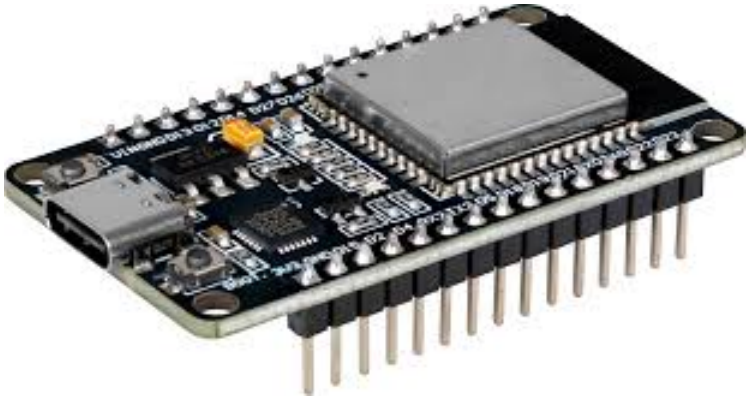
Proyecto de Control de Puertas con ESP32

Sistemas Embebidos

Máster Universitario de Informática Industrial y Robótica.

Autor: Álvaro Viña Pérez
Fecha: 4 de noviembre de 2025



Universidad de La Laguna

Índice

1. Descripción general

El proyecto consiste en el desarrollo de un sistema de control de dos puertas mediante una placa **ESP32**, integrando sensores, servomotores, un sensor de temperatura y humedad **DHT11**, un indicador luminoso RGB y un **buzzer** acústico.

El sistema se conecta a una red WiFi local y actúa como un *servidor web*, permitiendo al usuario interactuar con las puertas y consultar los datos de los sensores desde cualquier dispositivo mediante un navegador.

Además, la ESP32 incorpora un modo de bajo consumo (*deep sleep*) que puede activarse remotamente desde la interfaz web, despertando posteriormente mediante el **sensor táctil integrado en el pin GPIO 4**.

2. Funcionamiento del sistema

Al arrancar, la ESP32 establece la conexión WiFi y habilita el servidor web. Una vez conectado, el usuario accede a una página que muestra:

- El estado de cada puerta (abierta o cerrada).
- Los valores de temperatura y humedad obtenidos del sensor DHT11.
- El color actual del LED RGB, que indica el estado global del sistema.

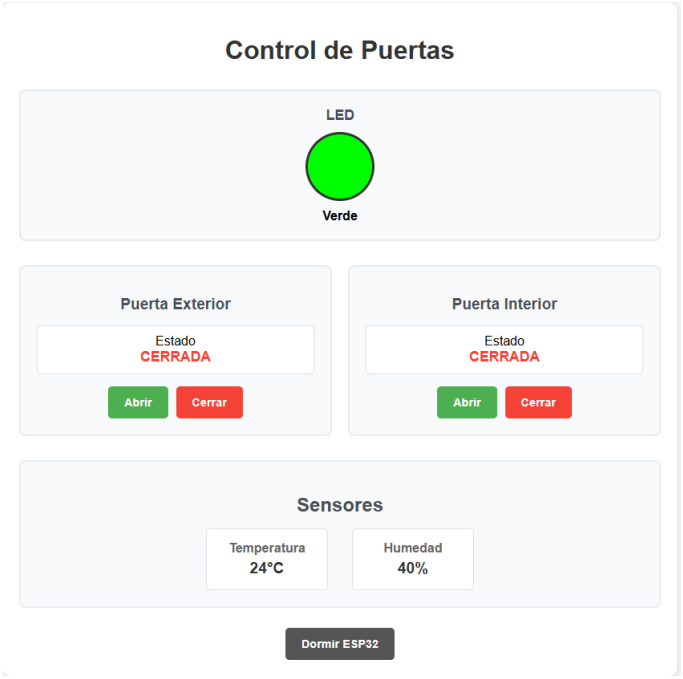


Figura 1: Interfaz web del sistema de control

El sistema incluye dos servomotores encargados de accionar las puertas, y dos finales de carrera (simulados mediante pulsadores) que determinan el estado de cada una. Para evitar conflictos, las puertas no pueden abrirse simultáneamente:

- Si la puerta exterior está abierta, la interior no puede abrirse.

- Si la puerta interior está abierta, la exterior no puede abrirse.

El LED RGB refleja visualmente el estado general:

- **Verde:** ambas puertas cerradas.
- **Rojo:** ambas puertas abiertas (error).
- **Azul:** una puerta abierta.
- **Blanco o apagado:** estados transitorios o inactivos.

El buzzer se activa únicamente cuando se produce un error de seguridad (por ejemplo, intento de abrir ambas puertas). Cada 2 segundos el sistema actualiza los valores del sensor DHT11 y verifica el estado de las puertas mediante un temporizador no bloqueante.

Cuando se activa el modo *deep sleep*, el sistema apaga el WiFi, deshabilita las interrupciones y coloca los servos y el LED en un estado seguro. El despertar se produce al tocar el pin táctil T0 (GPIO 4).

3. Código fuente

```

#include <ESP32Servo.h>
#include <DHT11.h>
#include <WiFi.h>
#include "esp_sleep.h"
#include "driver/touch_pad.h"


// === Prototipos ===
void ledVerde();
void ledBlanco();
void ledAzul();
void ledRojo();
void ledApagado();
void leerSensor();
void EstadoPuerta1();
void EstadoPuerta2();
void controlPuertas();
void beep(int frecuencia, int duracion);
void manejarControl(WiFiClient &client, String request);
void manejarEstados(WiFiClient &client);
void manejarSleep(WiFiClient &client);
void goToSleep();

// === WIFI ===
const char* ssid = "OnePlus3";
const char* password = "wifioneplus7T";
WiFiServer server(80);

// === DHT11 ===
#define DHT_PIN 5
DHT11 dht(DHT_PIN);
int temperatura;
int humedad;

// === Timer no bloqueante ===
unsigned long t0 = 0;
unsigned long intervalo = 2000;

// === RGB ===
const int PIN_R = 27, PIN_G = 26, PIN_B = 25;
String estadoLED = "apagado";

// === Servos / puertas ===
Servo puerta1;
Servo puerta2;
int servo1 = 17;
int servo2 = 16;
const int ANG_CERR = 0;
```

```
50 const int ANG_ABR  = 90;
51 volatile bool puertaCerrada1 = true;
52 volatile bool puertaCerrada2 = true;
53 volatile bool abrirPuerta1 = false;
54 volatile bool abrirPuerta2 = false;
55 volatile bool error = false;
56
57 // === Botones / final de carrera ===
58 const byte boton1 = 13;
59 const byte boton2 = 14;
60 unsigned long debounce = 1000;
61 volatile unsigned long ULT_ISR1 = 0;
62 volatile unsigned long ULT_ISR2 = 0;
63
64 // === Buzzer ===
65 int buzzer = 23;
66
67 // === Touchpad (despertar) ===
68 // T0 = GPIO 4 (tocar para despertar del deep sleep)
69 #define TOUCH_CH T0
70 #define TOUCH_THRESHOLD 200
71
72
73 // ---------------- HTML ----------------
74 String htmlTest() {
75   String html = R"HTML(
76 <!DOCTYPE html>
77 <html lang="es">
78 <head>
79   <meta charset="utf-8">
80   <meta name="viewport" content="width=device-width,initial-scale
      =1">
81   <title>Control de Puertas ESP32</title>
82   <style>
83     body { font-family: Arial, sans-serif; margin: 20px;
        background-color: #f5f5f5; }
84     .container { max-width: 800px; margin: 0 auto; background:
        white; padding: 20px; border-radius: 10px; box-shadow: 0 2
        px 10px rgba(0,0,0,0.1); }
85     h1 { text-align: center; color: #333; margin-bottom: 30px; }
86     .puertas-container { display: flex; justify-content: space-
        between; gap: 20px; margin-bottom: 30px; }
87     .puerta { background: #f8f9fa; padding: 20px; border-radius:
        8px; border: 2px solid #e9ecef; flex: 1; text-align: center
        ; }
88     .puerta h2 { color: #495057; margin-bottom: 15px; font-size:
        1.2em; }
89     .estado-puerta { background: white; padding: 10px; border-
        radius: 5px; margin: 15px 0; border: 1px solid #dee2e6; }
90     .estado { font-weight: bold; font-size: 1.1em; }
91     .abierta { color: #4CAF50; }
```

```
    .cerrada { color: #f44336; }
    .botones { display: flex; gap: 10px; justify-content: center; margin-top: 15px; flex-wrap: wrap; }
    button { padding: 12px 20px; border: none; border-radius: 5px; cursor: pointer; font-size: 0.9em; font-weight: bold; }
    .abrir { background: #4CAF50; color: white; }
    .abrir:hover { background: #45a049; }
    .cerrar { background: #f44336; color: white; }
    .cerrar:hover { background: #da190b; }
    .sleep { background: #555; color: white; }
    .sleep:hover { background: #444; }
    .led-section { background: #f8f9fa; padding: 20px; border-radius: 8px; border: 2px solid #e9ecef; text-align: center; margin-bottom: 30px; }
    .led-label { font-weight: bold; margin-bottom: 10px; color: #495057; font-size: 1.1em; }
    .led-indicator { width: 80px; height: 80px; border-radius: 50%; border: 3px solid #333; margin: 10px auto; transition: background-color 0.3s ease; }
    .led-text { font-weight: bold; margin-top: 5px; }
    .sensor-section { background: #f8f9fa; padding: 20px; border-radius: 8px; border: 2px solid #e9ecef; text-align: center; }
    .sensor-section h2 { color: #495057; margin-bottom: 15px; }
    .sensor-data { display: flex; justify-content: center; gap: 30px; }
    .sensor-item { background: white; padding: 15px; border-radius: 5px; border: 1px solid #dee2e6; min-width: 120px; }
    .sensor-item .label { font-weight: bold; color: #666; margin-bottom: 5px; }
    .sensor-item .value { font-size: 1.2em; font-weight: bold; color: #333; }
    .debug { display: none; }
  </style>
</head>
<body>
  <div class="container">
    <h1>Control de Puertas</h1>

    <!-- Sección LED -->
    <div class="led-section">
      <div class="led-label">LED</div>
      <div id="ledIndicator" class="led-indicator" style="background-color: #000000;"></div>
      <div id="ledText" class="led-text">Apagado</div>
    </div>

    <!-- Contenedor de puertas -->
    <div class="puertas-container">
      <!-- Puerta 1 -->
      <div class="puerta">
```

```
129         <h2>Puerta Exterior</h2>
130         <div class="estado-puerta">
131           <div class="label">Estado</div>
132           <div class="estado" id="estado1">Cargando...</div>
133         </div>
134         <div class="botones">
135           <button class="abrir" onclick="controlPuerta(1, 'abrir'
                )">Abrir</button>
136           <button class="cerrar" onclick="controlPuerta(1, '
                cerrar')">Cerrar</button>
137         </div>
138       </div>
139
140       <!-- Puerta 2 -->
141       <div class="puerta">
142         <h2>Puerta Interior</h2>
143         <div class="estado-puerta">
144           <div class="label">Estado</div>
145           <div class="estado" id="estado2">Cargando...</div>
146         </div>
147         <div class="botones">
148           <button class="abrir" onclick="controlPuerta(2, 'abrir'
                )">Abrir</button>
149           <button class="cerrar" onclick="controlPuerta(2, '
                cerrar')">Cerrar</button>
150         </div>
151       </div>
152     </div>
153
154     <!-- Sección Sensores -->
155     <div class="sensor-section">
156       <h2>Sensores</h2>
157       <div class="sensor-data">
158         <div class="sensor-item">
159           <div class="label">Temperatura</div>
160           <div class="value" id="temp">--</div>
161         </div>
162         <div class="sensor-item">
163           <div class="label">Humedad</div>
164           <div class="value" id="hum">--</div>
165         </div>
166       </div>
167     </div>
168
169     <!-- Botón de dormir -->
170     <div style="text-align:center; margin-top:25px;">
171       <button class="sleep" onclick="dormir()">Dormir ESP32</
            button>
172     </div>
173
174     <div class="debug" id="debugInfo"></div>
```

```
  </div>

  <script>
    function controlPuerta(numero, accion) {
      fetch('/control?puerta=' + numero + '&accion=' + accion)
        .then(r => r.text())
        .then(_ => actualizarEstados())
        .catch(console.error);
    }

    function dormir() {
      fetch('/sleep')
        .then(r => r.text())
        .then(txt => { alert(txt); })
        .catch(console.error);
    }

    function actualizarEstados() {
      fetch('/estados')
        .then(r => { if (!r.ok) throw new Error('HTTP ' + r.
           status); return r.text(); })
        .then(text => {
          const data = JSON.parse(text);
          document.getElementById('estado1').textContent = data.
             puerta1 ? 'CERRADA' : 'ABIERTA';
          document.getElementById('estado2').textContent = data.
             puerta2 ? 'CERRADA' : 'ABIERTA';
          document.getElementById('estado1').className = 'estado
             ' + (data.puerta1 ? 'cerrada' : 'abierta');
          document.getElementById('estado2').className = 'estado
             ' + (data.puerta2 ? 'cerrada' : 'abierta');
          document.getElementById('temp').textContent = (data.
             temperatura !== 'undefined' ? data.temperatura +
             '&#176;C' : '--');
          document.getElementById('hum').textContent = (data.
             humedad !== undefined ? data.humedad + '%' : '--');
          const ledIndicator = document.getElementById('
             ledIndicator');
          const ledText = document.getElementById('ledText');
          if (data.led_color) ledIndicator.style.backgroundColor
             = data.led_color;
          if (data.led_text)  ledText.textContent = data.led_text
             ;
        })
        .catch(console.error);
    }

    setInterval(actualizarEstados, 3000);
    actualizarEstados();
  </script>
</body>
```

```
</html>
)HTML";
  return html;
}

// ---------------- SETUP ----------------
void setup() {
  Serial.begin(115200);
  while(!Serial){;}

  // WiFi
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(500); Serial.
     print("."); }
  Serial.println("\nWiFi connected.");
  Serial.print("IP address: "); Serial.println(WiFi.localIP());
  server.begin();

  // LED RGB
  pinMode(PIN_R,OUTPUT);
  pinMode(PIN_G,OUTPUT);
  pinMode(PIN_B,OUTPUT);

  // Buzzer
  pinMode(buzzer,OUTPUT);

  // Servos
  puerta1.attach(servo1);
  puerta2.attach(servo2);
  puertaCerrada1 = true;
  puertaCerrada2 = true;
  puerta1.write(ANG_CERR);
  puerta2.write(ANG_CERR);

  // Botones / finales
  pinMode(boton1,INPUT_PULLUP);
  pinMode(boton2,INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(boton1),EstadoPuerta1,
     FALLING);
  attachInterrupt(digitalPinToInterrupt(boton2),EstadoPuerta2,
     FALLING);
}


void loop() {
  WiFiClient client = server.available();

  if (WiFi.status() == WL_CONNECTED){
    if (client) {
```

```
263       String request = client.readStringUntil('\r');
264       client.flush();
265
266       if (request.indexOf("GET / ") != -1) {
267         client.println("HTTP/1.1 200 OK");
268         client.println("Content-Type: text/html; charset=utf-8");
269         client.println("Connection: close");
270         client.println();
271         client.println(htmlTest());
272       }
273       else if (request.indexOf("/control") != -1) {
274         manejarControl(client, request);
275       }
276       else if (request.indexOf("/estados") != -1) {
277         manejarEstados(client);
278       }
279       else if (request.indexOf("/sleep") != -1) {
280         manejarSleep(client);
281       }
282
283       client.stop();
284     }
285
286     unsigned long t_actual = millis();
287     if(t_actual - t0 > intervalo){
288       leerSensor();
289       controlPuertas();
290       t0 = t_actual;
291     }
292   }else{
293     Serial.print("Conexión WIFI perdida, revise la conexión");
294     while (WiFi.status() != WL_CONNECTED) { delay(500); Serial.
          print("."); }
295     Serial.println("\nWiFi reconectado.");
296
297   }
298 }
299
300
301 void manejarControl(WiFiClient &client, String request) {
302   int puerta = 0;
303   String accion = "";
304   String respuesta = "Comando ejecutado";
305
306   if (request.indexOf("puerta=1") != -1) puerta = 1;
307   else if (request.indexOf("puerta=2") != -1) puerta = 2;
308
309   if (request.indexOf("accion=abrir") != -1) accion = "abrir";
310   else if (request.indexOf("accion=cerrar") != -1) accion = "
        cerrar";
311
```

```
312   if (puerta == 1 && accion == "abrir") {
313     if(puertaCerrada2){
314       abrirPuerta1 = true;
315       respuesta = "Abriendo puerta exterior";
316     }else{
317       respuesta = "ERROR: No se puede abrir la puerta 1 - Puerta
          2 abierta";
318       error = true;
319     }
320   }
321   else if (puerta == 1 && accion == "cerrar") {
322     abrirPuerta1 = false;
323     respuesta = "Cerrando puerta exterior";
324   }
325   else if (puerta == 2 && accion == "abrir") {
326     if(puertaCerrada1){
327       abrirPuerta2 = true;
328       respuesta = "Abriendo puerta interior";
329     }else{
330       respuesta = "ERROR: No se puede abrir la puerta 2 - Puerta
          1 abierta";
331       error = true;
332     }
333   }
334   else if (puerta == 2 && accion == "cerrar") {
335     abrirPuerta2 = false;
336     respuesta = "Cerrando puerta interior";
337   }
338
339   client.println("HTTP/1.1 200 OK");
340   client.println("Content-Type: text/plain");
341   client.println("Connection: close");
342   client.println();
343   client.println(respuesta);
344   Serial.println(respuesta);
345 }
346
347 void manejarEstados(WiFiClient &client) {
348   client.println("HTTP/1.1 200 OK");
349   client.println("Content-Type: application/json");
350   client.println("Connection: close");
351   client.println();
352
353   String json = "{";
354   json += "\"puerta1\":"; json += puertaCerrada1 ? "true" : "
        false";
355   json += ",\"puerta2\":"; json += puertaCerrada2 ? "true" : "
        false";
356   json += ",\"temperatura\":"; json += String(temperatura);
357   json += ",\"humedad\":"; json += String(humedad);
358   json += ",\"led_color\":\"";
```

```
359
360   if (estadoLED == "Verde")       json += "#00ff00";
361   else if (estadoLED == "Rojo")   json += "#ff0000";
362   else if (estadoLED == "Azul")   json += "#0000ff";
363   else if (estadoLED == "Blanco")json += "#ffffff";
364   else                            json += "#000000";
365
366   json += "\",\"led_text\":\"";
367   json += estadoLED;
368   json += "\"}";
369
370   client.println(json);
371 }
372
373 void manejarSleep(WiFiClient &client){
374   client.println("HTTP/1.1 200 OK");
375   client.println("Content-Type: text/plain");
376   client.println("Connection: close");
377   client.println();
378   client.println("Entrando en deep sleep. Toca el pin táctil GPIO
        4 (T0) para despertar.");
379   client.flush();
380
381   delay(150);
382   goToSleep();
383 }
384
385
386 void controlPuertas() {
387   if ((abrirPuerta1 && puertaCerrada1) || error) {
388     if (puertaCerrada2) {
389       puerta1.write(ANG_ABR);
390     } else {
391       beep(1000, 250);
392       error = false;
393     }
394   }
395   else if (!abrirPuerta1 && !puertaCerrada1) {
396     puerta1.write(ANG_CERR);
397   }
398
399   if ((abrirPuerta2 && puertaCerrada2) || error) {
400     if (puertaCerrada1) {
401       puerta2.write(ANG_ABR);
402     } else {
403       beep(1000, 250);
404       error = false;
405     }
406   }
407   else if (!abrirPuerta2 && !puertaCerrada2) {
408     puerta2.write(ANG_CERR);
```

```
409   }
410
411   // LED + Buzzer dependen únicamente de las puertas
412   if (puertaCerrada1 && puertaCerrada2) {
413     ledVerde(); noTone(buzzer);
414   } else if (!puertaCerrada1 && !puertaCerrada2) {
415     tone(buzzer,2500); ledRojo();
416   } else {
417     ledAzul(); noTone(buzzer);
418   }
419 }
420
421 void ledVerde(){  analogWrite(PIN_R,0);   analogWrite(PIN_G,255);
      analogWrite(PIN_B,0);   estadoLED = "Verde"; }
422 void ledRojo(){   analogWrite(PIN_R,255); analogWrite(PIN_G,0);
       analogWrite(PIN_B,0);   estadoLED = "Rojo"; }
423 void ledApagado(){analogWrite(PIN_R,0);   analogWrite(PIN_G,0);
       analogWrite(PIN_B,0);   estadoLED = "Apagado"; }
424 void ledBlanco(){ analogWrite(PIN_R,255); analogWrite(PIN_G,255);
      analogWrite(PIN_B,255); estadoLED = "Blanco"; }
425 void ledAzul(){   analogWrite(PIN_R,0);   analogWrite(PIN_G,0);
       analogWrite(PIN_B,255); estadoLED = "Azul"; }
426
427 // ---------------- Sensores ----------------
428 void leerSensor(){
429   temperatura = dht.readTemperature();
430   humedad = dht.readHumidity();
431 }
432
433 void EstadoPuerta1(){
434   unsigned long t = millis();
435   if(t - ULT_ISR1 > debounce){
436     puertaCerrada1 = !puertaCerrada1;
437     ULT_ISR1 = t;
438   }
439 }
440 void EstadoPuerta2(){
441   unsigned long t = millis();
442   if(t - ULT_ISR2 > debounce){
443     puertaCerrada2 = !puertaCerrada2;
444     ULT_ISR2 = t;
445   }
446 }
447
448
449 void beep(int frecuencia, int duracion) {
450   tone(buzzer, frecuencia);
451   delayMicroseconds(duracion * 1000);
452   noTone(buzzer);
453 }
454
```

```
455
456 void goToSleep(){
457   // Estado seguro
458   noTone(buzzer);
459   ledApagado();
460   puerta1.write(ANG_CERR);
461   puerta2.write(ANG_CERR);
462   delay(150);
463   // Quitar interrupciones de botones para evitar ruido al dormir
464   detachInterrupt(digitalPinToInterrupt(boton1));
465   detachInterrupt(digitalPinToInterrupt(boton2));
466
467   // Apagar WiFi para bajar consumo
468   WiFi.disconnect(true);
469   WiFi.mode(WIFI_OFF);
470
471   // Configurar touchpad
472   int THRESHOLD = touchRead(TOUCH_CH) - TOUCH_THRESHOLD;
473   touchAttachInterrupt(TOUCH_CH, nullptr, THRESHOLD);
474   esp_sleep_enable_touchpad_wakeup();
475
476   Serial.println("Deep sleep activado. Toca el pin táctil GPIO 4
        (T0) para despertar.");
477   delay(50);
478   esp_deep_sleep_start();
479 }
```

4. Explicación del código

El programa se estructura en bloques bien diferenciados:

1. Inclusión de librerías y variables

Se emplean librerías para controlar los servos, la red WiFi, el sensor DHT11 y el modo de suspensión. Se definen pines, variables globales y banderas de control para los estados de las puertas y sensores.

2. Servidor web y HTML integrado

La función `htmlTest()` contiene el código HTML, CSS y JavaScript embebido que define la página web servida por la ESP32. A través de esta página el usuario puede:

- Abrir o cerrar cada puerta.
- Consultar los estados actuales.
- Activar el modo de suspensión.

Las peticiones HTTP del cliente se gestionan en el bucle principal, redirigiendo según la URL a funciones específicas: `/control`, `/estados` y `/sleep`.

3. Función `setup()`

Inicializa los pines, conecta la red WiFi, configura el servidor, adjunta los servos a sus pines y asocia las interrupciones de los finales de carrera. También establece los estados iniciales de las puertas y el LED.

4. Bucle principal `loop()`

El bucle comprueba continuamente si hay un cliente conectado. Según la solicitud recibida, se ejecuta una de las siguientes funciones:

- `manejarControl()`: interpreta los comandos de apertura o cierre.
- `manejarEstados()`: devuelve el estado del sistema en formato JSON.
- `manejarSleep()`: activa el modo de bajo consumo.

Cada 2 segundos se leen los sensores mediante `leerSensor()` y se actualiza la lógica de seguridad con `controlPuertas()`.

5. Control de puertas

La función `controlPuertas()` evalúa el estado actual y determina si los servos deben moverse, generando además las señales visuales y acústicas correspondientes. Si se detecta una condición no segura (por ejemplo, apertura simultánea), se genera un aviso sonoro y se bloquea la acción.

6. Gestión del modo Sleep

La función `goToSleep()` asegura un estado estable antes de dormir:

- Cierra ambas puertas.
- Apaga el LED y el buzzer.
- Desactiva WiFi e interrupciones.

Posteriormente configura el *touchpad* para permitir el despertar por contacto táctil.

5. Conclusiones

El sistema desarrollado demuestra la versatilidad de la plataforma ESP32 al integrar control remoto, sensores ambientales, actuadores y modos de bajo consumo dentro de un único microcontrolador.

El uso de un servidor web embebido facilita la interacción sin necesidad de hardware adicional, permitiendo un control visual, intuitivo y multiplataforma.

Además, la lógica implementada garantiza seguridad en el funcionamiento de las puertas, evitando aperturas simultáneas y proporcionando una señalización clara mediante LED y buzzer.

Como posibles mejoras futuras se plantea:

- Añadir almacenamiento en EEPROM para guardar estados.
- Integrar un sistema de notificaciones mediante el servidor Web o registro de eventos.
- Comunicación mediante MQTT entre varias placas para el control de las puertas.
- Uso de infrarrojos para la apertura y cierre de las puertas.