Perguntas Aula de IoT

Ana Luisa Sanchez

1. Qual a tensão (V) para ligar o LED?

* Um LED comum funciona entre 1.8V e 2.2V (dependendo da cor):
* Vermelho: ~1.8V
* Verde: ~2.0V
* Azul: ~3.0V

1. Qual pino do ESP32 tem tensão para alimentar o LED?

* Qualquer pino digital configurado como OUTPUT.
* Exemplo comum: GPIO 2, GPIO 5, GPIO 13, etc.

1. Qual é a tensão (V) de saída do ESP32?

* Pinos digitais do ESP32 fornecem 3.3V quando estão em nível HIGH.
* Em nível lógico LOW, fornecem ~0V

Resistor

1. Qual é a função do resistor?

* Limitar a corrente elétrica que passa pelo LED, protegendo-o contra queima.
* Sem resistor, o LED pode queimar rapidamente por excesso de corrente.
* Também pode dividir tensão, ajustar sensibilidade de sensores, etc.

1. Qual é a simbologia do resistor?

* **Padrão americano: uma linha em zigue-zague → —/\/\/—**
* **Padrão europeu: um retângulo → —□—**

1. Quais são os valores de resistência dos resistores disponibilizados. Explique como interpretar as cores.

* Valores comuns para LED: **220 Ω**, **330 Ω**, **1 kΩ**.
* Interpretação das cores → **Código de cores de resistores** (4 faixas):
  + 1ª faixa: primeiro dígito
  + 2ª faixa: segundo dígito
  + 3ª faixa: multiplicador (quantos zeros acrescentar)
  + 4ª faixa: tolerância

1. Publique dados na planilha googlesheets com ESP 32. Evidencie código do ESP32, código do AppScript e sua respectiva planilha com coluna em seu nome.

* Código ESP32

#include <WiFi.h>

#include <HTTPClient.h>

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

const char\* password = "SENAICAMPINAS501";

const char\* scriptURL = "https://script.google.com/macros/s/SEU\_SCRIPT\_ID/exec";

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.print("Conectando ao Wi-Fi");

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

delay(500);

Serial.print(".");

}

Serial.println("\nConectado!");

HTTPClient http;

String url = String(scriptURL) + "?nome=Ana+Luisa&valor=123";

http.begin(url);

int httpCode = http.GET();

Serial.println("Código HTTP: " + String(httpCode));

http.end();

}

void loop () {

}

* Código Google Apps Script

function doGet(e) {

var ss = SpreadsheetApp.getActiveSpreadsheet();

var sheet = ss.getActiveSheet();

var nome = e.parameter.nome;

var valor = e.parameter.valor;

var data = new Date ();

sheet.appendRow([data, nome, valor]);

return ContentService.createTextOutput("OK");

}

* Google sheets

//AULA3\_ITCOI\_Publicar Dados Google Sheets

//prof.Modesto

// Abre a planilha pelo seu ID único (que aparece na URL entre /d/ e /edit)

// Substitua este ID pelo ID real da sua planilha (está na URL entre /d/ e /edit)

var SS = SpreadsheetApp.openById('XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX');

// Função que será chamada automaticamente quando o ESP32 enviar dados via POST

function doPost(e) {

var parsedData; // Variável para guardar os dados enviados

var result = {}; // Pode ser usada para resposta futura (não usada aqui)

try {

// Converte os dados recebidos (em formato texto) para um objeto JSON

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

} catch (err) {

// Caso os dados não estejam em formato JSON válido, retorna um erro

return ContentService.createTextOutput("Erro ao analisar JSON: " + err.message);

}

// Verifica se os campos essenciais foram enviados

if (!parsedData || !parsedData.values || !parsedData.sheet\_name) {

return ContentService.createTextOutput("Erro! Dados ausentes ou em formato incorreto.");

}

// Tenta localizar a aba da planilha com o nome enviado

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

if (!sheet) {

// Se não encontrar, retorna erro

return ContentService.createTextOutput("Erro! Nome da aba não encontrado: " + parsedData.sheet\_name);

}

// Separa os valores recebidos (espera que venham separados por vírgula)

var dataArr = parsedData.values.split(",");

// Captura a data e hora atuais no fuso horário configurado no script

var timeZone = Session.getScriptTimeZone(); // Exemplo: "America/Sao\_Paulo"

var date\_now = Utilities.formatDate(new Date(), timeZone, "yyyy-MM-dd"); // Formato AAAA-MM-DD

var time\_now = Utilities.formatDate(new Date(), timeZone, "HH:mm:ss"); // Formato HH:MM:SS

// Guarda cada valor recebido em variáveis separadas (se não existir, usa vazio "")

var value0 = dataArr[0] || "";

var value1 = dataArr[1] || "";

var value2 = dataArr[2] || "";

// Escolhe o que fazer de acordo com o "comando" enviado

switch (parsedData.command) {

case "insert\_row":

// Insere uma nova linha logo abaixo do cabeçalho (linha 2)

sheet.insertRows(2);

// Preenche as colunas A, B, C, D e E com data, hora e valores recebidos

sheet.getRange('A2').setValue(date\_now);

sheet.getRange('B2').setValue(time\_now);

sheet.getRange('C2').setValue(value0);

sheet.getRange('D2').setValue(value1);

sheet.getRange('E2').setValue(value2);

// Garante que as alterações sejam aplicadas imediatamente

SpreadsheetApp.flush();

// Retorna mensagem de sucesso

return ContentService.createTextOutput("Sucesso - linha inserida");

case "append\_row":

// Cria um array com data, hora e valores para adicionar no final da planilha

var publish\_array = [date\_now, time\_now, value0, value1, value2];

sheet.appendRow(publish\_array);

SpreadsheetApp.flush();

return ContentService.createTextOutput("Sucesso - linha adicionada");

default:

// Caso o comando enviado não seja reconhecido

return ContentService.createTextOutput("Erro! Comando inválido.");

}

}