d

Y si nos fijamos, la información que envié, del tipo **EMPEZAR{"exito":"1","comentario":"Estación A activo: 0"}TERMINAR,** está enmascarada en dicha respuesta.

Así que las siguientes líneas de la función lo que hacen es extraer el texto entre las palabra EMPEZAR y TERMINAR, sustituir los códigos por sus caracteres, y reconstruir el objeto JSON con las funciones de la biblioteca.

También es cierto que a veces no se obtiene esta respuesta (¿parece que un delay de 500ms al final del procedimiento sendDataToGoogleSheets lo arregla?). Un número indeterminado de veces no se consigue esta información. Hay que seguir investigando. Si consigo un procedimiento seguro para obtener esta información en un NodeCMU, podremos hacer una central de alarma fiable a través de este método.