01 Sensor ultrasónico

// Pines del sensor ultrasónico

const int pinTrig = 5;

const int pinEcho = 18;

long duracion;

int distancia;

void setup() {

Serial.begin(115200);

pinMode(pinTrig, OUTPUT);

pinMode(pinEcho, INPUT);

}

void loop() {

// Generar pulso de activación

digitalWrite(pinTrig, LOW);

delayMicroseconds(2);

digitalWrite(pinTrig, HIGH);

delayMicroseconds(10);

digitalWrite(pinTrig, LOW);

// Medir duración del pulso de respuesta

duracion = pulseIn(pinEcho, HIGH);

// Calcular distancia (en cm)

distancia = duracion \* 0.034 / 2;

// Mostrar en consola

Serial.print("Distancia: ");

Serial.print(distancia);

Serial.println(" cm");

delay(500);

}

02 led

const int pinTrig = 5;

const int pinEcho = 18;

const int pinLED = 13;

long duracion;

int distancia;

void setup() {

Serial.begin(115200);

pinMode(pinTrig, OUTPUT);

pinMode(pinEcho, INPUT);

pinMode(pinLED, OUTPUT);

}

void loop() {

// Disparo ultrasónico

digitalWrite(pinTrig, LOW); delayMicroseconds(2);

digitalWrite(pinTrig, HIGH); delayMicroseconds(10);

digitalWrite(pinTrig, LOW);

// Medición de pulso

duracion = pulseIn(pinEcho, HIGH);

distancia = duracion \* 0.034 / 2;

Serial.print("Distancia: ");

Serial.print(distancia);

Serial.println(" cm");

// Control del LED

if (distancia > 0 && distancia <= 30) {

digitalWrite(pinLED, HIGH); // Obstáculo cercano

} else {

digitalWrite(pinLED, LOW); // Todo despejado

}

delay(500);

}

03 Servo

#include <Servo.h>

const int pinTrig = 5;

const int pinEcho = 18;

const int pinLED = 13;

const int pinServo = 25;

Servo servo;

long duracion;

int distancia;

void setup() {

Serial.begin(115200);

pinMode(pinTrig, OUTPUT);

pinMode(pinEcho, INPUT);

pinMode(pinLED, OUTPUT);

servo.attach(pinServo);

servo.write(0); // posición inicial

}

void loop() {

// Medición ultrasónica

digitalWrite(pinTrig, LOW); delayMicroseconds(2);

digitalWrite(pinTrig, HIGH); delayMicroseconds(10);

digitalWrite(pinTrig, LOW);

duracion = pulseIn(pinEcho, HIGH);

distancia = duracion \* 0.034 / 2;

Serial.print("Distancia: ");

Serial.print(distancia);

Serial.println(" cm");

if (distancia > 0 && distancia <= 30) {

digitalWrite(pinLED, HIGH);

servo.write(90); // Gira al frente

} else {

digitalWrite(pinLED, LOW);

servo.write(0); // Regresa a reposo

}

delay(500);

}

04 Push button

#include <Servo.h>

const int pinTrig = 5;

const int pinEcho = 18;

const int pinLED = 13;

const int pinServo = 25;

const int pinBoton = 19;

Servo servo;

long duracion;

int distancia;

void setup() {

Serial.begin(115200);

pinMode(pinTrig, OUTPUT);

pinMode(pinEcho, INPUT);

pinMode(pinLED, OUTPUT);

pinMode(pinBoton, INPUT\_PULLUP); // activa resistencia interna

servo.attach(pinServo);

servo.write(0);

}

void loop() {

bool activarSistema = digitalRead(pinBoton) == LOW;

if (activarSistema) {

// Medición ultrasónica

digitalWrite(pinTrig, LOW); delayMicroseconds(2);

digitalWrite(pinTrig, HIGH); delayMicroseconds(10);

digitalWrite(pinTrig, LOW);

duracion = pulseIn(pinEcho, HIGH);

distancia = duracion \* 0.034 / 2;

Serial.print("Distancia: ");

Serial.print(distancia);

Serial.println(" cm");

if (distancia > 0 && distancia <= 30) {

digitalWrite(pinLED, HIGH);

servo.write(90);

} else {

digitalWrite(pinLED, LOW);

servo.write(0);

}

} else {

// Sistema inactivo

digitalWrite(pinLED, LOW);

servo.write(0);

}

delay(500);

}

05 Codigo Estructurado

#include <WiFi.h>

#include <Servo.h>

// Wi-Fi

**const char\* ssid = "TU\_SSID";**

**const char\* password = "TU\_CONTRASEÑA";**

// Pines sensores

const int pinTrig = 5;

const int pinEcho = 18;

const int pinLED = 13;

const int pinServo = 25;

const int pinBoton = 19;

WiFiServer server(80);

Servo servo;

// Variables

long duracion;

int distancia;

String estado = "inactivo";

void setup() {

Serial.begin(115200);

// Configuración pines

pinMode(pinTrig, OUTPUT);

pinMode(pinEcho, INPUT);

pinMode(pinLED, OUTPUT);

pinMode(pinBoton, INPUT\_PULLUP);

servo.attach(pinServo);

servo.write(0); // posición inicial

// Conectar Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Conectando");

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

delay(500); Serial.print(".");

}

Serial.println("\n Conectado");

Serial.print("Dirección IP local: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop() {

bool sistemaActivo = digitalRead(pinBoton) == LOW;

if (sistemaActivo) {

medirDistancia();

if (distancia > 0 && distancia <= 30) {

digitalWrite(pinLED, HIGH);

servo.write(90);

estado = "peligro";

} else {

digitalWrite(pinLED, LOW);

servo.write(0);

estado = "seguro";

}

} else {

digitalWrite(pinLED, LOW);

servo.write(0);

estado = "inactivo";

distancia = 0;

}

manejarServidor();

delay(200);

}

void medirDistancia() {

digitalWrite(pinTrig, LOW); delayMicroseconds(2);

digitalWrite(pinTrig, HIGH); delayMicroseconds(10);

digitalWrite(pinTrig, LOW);

duracion = pulseIn(pinEcho, HIGH);

distancia = duracion \* 0.034 / 2;

Serial.print("Distancia: ");

Serial.print(distancia);

Serial.println(" cm");

}

void manejarServidor() {

WiFiClient cliente = server.available();

if (cliente) {

String request = cliente.readStringUntil('\r');

cliente.flush();

if (request.indexOf("GET /distancia") >= 0) {

cliente.println("HTTP/1.1 200 OK");

cliente.println("Content-Type: application/json");

cliente.println();

cliente.print("{\"distancia\": ");

cliente.print(distancia);

cliente.println("}");

}

else if (request.indexOf("GET /estado") >= 0) {

cliente.println("HTTP/1.1 200 OK");

cliente.println("Content-Type: application/json");

cliente.println();

cliente.print("{\"estado\": \"");

cliente.print(estado);

cliente.println("\"}");

}

delay(10);

cliente.stop();

}

}

06 CONEXION-PRUEBA

include <WiFi.h>

// 🔧 Credenciales de tu red

**const char\* ssid = "INFINITUMBE3E";**

**const char\* password = "5365628613";**

void setup() {

Serial.begin(115200);

delay(1000);

Serial.println("Conectando a WiFi...");

WiFi.begin(ssid, password);

// Esperar conexión

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

delay(500);

Serial.print(".");

}

Serial.println("\nConectado a WiFi");

Serial.print("IP del ESP32: ");

Serial.println(WiFi.localIP());

}

void loop() {

// No hacemos nada aquí todavía

}

07 WebServer

#include <WiFi.h>

#include <WebServer.h>

//Credenciales de tu red

**const char\* ssid = "INFINITUMBE3E";**

**const char\* password = "5365628613";**

// Crear servidor en el puerto 80 (por defecto)

WebServer server(80);

// Página web en HTML (entre comillas largas)

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Sensor Web</title>

<style>

body { font-family: sans-serif; text-align: center; margin-top: 50px; }

button { padding: 15px 25px; font-size: 18px; background-color: #4CAF50; color: white; border: none; border-radius: 5px; }

</style>

</head>

<body>

<h2> Botón Virtual</h2>

<button onclick="accion()">Presionar</button>

<p id="mensaje">Esperando acción...</p>

<script>

function accion() {

fetch("/presionar")

.then(response => response.text())

.then(data => {

document.getElementById("mensaje").innerText = data;

});

}

</script>

</body>

</html>

)rawliteral";

void setup() {

Serial.begin(115200);

// Conectarse a Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n Conectado a WiFi");

Serial.print(" IP del ESP32: ");

Serial.println(WiFi.localIP());

// Ruta raíz muestra la página web

server.on("/", []() {

server.send(200, "text/html", htmlPage);

});

// Ruta que responde al botón virtual

server.on("/presionar", []() {

server.send(200, "text/plain", "Botón presionado correctamente");

});

server.begin();

}

void loop() {

server.handleClient();

}

08 Ultrasonico

#include <WiFi.h>

#include <WebServer.h>

// Datos de red

const char\* ssid = "INFINITUMBE3E";

const char\* password = "5365628613";

WebServer server(80);

// Pines del sensor ultrasónico

const int trigPin = 5;

const int echoPin = 18;

// Estado del sistema

bool sistemaActivo = false;

// Página web con botón virtual

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Sensor Web</title>

<style>

body { font-family: sans-serif; text-align: center; margin-top: 50px; }

button { padding: 15px 25px; font-size: 18px; background-color: #007BFF; color: white; border: none; border-radius: 5px; }

#estado { font-size: 20px; margin-top: 10px; }

#resultado { font-size: 18px; margin-top: 15px; }

</style>

</head>

<body>

<h2> Sistema de Medición</h2>

<button id="botonSistema" onclick="toggleSistema()">Sistema apagado</button>

<p id="estado">Sistema inactivo</p>

<p id="resultado">Sin lectura</p>

<script>

let encendido = false;

let intervalo;

function toggleSistema() {

encendido = !encendido;

let accion = encendido ? "encender" : "apagar";

fetch("/" + accion)

.then(response => response.text())

.then(data => {

document.getElementById("botonSistema").innerText = encendido ? "Sistema encendido" : "Sistema apagado";

document.getElementById("estado").innerText = encendido ? " Activo" : "Apagado";

document.getElementById("resultado").innerText = data;

});

// Activar lectura continua

if (encendido) {

intervalo = setInterval(() => {

fetch("/medir")

.then(r => r.text())

.then(d => {

document.getElementById("resultado").innerText = d;

});

}, 1000); // cada 1 segundo

} else {

clearInterval(intervalo);

document.getElementById("resultado").innerText = "Sistema apagado. Sin lectura.";

}

}

</script>

</body>

</html>

)rawliteral";

// Función para medir distancia

long medirDistancia() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duracion = pulseIn(echoPin, HIGH);

long distancia = duracion \* 0.034 / 2; // cm

return distancia;

}

void setup() {

Serial.begin(115200);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

// Conexión Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n Conectado!");

Serial.print("IP del ESP32: ");

Serial.println(WiFi.localIP());

// Ruta principal

server.on("/", []() {

server.send(200, "text/html", htmlPage);

});

// Encender sistema

server.on("/encender", []() {

sistemaActivo = true;

long distancia = medirDistancia();

String respuesta = "Distancia inicial: " + String(distancia) + " cm";

server.send(200, "text/plain", respuesta);

});

// Apagar sistema

server.on("/apagar", []() {

sistemaActivo = false;

server.send(200, "text/plain", "Sistema apagado. Sin lectura.");

});

// Lectura continua

server.on("/medir", []() {

if (sistemaActivo) {

long distancia = medirDistancia();

String resultado = "Distancia: " + String(distancia) + " cm";

server.send(200, "text/plain", resultado);

} else {

server.send(200, "text/plain", "Sistema inactivo");

}

});

server.begin();

}

void loop() {

server.handleClient();

}

09 LED

#include <WiFi.h>

#include <WebServer.h>

// Credenciales de red

const char\* ssid = "INFINITUMBE3E";

const char\* password = "5365628613";

// Servidor web en puerto 80

WebServer server(80);

// Pines del sensor y LED

const int trigPin = 5;

const int echoPin = 18;

**const int ledPin = 13;**

// Estado del sistema

bool sistemaActivo = false;

// Página web alojada en el ESP32

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Sensor Web</title>

<style>

body { font-family: sans-serif; text-align: center; margin-top: 50px; }

button {

padding: 15px 25px;

font-size: 18px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 5px;

}

#estado { font-size: 20px; margin-top: 10px; }

#resultado { font-size: 18px; margin-top: 15px; }

</style>

</head>

<body>

<h2>Sistema de Medición</h2>

<button id="botonSistema" onclick="toggleSistema()">Sistema apagado</button>

<p id="estado">Sistema inactivo</p>

<p id="resultado">Sin lectura</p>

<script>

let encendido = false;

let intervalo;

function toggleSistema() {

encendido = !encendido;

let accion = encendido ? "encender" : "apagar";

fetch("/" + accion)

.then(response => response.text())

.then(data => {

document.getElementById("botonSistema").innerText = encendido ? "Sistema encendido" : "Sistema apagado";

document.getElementById("estado").innerText = encendido ? " Activo" : " Apagado";

document.getElementById("resultado").innerText = data;

});

if (encendido) {

intervalo = setInterval(() => {

fetch("/medir")

.then(r => r.text())

.then(d => {

document.getElementById("resultado").innerText = d;

});

}, 1000); // cada segundo

} else {

clearInterval(intervalo);

document.getElementById("resultado").innerText = " Sistema apagado. Sin lectura.";

}

}

</script>

</body>

</html>

)rawliteral";

// Función para medir distancia con sensor HC-SR04

long medirDistancia() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duracion = pulseIn(echoPin, HIGH);

long distancia = duracion \* 0.034 / 2; // cm

return distancia;

}

void setup() {

Serial.begin(115200);

// Configurar pines

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

**pinMode(ledPin, OUTPUT);**

digitalWrite(ledPin, LOW); // LED apagado al iniciar

// Conectarse al Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n Conectado!");

Serial.print(" IP del ESP32: ");

Serial.println(WiFi.localIP());

// Mostrar página web

server.on("/", []() {

server.send(200, "text/html", htmlPage);

});

// Activar sistema

server.on("/encender", []() {

sistemaActivo = true;

long distancia = medirDistancia();

String respuesta = " Distancia inicial: " + String(distancia) + " cm";

server.send(200, "text/plain", respuesta);

});

// Apagar sistema

server.on("/apagar", []() {

sistemaActivo = false;

digitalWrite(ledPin, LOW); // apagar LED por seguridad

server.send(200, "text/plain", " Sistema apagado. Sin lectura.");

});

// Lecturas periódicas

server.on("/medir", []() {

if (sistemaActivo) {

long distancia = medirDistancia();

String resultado = " Distancia: " + String(distancia) + " cm";

if (distancia < 30) {

digitalWrite(ledPin, HIGH); // LED encendido

resultado += "\n ¡Distancia peligrosa! LED encendido.";

} else {

digitalWrite(ledPin, LOW); // LED apagado

resultado += "\n Zona segura. LED apagado.";

}

server.send(200, "text/plain", resultado);

} else {

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", " Sistema inactivo");

}

});

server.begin();

}

void loop() {

server.handleClient();

}

10 Adaptado

#include <WiFi.h>

#include <WebServer.h>

// Datos de red Wi-Fi

const char\* ssid = "INFINITUMBE3E";

const char\* password = "5365628613";

// Servidor HTTP en puerto estándar

WebServer server(80);

// Pines del sensor ultrasónico HC-SR04

const int trigPin = 5;

const int echoPin = 18;

// Pin del LED de alerta

const int ledPin = 13;

// Variable que controla si el sistema está activo

bool sistemaActivo = false;

// Interfaz web adaptada para celular, con estilos visuales

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sensor Web</title>

<style>

body {

font-family: sans-serif;

text-align: center;

margin: 0;

padding: 0;

background-color: #f5f5f5;

}

h2 {

margin-top: 30px;

color: #333;

}

button {

padding: 15px 25px;

font-size: 18px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 8px;

margin-top: 20px;

width: 80%;

}

#estado {

font-size: 20px;

margin-top: 15px;

font-weight: bold;

}

#resultado {

font-size: 18px;

margin-top: 10px;

}

</style>

</head>

<body>

<h2>📲 Sistema de Medición</h2>

<button id="botonSistema" onclick="toggleSistema()">Sistema apagado</button>

<p id="estado">Sistema inactivo</p>

<p id="resultado">Sin lectura</p>

<script>

let encendido = false;

let intervalo;

function toggleSistema() {

encendido = !encendido;

let accion = encendido ? "encender" : "apagar";

fetch("/" + accion)

.then(response => response.text())

.then(data => {

document.getElementById("botonSistema").innerText = encendido ? "Sistema encendido" : "Sistema apagado";

document.getElementById("estado").innerText = encendido ? "🔴 Activo" : "⚪ Apagado";

document.getElementById("estado").style.color = encendido ? "red" : "gray";

document.getElementById("resultado").innerText = data;

document.getElementById("resultado").style.color = "black";

});

if (encendido) {

intervalo = setInterval(() => {

fetch("/medir")

.then(r => r.text())

.then(d => {

document.getElementById("resultado").innerText = d;

if (d.includes("peligrosa")) {

document.getElementById("resultado").style.color = "red";

} else if (d.includes("segura")) {

document.getElementById("resultado").style.color = "green";

} else {

document.getElementById("resultado").style.color = "black";

}

});

}, 1000);

} else {

clearInterval(intervalo);

document.getElementById("resultado").innerText = "🛑 Sistema apagado. Sin lectura.";

document.getElementById("resultado").style.color = "gray";

}

}

</script>

</body>

</html>

)rawliteral";

// Función para medir distancia desde el sensor

long medirDistancia() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duracion = pulseIn(echoPin, HIGH);

long distancia = duracion \* 0.034 / 2; // Convertir tiempo en cm

return distancia;

}

void setup() {

Serial.begin(115200);

// Configurar pines del sensor y LED

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW); // LED apagado al iniciar

// Conexión a la red Wi-Fi

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n Conectado!");

Serial.print(" IP del ESP32: ");

Serial.println(WiFi.localIP());

// Mostrar página principal en navegador

server.on("/", []() {

server.send(200, "text/html", htmlPage);

});

// Activar sistema (encendido desde el botón)

server.on("/encender", []() {

sistemaActivo = true;

long distancia = medirDistancia();

String respuesta = " Distancia inicial: " + String(distancia) + " cm";

server.send(200, "text/plain", respuesta);

});

// Apagar sistema (apagado desde el botón)

server.on("/apagar", []() {

sistemaActivo = false;

digitalWrite(ledPin, LOW); // Seguridad: apagar LED

server.send(200, "text/plain", "🛑 Sistema apagado. Sin lectura.");

});

// Lecturas automáticas cada segundo

server.on("/medir", []() {

if (sistemaActivo) {

long distancia = medirDistancia();

String resultado = "📏 Distancia: " + String(distancia) + " cm";

if (distancia < 30) {

digitalWrite(ledPin, HIGH);

resultado += "\n⚠️ ¡Distancia peligrosa! LED encendido.";

} else {

digitalWrite(ledPin, LOW);

resultado += "\n✅ Zona segura. LED apagado.";

}

server.send(200, "text/plain", resultado);

} else {

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema inactivo");

}

});

server.begin(); // Iniciar el servidor web

}

void loop() {

server.handleClient(); // Mantener la conexión

}

11 HISTORIAL

#include <WiFi.h>

#include <WebServer.h>

// 📶 Datos de red

const char\* ssid = "INFINITUMBE3E";

const char\* password = "5365628613";

// 🌐 Servidor HTTP

WebServer server(80);

// 📍 Pines del sensor y LED

const int trigPin = 5;

const int echoPin = 18;

const int ledPin = 13;

// 🔁 Estado del sistema

bool sistemaActivo = false;

// 🌍 Interfaz web

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sensor Web</title>

<style>

body {

font-family: sans-serif;

text-align: center;

margin: 0;

padding: 0;

background-color: #f5f5f5;

}

h2 {

margin-top: 30px;

color: #333;

}

button {

padding: 15px 25px;

font-size: 18px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 8px;

margin-top: 20px;

width: 80%;

}

#estado {

font-size: 20px;

margin-top: 15px;

font-weight: bold;

}

#resultado {

font-size: 18px;

margin-top: 10px;

white-space: pre-line;

}

table {

margin: 20px auto;

border-collapse: collapse;

width: 90%;

}

th, td {

border: 1px solid #ccc;

padding: 8px;

font-size: 16px;

}

th {

background-color: #e0e0e0;

}

</style>

</head>

<body>

<h2>📲 Sistema de Medición</h2>

<button id="botonSistema" onclick="toggleSistema()">Sistema apagado</button>

<p id="estado">Sistema inactivo</p>

<p id="resultado">Sin lectura</p>

<h3>📊 Historial de Lecturas</h3>

<table id="tablaHistorial">

<tr><th>#</th><th>Distancia (cm)</th><th>Estado</th><th>Hora</th></tr>

</table>

<script>

let encendido = false;

let intervalo;

let contador = 1;

function toggleSistema() {

encendido = !encendido;

let accion = encendido ? "encender" : "apagar";

fetch("/" + accion)

.then(response => response.text())

.then(data => {

document.getElementById("botonSistema").innerText = encendido ? "Sistema encendido" : "Sistema apagado";

document.getElementById("estado").innerText = encendido ? "🔴 Activo" : "⚪ Apagado";

document.getElementById("estado").style.color = encendido ? "red" : "gray";

document.getElementById("resultado").innerText = data;

document.getElementById("resultado").style.color = "black";

});

if (encendido) {

intervalo = setInterval(() => {

fetch("/medir")

.then(r => r.text())

.then(d => {

document.getElementById("resultado").innerText = d;

let lineas = d.split("\n");

let distancia = lineas[0].split(":")[1].trim().replace("cm", "");

let estado = lineas[1] || "";

let hora = new Date().toLocaleTimeString();

let tabla = document.getElementById("tablaHistorial");

let fila = tabla.insertRow(-1);

fila.insertCell(0).innerText = contador++;

fila.insertCell(1).innerText = distancia;

fila.insertCell(2).innerText = estado;

fila.insertCell(3).innerText = hora;

if (estado.includes("peligrosa")) {

document.getElementById("resultado").style.color = "red";

} else if (estado.includes("segura")) {

document.getElementById("resultado").style.color = "green";

} else {

document.getElementById("resultado").style.color = "black";

}

});

}, 1000);

} else {

clearInterval(intervalo);

document.getElementById("resultado").innerText = "🛑 Sistema apagado. Sin lectura.";

document.getElementById("resultado").style.color = "gray";

}

}

</script>

</body>

</html>

)rawliteral";

// 📏 Medir distancia en cm

long medirDistancia() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duracion = pulseIn(echoPin, HIGH);

long distancia = duracion \* 0.034 / 2;

return distancia;

}

void setup() {

Serial.begin(115200);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n✅ Conectado!");

Serial.print("📲 IP del ESP32: ");

Serial.println(WiFi.localIP());

server.on("/", []() {

server.send(200, "text/html", htmlPage);

});

server.on("/encender", []() {

sistemaActivo = true;

long distancia = medirDistancia();

String respuesta = "📏 Distancia inicial: " + String(distancia) + " cm\n";

server.send(200, "text/plain", respuesta);

});

server.on("/apagar", []() {

sistemaActivo = false;

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema apagado. Sin lectura.");

});

server.on("/medir", []() {

if (sistemaActivo) {

long distancia = medirDistancia();

String resultado = "📏 Distancia: " + String(distancia) + " cm\n";

if (distancia < 30) {

digitalWrite(ledPin, HIGH);

resultado += "⚠️ ¡Distancia peligrosa! LED encendido.";

} else {

digitalWrite(ledPin, LOW);

resultado += "✅ Zona segura. LED apagado.";

}

server.send(200, "text/plain", resultado);

} else {

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema inactivo");

}

});

server.begin();

}

void loop() {

server.handleClient();

}

12 Exportar

#include <WiFi.h>

#include <WebServer.h>

// 🔧 Red Wi-Fi

const char\* ssid = "INFINITUMBE3E";

const char\* password = "5365628613";

// 🌐 Servidor Web

WebServer server(80);

// 📍 Pines del sensor y LED

const int trigPin = 5;

const int echoPin = 18;

const int ledPin = 13;

// 🔁 Estado del sistema

bool sistemaActivo = false;

// 🌍 Página web completa con tabla y exportación

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sensor Web</title>

<style>

body { font-family: sans-serif; text-align: center; margin: 0; padding: 0; background-color: #f5f5f5; }

h2 { margin-top: 30px; color: #333; }

button {

padding: 15px 25px;

font-size: 18px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 8px;

margin-top: 15px;

width: 80%;

}

#estado { font-size: 20px; margin-top: 10px; font-weight: bold; }

#resultado { font-size: 18px; margin-top: 10px; white-space: pre-line; }

table {

margin: 20px auto;

border-collapse: collapse;

width: 95%;

}

th, td {

border: 1px solid #ccc;

padding: 8px;

font-size: 16px;

text-align: center;

}

th {

background-color: #e0e0e0;

}

</style>

</head>

<body>

<h2>📲 Sistema de Medición</h2>

<button id="botonSistema" onclick="toggleSistema()">Sistema apagado</button>

<p id="estado">Sistema inactivo</p>

<p id="resultado">Sin lectura</p>

<h3>📊 Historial de Lecturas</h3>

<table id="tablaHistorial">

<tr><th>#</th><th>Distancia (cm)</th><th>Estado</th><th>Hora</th></tr>

</table>

<button onclick="exportarCSV()">💾 Exportar a Excel</button>

<script>

let encendido = false;

let intervalo;

let contador = 1;

function toggleSistema() {

encendido = !encendido;

let accion = encendido ? "encender" : "apagar";

fetch("/" + accion)

.then(response => response.text())

.then(data => {

document.getElementById("botonSistema").innerText = encendido ? "Sistema encendido" : "Sistema apagado";

document.getElementById("estado").innerText = encendido ? "🔴 Activo" : "⚪ Apagado";

document.getElementById("estado").style.color = encendido ? "red" : "gray";

document.getElementById("resultado").innerText = data;

document.getElementById("resultado").style.color = "black";

});

if (encendido) {

intervalo = setInterval(() => {

fetch("/medir")

.then(r => r.text())

.then(d => {

document.getElementById("resultado").innerText = d;

let lineas = d.split("\n");

let distancia = lineas[0].split(":")[1].trim().replace("cm", "");

let estado = lineas[1] || "";

let hora = new Date().toLocaleTimeString();

let tabla = document.getElementById("tablaHistorial");

let fila = tabla.insertRow(-1);

fila.insertCell(0).innerText = contador++;

fila.insertCell(1).innerText = distancia;

fila.insertCell(2).innerText = estado;

fila.insertCell(3).innerText = hora;

if (estado.includes("peligrosa")) {

document.getElementById("resultado").style.color = "red";

} else if (estado.includes("segura")) {

document.getElementById("resultado").style.color = "green";

} else {

document.getElementById("resultado").style.color = "black";

}

});

}, 1000);

} else {

clearInterval(intervalo);

document.getElementById("resultado").innerText = "🛑 Sistema apagado. Sin lectura.";

document.getElementById("resultado").style.color = "gray";

}

}

function exportarCSV() {

let tabla = document.getElementById("tablaHistorial");

let filas = tabla.querySelectorAll("tr");

let csv = [];

for (let fila of filas) {

let columnas = fila.querySelectorAll("th, td");

let filaCSV = Array.from(columnas).map(c => `"${c.innerText}"`).join(",");

csv.push(filaCSV);

}

let contenido = csv.join("\n");

let blob = new Blob([contenido], { type: "text/csv;charset=utf-8;" });

let enlace = document.createElement("a");

enlace.href = URL.createObjectURL(blob);

enlace.download = "historial\_lecturas.csv";

enlace.click();

}

</script>

</body>

</html>

)rawliteral";

// 📏 Función de medición

long medirDistancia() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duracion = pulseIn(echoPin, HIGH);

return duracion \* 0.034 / 2;

}

void setup() {

Serial.begin(115200);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n✅ Conectado!");

Serial.print("📲 IP del ESP32: ");

Serial.println(WiFi.localIP());

server.on("/", []() { server.send(200, "text/html", htmlPage); });

server.on("/encender", []() {

sistemaActivo = true;

long distancia = medirDistancia();

String respuesta = "📏 Distancia inicial: " + String(distancia) + " cm\n";

server.send(200, "text/plain", respuesta);

});

server.on("/apagar", []() {

sistemaActivo = false;

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema apagado. Sin lectura.");

});

server.on("/medir", []() {

if (sistemaActivo) {

long distancia = medirDistancia();

String resultado = "📏 Distancia: " + String(distancia) + " cm\n";

if (distancia < 30) {

digitalWrite(ledPin, HIGH);

resultado += "⚠️ ¡Distancia peligrosa! LED encendido.";

} else {

digitalWrite(ledPin, LOW);

resultado += "✅ Zona segura. LED apagado.";

}

server.send(200, "text/plain", resultado);

} else {

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema inactivo");

}

});

server.begin();

}

void loop() {

server.handleClient();

}

13 Grafica-corregir

#include <WiFi.h>

#include <WebServer.h>

// 🔧 Red Wi-Fi

const char\* ssid = "INFINITUMBE3E";

const char\* password = "5365628613";

// 🌐 Servidor Web

WebServer server(80);

// 📍 Pines del sensor y LED

const int trigPin = 5;

const int echoPin = 18;

const int ledPin = 13;

// 🔁 Estado del sistema

bool sistemaActivo = false;

// 🌍 Página web completa con tabla y exportación

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sensor Web</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<style>

body { font-family: sans-serif; text-align: center; margin: 0; padding: 0; background-color: #f5f5f5; }

h2 { margin-top: 30px; color: #333; }

button {

padding: 15px 25px;

font-size: 18px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 8px;

margin-top: 15px;

width: 80%;

}

#estado { font-size: 20px; margin-top: 10px; font-weight: bold; }

#resultado { font-size: 18px; margin-top: 10px; white-space: pre-line; }

table {

margin: 20px auto;

border-collapse: collapse;

width: 95%;

}

th, td {

border: 1px solid #ccc;

padding: 8px;

font-size: 16px;

text-align: center;

}

th {

background-color: #e0e0e0;

}

</style>

</head>

<body>

<h2>📲 Sistema de Medición</h2>

<button id="botonSistema" onclick="toggleSistema()">Sistema apagado</button>

<p id="estado">Sistema inactivo</p>

<p id="resultado">Sin lectura</p>

<h3>📊 Historial de Lecturas</h3>

<table id="tablaHistorial">

<tr><th>#</th><th>Distancia (cm)</th><th>Estado</th><th>Hora</th></tr>

</table>

<button onclick="exportarCSV()">💾 Exportar a Excel</button>

<h3>📈 Gráfica de Distancia</h3>

<canvas id="graficaDistancia" width="95%" height="300"></canvas>

<script>

let encendido = false;

let intervalo;

let contador = 1;

let grafica;

let datosGrafica = {

labels: [],

datasets: [{

label: "Distancia (cm)",

data: [],

borderColor: "blue",

backgroundColor: "rgba(0,0,255,0.1)",

fill: true,

tension: 0.3

}]

};

window.onload = function() {

let ctx = document.getElementById("graficaDistancia").getContext("2d");

grafica = new Chart(ctx, {

type: 'line',

data: datosGrafica,

options: {

responsive: true,

scales: {

y: { beginAtZero: true }

}

}

});

};

function toggleSistema() {

encendido = !encendido;

let accion = encendido ? "encender" : "apagar";

fetch("/" + accion)

.then(response => response.text())

.then(data => {

document.getElementById("botonSistema").innerText = encendido ? "Sistema encendido" : "Sistema apagado";

document.getElementById("estado").innerText = encendido ? "🔴 Activo" : "⚪ Apagado";

document.getElementById("estado").style.color = encendido ? "red" : "gray";

document.getElementById("resultado").innerText = data;

document.getElementById("resultado").style.color = "black";

});

if (encendido) {

intervalo = setInterval(() => {

fetch("/medir")

.then(r => r.text())

.then(d => {

document.getElementById("resultado").innerText = d;

let lineas = d.split("\n");

let distancia = lineas[0].split(":")[1].trim().replace("cm", "");

let estado = lineas[1] || "";

let hora = new Date().toLocaleTimeString();

let tabla = document.getElementById("tablaHistorial");

let fila = tabla.insertRow(-1);

fila.insertCell(0).innerText = contador++;

fila.insertCell(1).innerText = distancia;

fila.insertCell(2).innerText = estado;

fila.insertCell(3).innerText = hora;

// 📈 Actualizar la gráfica

datosGrafica.labels.push(hora);

datosGrafica.datasets[0].data.push(Number(distancia));

if (datosGrafica.labels.length > 20) {

datosGrafica.labels.shift(); // Elimina el más antiguo

datosGrafica.datasets[0].data.shift(); // Mantiene 20 valores

}

grafica.update(); // Redibuja la curva

if (estado.includes("peligrosa")) {

document.getElementById("resultado").style.color = "red";

} else if (estado.includes("segura")) {

document.getElementById("resultado").style.color = "green";

} else {

document.getElementById("resultado").style.color = "black";

}

});

}, 1000);

} else {

clearInterval(intervalo);

document.getElementById("resultado").innerText = "🛑 Sistema apagado. Sin lectura.";

document.getElementById("resultado").style.color = "gray";

}

}

function exportarCSV() {

let tabla = document.getElementById("tablaHistorial");

let filas = tabla.querySelectorAll("tr");

let csv = [];

for (let fila of filas) {

let columnas = fila.querySelectorAll("th, td");

let filaCSV = Array.from(columnas).map(c => `"${c.innerText}"`).join(",");

csv.push(filaCSV);

}

let contenido = csv.join("\n");

let blob = new Blob([contenido], { type: "text/csv;charset=utf-8;" });

let enlace = document.createElement("a");

enlace.href = URL.createObjectURL(blob);

enlace.download = "historial\_lecturas.csv";

enlace.click();

}

</script>

</body>

</html>

)rawliteral";

// 📏 Función de medición

long medirDistancia() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duracion = pulseIn(echoPin, HIGH);

return duracion \* 0.034 / 2;

}

void setup() {

Serial.begin(115200);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

WiFi.begin(ssid, password);

Serial.print("Conectando a WiFi...");

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

delay(500);

Serial.print(".");

}

Serial.println("\n✅ Conectado!");

Serial.print("📲 IP del ESP32: ");

Serial.println(WiFi.localIP());

server.on("/", []() { server.send(200, "text/html", htmlPage); });

server.on("/encender", []() {

sistemaActivo = true;

long distancia = medirDistancia();

String respuesta = "📏 Distancia inicial: " + String(distancia) + " cm\n";

server.send(200, "text/plain", respuesta);

});

server.on("/apagar", []() {

sistemaActivo = false;

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema apagado. Sin lectura.");

});

server.on("/medir", []() {

if (sistemaActivo) {

long distancia = medirDistancia();

String resultado = "📏 Distancia: " + String(distancia) + " cm\n";

if (distancia < 30) {

digitalWrite(ledPin, HIGH);

resultado += "⚠️ ¡Distancia peligrosa! LED encendido.";

} else {

digitalWrite(ledPin, LOW);

resultado += "✅ Zona segura. LED apagado.";

}

server.send(200, "text/plain", resultado);

} else {

digitalWrite(ledPin, LOW);

server.send(200, "text/plain", "🛑 Sistema inactivo");

}

});

server.begin();

}

void loop() {

server.handleClient();

}

otras

/\*

TB6600-Stepper-Motor-Driver

made on 15 Dec 2020

by Amir Mohammad Shojaee @ Electropeak

Home

\*/

#define dirPin1 33

#define stepPin1 13

#define dirPin2 32

#define stepPin2 14

void setup() {

// Declare pins as output:

pinMode(stepPin1, OUTPUT);

pinMode(dirPin1, OUTPUT);

pinMode(stepPin2, OUTPUT);

pinMode(dirPin2, OUTPUT);

// Set the spinning direction CW/CCW:

digitalWrite(dirPin1, HIGH);

digitalWrite(dirPin2, LOW);

}

void loop() {

// These four lines result in 1 step:

digitalWrite(stepPin1, HIGH);

digitalWrite(stepPin2, HIGH);

delayMicroseconds(500);

digitalWrite(stepPin1, LOW);

digitalWrite(stepPin2, LOW);

delayMicroseconds(500);

}