MICROCONTROLADORES ESP8266 e ESP32
(TEORIA E PRÁTICA)

Edição
Julho/2025

14 - ETAPA 2 — Registrador de Distâncias com Sensor Ultrassônico HC-SR04, ESP32 e SPIFFS (.CSV)

Objetivo:

Montar um sistema com o **ESP32 DevKit V1 (30 pinos Type-C)** e sensor **HC-SR04** para:

1. Medir distâncias continuamente.
2. Registrar as medições com data e hora em um **arquivo CSV no SPIFFS (memória interna Flash do ESP32)**.
3. Exibir o histórico de medições em uma página web.
4. **Permitir que o usuário baixe o arquivo `.csv` diretamente pela página HTML.**
5. O ESP32 estará conectado a uma **rede Wi-Fi existente (modo Station)**.

Materiais Necessários:

- ESP32 DevKit V1 Type-C (30 pinos)
- Sensor Ultrassônico HC-SR04
- Resistores para divisor de tensão (2kΩ e 1kΩ) ou level shifter lógico
- Jumpers, Protoboard
- Roteador Wi-Fi (ou rede Wi-Fi existente)
- Notebook/Celular para acesso via navegador

Esquema de Ligação (igual à Etapa 1):

Componente	Pino ESP32	Observações
HC-SR04 VCC	VIN (5V)	Alimentação 5V
HC-SR04 GND	GND	Terra
HC-SR04 Trig	D4 (GPIO4)	Saída de trigger
HC-SR04 Echo	D5 (GPIO5)	Entrada de echo (usar divisor de tensão 5V→3.3V)

Funcionamento Esperado:

1. O ESP32 realiza leituras do sensor HC-SR04 continuamente.
2. **Sempre que a distância for inferior a 400cm, um novo registro é salvo em um arquivo CSV (SPIFFS)**.
3. A página web exibe o histórico em tempo real e disponibiliza um **botão para baixar o arquivo .csv**.
4. Mesmo após reiniciar o ESP32, os dados continuam salvos.

Código Arduino IDE:

Antes de carregar o código, vá em **Ferramentas > Partitions Scheme > "Minimal SPIFFS (1.9MB APP/1.9MB SPIFFS)"** ou "Default (4MB)".

```
#include <WiFi.h>
#include <WebServer.h>
#include <FS.h>
#include <SPIFFS.h>
#include <time.h>
#include <sys/time.h>

const char* ssid = "SEU_SSID";
const char* password = "SUA_SENHA";

WebServer server(80);

const int trigPin = 4;
const int echoPin = 5;

#define CSV_FILE "/medicoes.csv"

// Função para medir a distância com HC-SR04
float measureDistance() {
  long duration;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH, 30000); // Timeout 30ms
  if (duration == 0) return 9999; // Sem leitura válida
  return (duration * 0.0343) / 2;
}

// Função para adicionar linha ao CSV no SPIFFS
void appendToCSV(String dataLine) {
  File file = SPIFFS.open(CSV_FILE, FILE_APPEND);
  if (!file) {
    Serial.println("Erro ao abrir o arquivo para escrita.");
    return;
  }
  file.println(dataLine);
  file.close();
}
```

```
// Configurar hora local com base em __DATE__ e __TIME__
void setLocalTimeFromBuild() {
  struct tm tm_build = {0};
  strptime(__DATE__ " " __TIME__, "%b %d %Y %H:%M:%S", &tm_build);
  time_t t = mktime(&tm_build);
  struct timeval now = { .tv_sec = t };
  settimeofday(&now, NULL);
}


// Página HTML com tabela de medições e link para download
void handleRoot() {
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>";
  html += "<meta http-equiv='refresh' content='5'>";
  html += "<title>Histórico de Medições</title></head><body>";
  html += "<h1>Histórico de Medições Registradas (&lt; 400 cm)</h1>";
  html += "<a href='/download'>📥 Baixar arquivo CSV</a><br><br>";
  html += "<table border='1'><tr><th>Data/Hora</th><th>Distância (cm)</th></tr>";

  File file = SPIFFS.open(CSV_FILE, FILE_READ);
  if (file) {
    while (file.available()) {
      String line = file.readStringUntil('\n');
      int sep = line.indexOf(',');
      if (sep > 0) {
        String dataHora = line.substring(0, sep);
        String distancia = line.substring(sep + 1);
        html += "<tr><td>" + dataHora + "</td><td>" + distancia + "</td></tr>";
      }
    }
    file.close();
  }

  html += "</table></body></html>";
  server.send(200, "text/html", html);
}

// Manipulador do botão de download
void handleDownload() {
  File file = SPIFFS.open(CSV_FILE, FILE_READ);
  if (!file) {
    server.send(500, "text/plain", "Arquivo não encontrado");
    return;
  }
  server.sendHeader("Content-Type", "text/csv");
  server.sendHeader("Content-Disposition", "attachment; filename=\"medicoes.csv\"");
  server.streamFile(file, "text/csv");
  file.close();
}

void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Inicializar SPIFFS
  if (!SPIFFS.begin(true)) {
    Serial.println("Erro ao montar SPIFFS");
    return;
  }
```

```
  // Conectar ao WiFi
  WiFi.begin(ssid, password);
  Serial.print("Conectando à rede");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado! IP: " + WiFi.localIP().toString());

  // Setar hora da compilação
  setLocalTimeFromBuild();

  // Iniciar servidor web
  server.on("/", handleRoot);
  server.on("/download", handleDownload);
  server.begin();
}

void loop() {
  float dist = measureDistance();

  if (dist < 400.0) {
    struct tm timeinfo;
    String dataHora = "N/A";

    if (getLocalTime(&timeinfo)) {
      char timeStringBuff[64];
      strftime(timeStringBuff, sizeof(timeStringBuff), "%d/%m/%Y %H:%M:%S", &timeinfo);
      dataHora = String(timeStringBuff);
    }

    String dataLine = dataHora + "," + String(dist, 2);
    Serial.println("Registrado: " + dataLine);
    appendToCSV(dataLine);

    delay(1000);  // Delay para evitar registros repetidos
  }

  server.handleClient();
}
```

Explicações Importantes:

Item	Descrição
SPIFFS	Sistema de arquivos interno (grava na Flash do ESP32)
Arquivo CSV	Nome: /medicoes.csv (armazenado em SPIFFS)
Atualização da página web	A cada 5 segundos mostra o histórico lido diretamente do arquivo
Botão de Download	Um link para /download permite baixar o arquivo CSV
Persistência dos dados	Mesmo após reiniciar o ESP32, os dados continuam no arquivo
Limitação de espaço SPIFFS	Normalmente até 1.5MB ou 3MB (ver Partitions Scheme nas configurações)

🧠 O que aprendemos nesta Etapa 2:

1. Como utilizar SPIFFS para armazenar dados no ESP32.
2. Como criar arquivos CSV dinamicamente no microcontrolador.

3. Como servir arquivos diretamente pelo navegador (download via HTTP).
4. Noções de sistemas de arquivos embarcados.
5. Conceito de "logger de sensores" persistente.

Bibliografia:
https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html
https://randomnerdtutorials.com/
http://www.practical-arduino.com/