Proyecto Final

Portón Automatizado

Integrantes:

- Díaz Francisco
- González Lautaro
- Peirone Santiago
- Zarza Valentín

Automatización de Portón con ESP8266

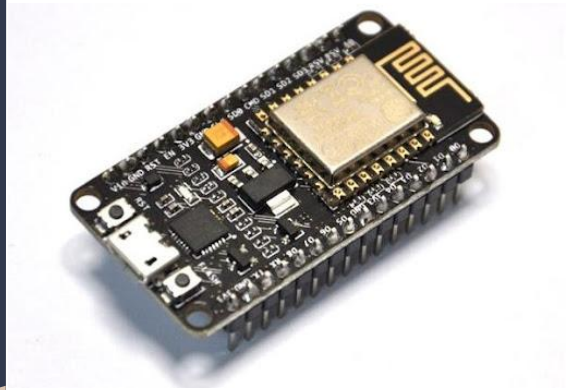
Descripción Breve:

Control de un portón automatizado utilizando ESP8266, servo motor y una aplicación móvil desarrollada con MIT App Inventor.

MIT APP INVENTOR

Componentes Principales

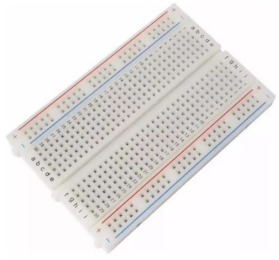
NODEMCU ESP8266



ARDUINO UNO

SERVO MOTOR

PROTOBOARD



Función de los componentes

NodeMCU ESP8266 

- Esta placa será nuestro maestro, permitiéndonos conectarnos al Wi-Fi e interactuando con ella de forma rer

Arduino Uno 

- El será nuestro esclavo, facilitandole los 5 volteos necesarios al servo motor y prendiendo un led amarillo simulando una luz calida dentro de la maqueta.

Servo motor 

- Como su nombre indica, es el motor que interactúa con el portón, abriendolo y cerrandolo.

Protoboard 

- Nos permite las conexiones para los diferentes componentes a través de los cables y sus entradas

Programación del ESP8266 (Maestro)

```
#include <ESP8266WiFi.h>
#include <Servo.h>

// Configuración Wi-Fi
const char* ssid = "red";        // nombre de tu red Wi-Fi
const char* password = "contraseña de la red"; // contraseña Wi-Fi

WiFiServer server(80);
Servo myServo;
int currentAngle = 0; // Variable para almacenar la posición actual del servo

// Función para mover el servo lentamente
void moveServoSlowly(int startAngle, int endAngle, int stepDelay) {
  if (startAngle < endAngle) {
    for (int pos = startAngle; pos <= endAngle; pos++) {
      myServo.write(pos);
      delay(stepDelay);
    }
  } else {
    for (int pos = startAngle; pos >= endAngle; pos--) {
      myServo.write(pos);
      delay(stepDelay);
    }
  }
  currentAngle = endAngle; // Actualiza la posición actual después de mover el servo
}

void setup() {
  Serial.begin(115200);
  myServo.attach(D4); // Pin GPIO2 (D4 en NodeMCU)
  myServo.write(0);   // Posición inicial (cerrado)
  currentAngle = 0;  // Inicializa la posición actual

  // Conexión a Wi-Fi
  WiFi.begin(ssid, password);
  Serial.println("Conectando al Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
```

Programación del ESP8266 (Maestro)

```
Serial.println("\nConexión Wi-Fi establecida.");
  Serial.println(WiFi.localIP()); // Imprime la IP asignada

  server.begin();
}

void loop() {
  WiFiClient client = server.available(); // Escucha conexiones entrantes

  if (client) {
    Serial.println("Cliente conectado.");
    String request = client.readStringUntil('\r');
    Serial.println("Petición recibida: " + request);
    client.flush();

    // Control del servo
    if (request.indexOf("/OPEN") != -1) {
      if (currentAngle != 180) { // Solo mueve si la posición actual no es 180°
        Serial.println("Abriendo portón lentamente...");
        moveServoSlowly(currentAngle, 180, 15); // Mueve de 0° a 180° con un retraso de 15ms por
paso
      }
    }
    if (request.indexOf("/CLOSE") != -1) {
      if (currentAngle != 0) { // Solo mueve si la posición actual no es 0°
        Serial.println("Cerrando portón lentamente...");
        moveServoSlowly(currentAngle, 0, 15); // Mueve de 180° a 0° con un retraso de 15ms por paso
      }
    }

    // Respuesta al cliente
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/plain");
    client.println("Connection: close"); // Cierra la conexión después de la respuesta
    client.println();
    client.println("OK"); // Respuesta breve y clara
  }
}
```

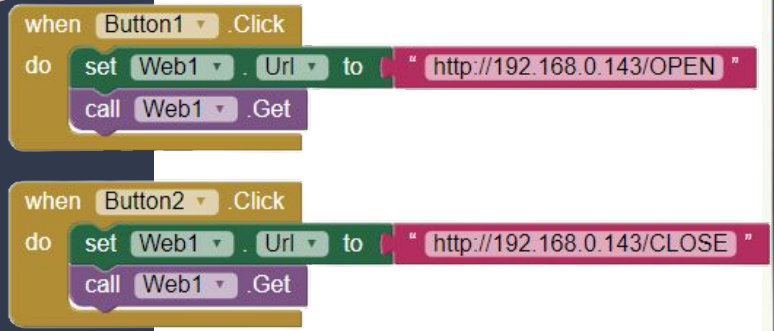
Programación del Arduino Uno (Esclavo)

```
void setup() {
  // Configurar el pin 13 como salida
  pinMode(13, OUTPUT);

  // Encender el LED
  digitalWrite(13, HIGH);
}

void loop() {
  // No es necesario hacer nada aquí
  // El LED permanece encendido porque
lo hemos configurado en el setup
}
```

MIT app Inventor

Programación en bloque

- Creación de botones y Labels (estética).
- Configuración del componente Web para enviar solicitudes HTTP.



Sinopsis

La idea de esto no es solo el uso de un portón automático, sino que el verdadero objetivos es el accionar de motores de forma remota.

Esta placa al permitirnos a través de conexiones Wi-Fi activar o desactivar algo, nos abre un nuevo mundo de posibilidades, tanto para la zona industrial como para la personal.

Automatización del hogar (domótica):

Este sistema puede adaptarse para controlar persianas, cortinas, puertas automáticas o incluso sistemas de ventilación y calefacción.

Aplicaciones industriales:

En entornos industriales, el control remoto de motores permite manejar maquinaria pesada o sistemas de transporte, optimizando recursos y mejorando la seguridad de los operarios.

Accesibilidad:

Las personas con movilidad reducida pueden beneficiarse de sistemas automatizados que les permitan operar dispositivos o máquinas sin necesidad de esfuerzo físico.

Bibliografía

Arduino IDE

- Página oficial de Arduino para descargar el software y acceder a la documentación.
- Enlace: https://www.arduino.cc/en/software

MIT App Inventor

- Herramienta para la creación de aplicaciones móviles utilizadas en este proyecto.
- Enlace: https://appinventor.mit.edu/

Librería Servo para Arduino

- Documentación oficial sobre la librería Servo.h utilizada para controlar el motor.
- Enlace: https://www.arduino.cc/en/reference/servo

ESP8266WiFi Library

- Documentación de la librería usada para la conexión Wi-Fi del ESP8266.
- Enlace: https://arduino-esp8266.readthedocs.io/en/latest/

NodeMCU ESP8266

- Información técnica sobre el microcontrolador utilizado.
- Enlace: https://www.nodemcu.com/

Mención Honorífica

- https://chatgpt.com

Fin