**Desafio 1**

**Projeto ESP 32\_GOOGLE\_SHEET**

O projeto realizado no Desafio 1 se consiste em um modelo físico realizo com:

- ESP32;

- Jumpers;

- Um botão e;

- Um Led;

Integrado a um código Arduino que salva o Status, Data e Hora do led.

O led funciona basicamente como um interruptor com o qual ao clicar no botão o Led acende e ao clicar novamente ele apaga, formando assim um fluxo contínuo.

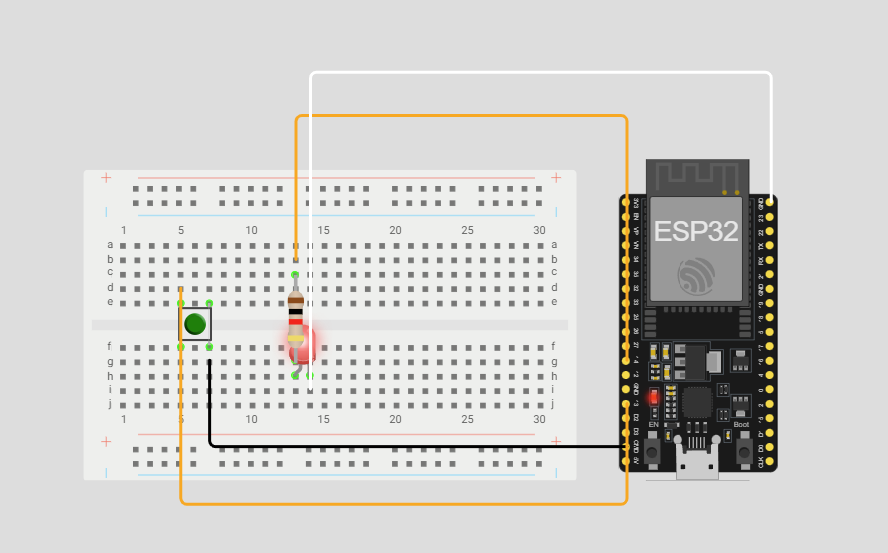
Nesta atividade pude obter a capacidade de realizar um código na plataforma Arduino integrado ao ESP32 com o qual necessitava de registrar no google sheet as seguintes informações:

- Data;

- Hora;

- Status (Ligado/Desligado);

Para a realização desse projeto realizei os seguintes passos:

1. Montei o modelo previamente no Wokiwi:  
   

Com o qual o **positivo (Vcc)** sai do ESP32 e alimenta o botão. Quando o botão é pressionado, ele envia esse sinal positivo para o pino do ESP32. Já o **LED** recebe energia de um pino de saída do ESP32, a corrente passa pelo LED, atravessa o resistor e segue para o **negativo (GND)**, fechando o circuito.

1. Após isso realizei o código na plataforma para a funcionalidade ser realizada:  
   // --------- PINOS ----------

const int ledPin = 14;

const int buttonPin = 13;

bool ledState = false;

int lastButtonState = HIGH;

void setup() {

**Serial**.begin(9600);

  pinMode(ledPin, OUTPUT);

  pinMode(buttonPin, INPUT\_PULLUP); // usa resistor interno de pull-up

}

void loop() {

  int buttonState = digitalRead(buttonPin);

  // Detecta clique (HIGH -> LOW)

  if (buttonState == LOW && lastButtonState == HIGH) {

    ledState = !ledState; // alterna estado

    digitalWrite(ledPin, ledState ? HIGH : LOW);

**Serial**.println(ledState ? "LED LIGADO" : "LED DESLIGADO");

    delay(300); // debounce

  }

  lastButtonState = buttonState;

}

Neste código, segui os seguintes passos:

O botão está configurado com INPUT\_PULLUP (resistor interno), então parado ele lê HIGH e pressionado lê LOW;

No loop, o ESP32 verifica a mudança de estado do botão ( solto / pressionado);

Quando detecta o clique, ele inverte o estado do LED (liga/desliga) e mostra no Serial Monitor se o LED está ligado ou desligado;

O delay(300) serve para evitar múltiplos cliques falsos (debounce).

1. Após isso montei o fluxo de forma física seguindo o Wokiwi;

Por último realizei o código na plataforma Arduino modificado para subir de forma correta no google sheet, realizei o seguinte código no Arduino:  
//código Giovana Cristina

#include <WiFi.h>

#include <WiFiClientSecure.h>

// === Rede Wi-Fi ===

const char\* ssid     = "WIFI\_EDUC\_CFP501";

const char\* password = "SENAICAMPINAS501";

// === Google Apps Script ===

const char\* host = "script.google.com";

const int httpsPort = 443;

const char\* GScriptId = "AKfycbyLI\_qG4klwnaDV26NhvTkLxJRYOcGHNbVshWO8jmiADuf1jV9VamgV\_0U80pkVCQ1B";

String url = String("/macros/s/") + GScriptId + "/exec";

// === Pinos ===

const int ledPin = 14;

const int buttonPin = 15;

// === Cliente HTTPS ===

WiFiClientSecure client;

// === Estado ===

bool ledState = false;

void setup() {

  Serial.begin(9600);

  pinMode(ledPin, OUTPUT);

  pinMode(buttonPin, INPUT\_PULLUP);

  // Conecta Wi-Fi

  WiFi.begin(ssid, password);

  Serial.print("Conectando-se à rede ");

  Serial.println(ssid);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("\nConectado!");

  Serial.print("IP: ");

  Serial.println(WiFi.localIP());

  client.setInsecure(); // Ignora certificado SSL

}

void sendToGoogleSheets(String status) {

  String payload = "{\"command\": \"append\_row\", \"sheet\_name\": \"Sheet1\", \"values\": \"" + status + "\"}";

  Serial.println("Enviando: " + payload);

  if (!client.connect(host, httpsPort)) {

    Serial.println("Falha na conexão HTTPS");

    return;

  }

  String request = String("POST ") + url + " HTTP/1.1\r\n" +

                   "Host: " + host + "\r\n" +

                   "Content-Type: application/json\r\n" +

                   "Content-Length: " + payload.length() + "\r\n\r\n" +

                   payload;

  client.print(request);

  while (client.connected()) {

    String line = client.readStringUntil('\n');

    if (line == "\r") break;

  }

  String response = client.readString();

  Serial.println("Resposta: " + response);

  client.stop();

}

void loop() {

  int buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) { // Botão pressionado

    ledState = !ledState;

    digitalWrite(ledPin, ledState ? HIGH : LOW);

    sendToGoogleSheets(ledState ? "Ativado" : "Desativado");

    Serial.println(ledState ? "LED Ativado" : "LED Desativado");

    delay(500); // debounce

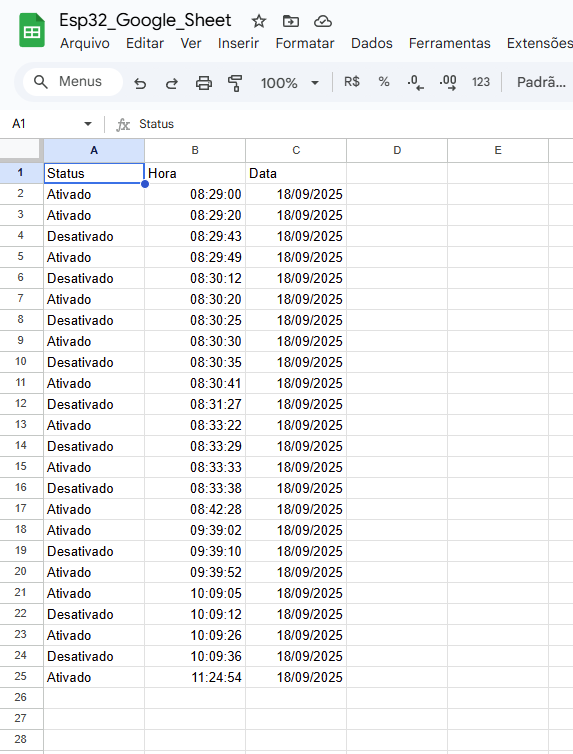
  }

}

Ele realize o seguinte fluxo:  
O LED liga ou desliga (inverte o estado).

O ESP32 envia uma mensagem (“Ativado” ou “Desativado”) para uma planilha do Google Sheets via Google Apps Script, usando uma requisição HTTPS.

O monitor serial mostra o que aconteceu (LED ligado/desligado + resposta da nuvem).

1. Após isso preparei uma planilha no Google Sheet com as seguintes colunas:  
   

Com a qual basicamente registra o status do led, o horário e a data que foi realizada a ação (clicar no botão e acender o led)

Após isso fui em “extensões”, “app script” e realizei o seguinte código:  
var SS = SpreadsheetApp.openById('1OpSlzok0LYWKxJSwwFRVHZuAI8wqJSxjASS-BoXCLoc');

// Troque pelo ID da sua planilha (entre /d/ e /edit na URL)

function doPost(e) {

  var parsedData;

  try {

    parsedData = JSON.parse(e.postData.contents);

  } catch (f) {

    return ContentService.createTextOutput("Erro ao interpretar JSON: " + f.message);

  }

  if (!parsedData) {

    return ContentService.createTextOutput("Erro! Corpo vazio ou formato incorreto.");

  }

  var sheet = SS.getSheetByName(parsedData.sheet\_name);

  if (!sheet) {

    return ContentService.createTextOutput("Erro: Planilha não encontrada.");

  }

  // Status vindo do ESP32

  var status = parsedData.values;

  // Pega hora e data do servidor Google

  var date\_now = Utilities.formatDate(new Date(), "America/Sao\_Paulo", "dd/MM/yyyy");

  var time\_now = Utilities.formatDate(new Date(), "America/Sao\_Paulo", "HH:mm:ss");

  switch (parsedData.command) {

    case "insert\_row":

      sheet.insertRows(2); // Insere nova linha no topo

      sheet.getRange('A2').setValue(status);   // Coluna A → Status

      sheet.getRange('B2').setValue(time\_now); // Coluna B → Hora

      sheet.getRange('C2').setValue(date\_now); // Coluna C → Data

      break;

    case "append\_row":

      sheet.appendRow([status, time\_now, date\_now]); // Adiciona no final

      break;

  }

  SpreadsheetApp.flush();

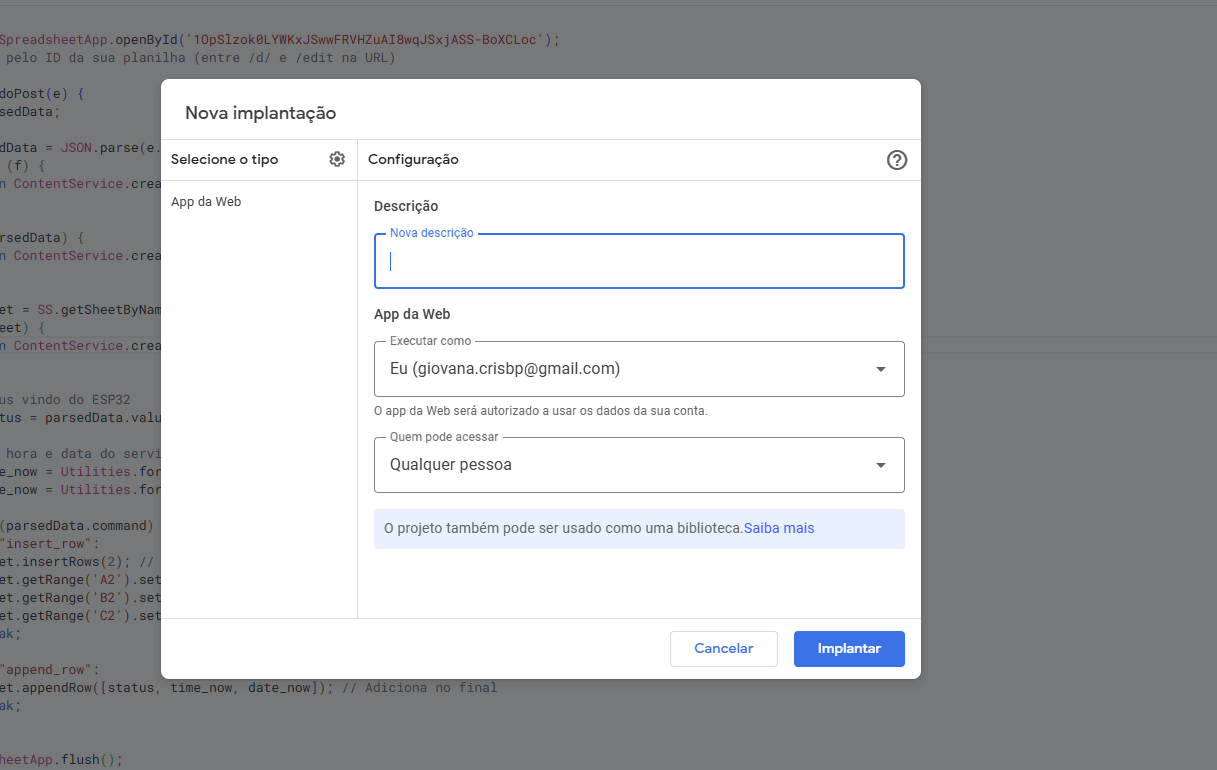
  return ContentService.createTextOutput("Success");

}

Ele registra os dados recebidos na planilha do google sheet, para ele fazer essa conexão passamos o código da url da planilha na parte que esta entre o d/ até o /edit.

1. Após isso fiz a implantação que vai ligar meu app script com o Arduino, para isso fui em:  
   - Implantar;

- Nova implantação;



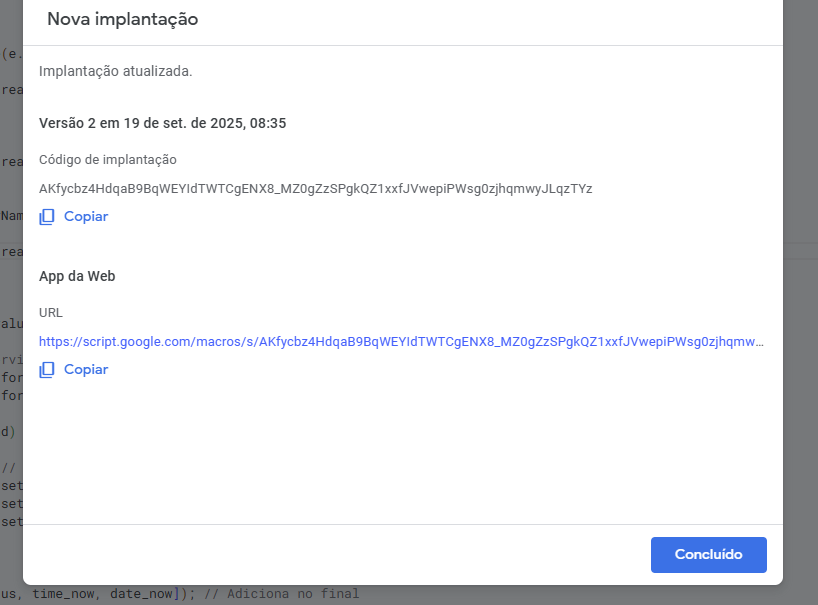
- Adicionei uma descrição;

- “Executar como”: Eu (meu E-mail);

- “Quem pode acessar”: Qualquer pessoa (é importante para o código Arduino poder reescrever sobre a planilha);

- Por fim cliquei em “Implantar”;

1. Por fim, copiei o código de implantação



- Colei nesta parte do código do Arduino:  
// === Google Apps Script ===

const char\* host = "script.google.com";

const int httpsPort = 443;

const char\* GScriptId = "AKfycbyLI\_qG4klwnaDV26NhvTkLxJRYOcGHNbVshWO8jmiADuf1jV9VamgV\_0U80pkVCQ1B";

String url = String("/macros/s/") + GScriptId + "/exec";

E executei, o resultado final foi a planilha do google sheet sendo atualizada de forma automática com a data, hora e status.