ROBOT MÓVIL AUTÓNOMO CONTROLADO POR WEB

Robótica I

Jeremy Valera Bedregal **2020204070**
Ever Quispe Ramos **2019204015**
Christian Rider Ajrota **2019204016**
Giovanni Callata Anccori **2021204045**

Docente:
TORRES CARPIO, LUIS MANUEL

26 de diciembre de 2024

Índice

1. Resumen

Este proyecto presenta el diseño e implementación de un robot móvil autónomo controlado mediante una interfaz web utilizando un microcontrolador Wemos D1 Mini. El sistema permite el control independiente de dos motores de corriente continua (DC) para dirección y aceleración progresiva, integrando funcionalidades innovadoras como un interruptor lógico y reducción eficiente de voltaje de 9V a 3.7V. Se desarrolló una interfaz web responsiva para la gestión remota y se realizaron pruebas funcionales exitosas en una simulación basada en Tinkercad. Este trabajo destaca por su innovación en el diseño, funcionalidad y aplicación de tecnologías modernas.

2. Introducción

El control eficiente de robots móviles es un desafío que combina hardware y software. Este proyecto busca desarrollar un sistema que permita gestionar de manera independiente dos motores DC, uno para la dirección y otro para el arranque, utilizando un enfoque innovador que incluye un interruptor lógico y un convertidor DC-DC para optimización de energía.

2.1. Objetivo del Proyecto

Desarrollar un robot móvil controlado remotamente a través de una interfaz web, utilizando el microcontrolador Wemos D1 Mini, capaz de realizar movimientos precisos y responder a comandos en tiempo real.

3. Estado del Arte

El desarrollo de robots móviles autónomos ha crecido significativamente en los últimos años, impulsado por avances en microcontroladores, sensores y conectividad inalámbrica. Los aspectos más relevantes incluyen:

3.1. Control de Motores DC

El uso de drivers como el L293D es común para el control de motores debido a su capacidad para manejar señales PWM y controlar direcciones. Se han explorado alternativas más avanzadas, como controladores integrados con retroalimentación de sensores para mayor precisión.

3.2. Interacción mediante Interfaz Web

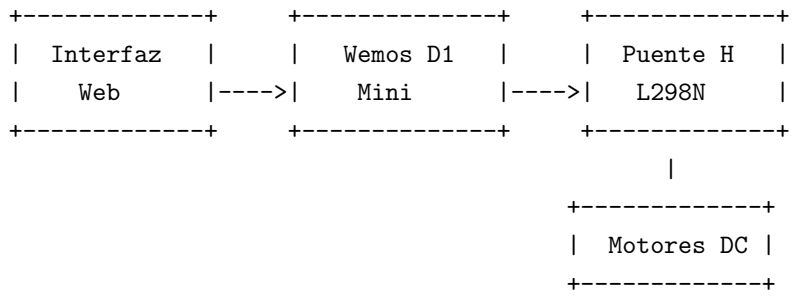
Con la proliferación de microcontroladores con conectividad Wi-Fi, como el ESP8266, es posible diseñar sistemas que se gestionan de forma remota. La combinación de tecnologías web (HTML, CSS, JavaScript) con protocolos como HTTP permite un control intuitivo y en tiempo real.

3.3. Optimización Energética

Los convertidores DC-DC step-down son la solución más eficiente para reducir voltaje en sistemas alimentados por baterías. Aunque los reguladores lineales son simples, su baja eficiencia los hace menos atractivos en proyectos donde la autonomía energética es crucial.

4. Metodología

4.1. Diagrama de Bloques



4.2. Pruebas Realizadas

Se llevaron a cabo pruebas en la plataforma Tinkercad, logrando un control eficiente de los motores. Las métricas obtenidas muestran precisión en la dirección y estabilidad en el control de velocidad.

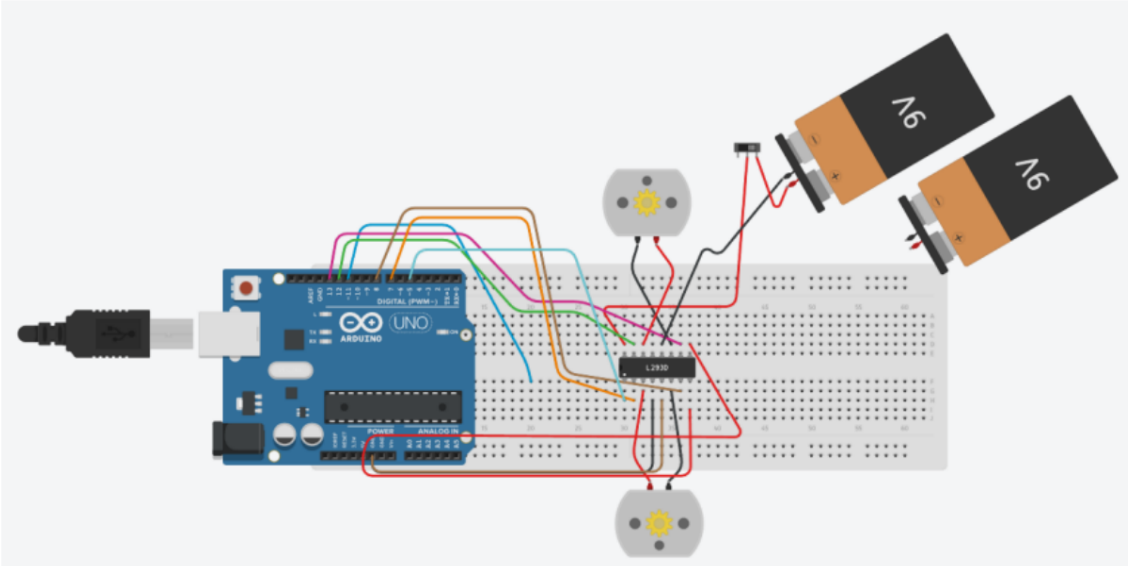


Figura 1: Simulacion realizada en Tinkercad.

4.3. Componentes Utilizados

- Wemos D1 Mini (microcontrolador basado en ESP8266)
- Driver L293D (controlador de motores)
- Motores DC x2 (dirección y arranque)
- Batería de 9V
- Interruptor (opcional)
- Resistencias y jumpers

4.4. Conexiones del Hardware

4.4.1. Conexiones en Wemos D1 Mini

- D1 – D2: Control de direccionales
- D6 – D7: Control de aceleración
- D5: Señal PWM para control de potencia
- D3: Control de iluminación

4.4.2. Conexiones en L293D

- Puertos 16 – 9: Alimentación 5V desde Wemos
- Puerto 8: Alimentación desde batería externa (positivo)
- Puerto 13: Conexión a batería externa (negativo)
- Puertos 3 – 6: Motor de aceleración
- Puertos 14 – 11: Motor de direccionales
- Puerto 4: Conexión a GND del Wemos
- Puerto 1: Conexión a D5 (PWM) del Wemos
- Puertos 2 – 7: Conexión a D6 – D7 (control de aceleración)
- Puertos 10 – 15: Conexión a D1 – D2 (control de direccionales)

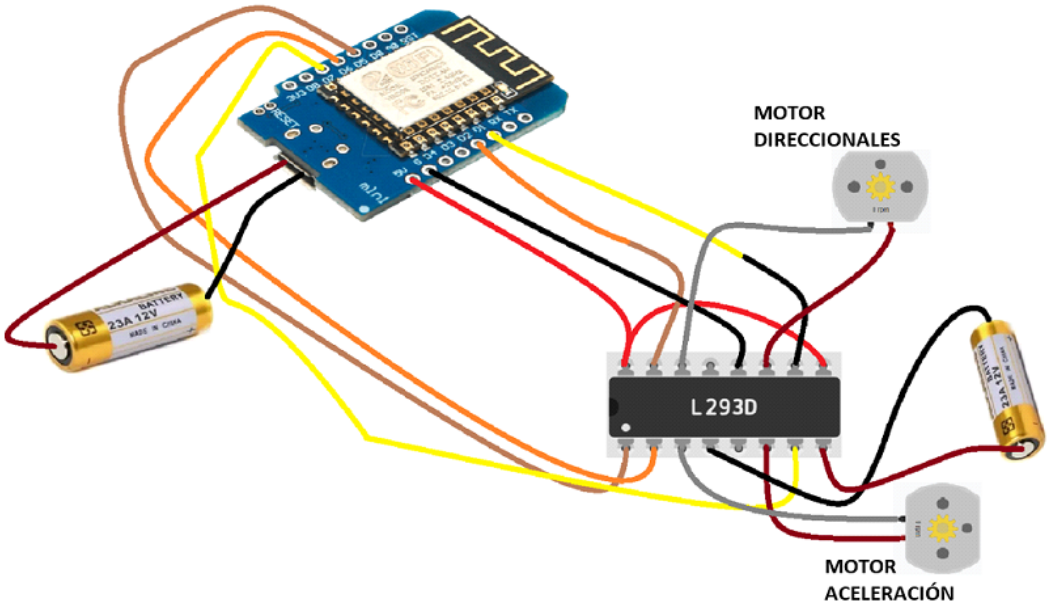


Figura 2: Bosquejo de Funcionalidad del Sistema

5. Diseño de la Interfaz Web

La interfaz web desarrollada para el control del carrito "Need for Speed"presenta un diseño moderno y minimalista, con un esquema de colores profesional basado en tonos de azul sobre un fondo blanco. La interfaz está contenida en un contenedor central con esquinas redondeadas y sombras suaves, que alberga tres grupos principales de controles: manejo de luces (encendido/apagado), control direccional (izquierda/derecha) y control de movimiento (acelerar/desacelerar/detener).

Todos los botones cuentan con efectos visuales interactivos que mejoran la experiencia del usuario, como cambios de escala y color al pasar el mouse o hacer clic. La interfaz implementa una conexión WebSocket para comunicación en tiempo real, con un indicador de estado en la parte inferior que utiliza códigos de color (verde para conectado, rojo para desconectado y naranja para errores) para mantener al usuario informado sobre el estado de la conexión. Los botones se deshabilitan automáticamente cuando no hay conexión, asegurando una experiencia de usuario fluida y evitando comandos incorrectos.

Figura 3: Diseño de la Interfaz Web

6. Mantenimiento y Consideraciones Técnicas

6.1. Proceso de Desensamblaje

El proceso de desensamblaje requiere precisión para evitar desconexiones accidentales, especialmente durante la carga de baterías. Se recomienda seguir estos pasos:

6.1.1. Paso 1: Desmontaje de la Carcasa

- Retirar los tornillos de la parte posterior
- Desplegar suavemente hacia abajo
- Jalar hacia atrás para separar el empalme

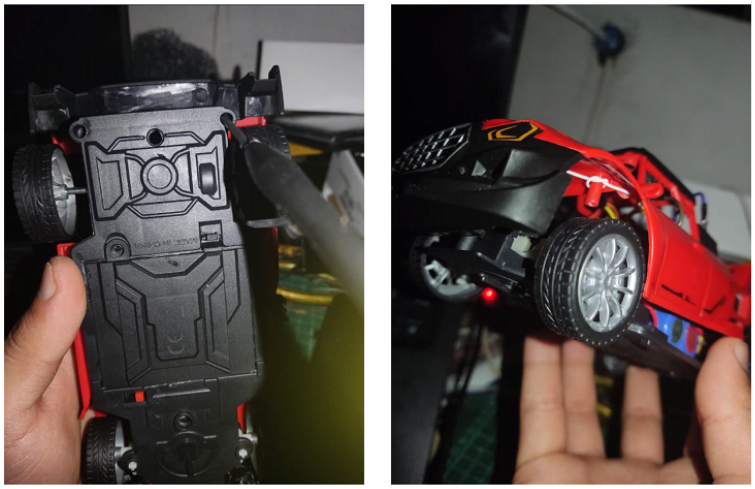


Figura 4: Guia de desmontaje.

6.1.2. Paso 2: Acceso a la Batería

- Levantar cuidadosamente la carcasa
- Localizar la conexión de la batería en la superficie
- Presionar el pin de seguridad
- Desconectar con cuidado los cables, prestando especial atención al cable con restricción de movimiento

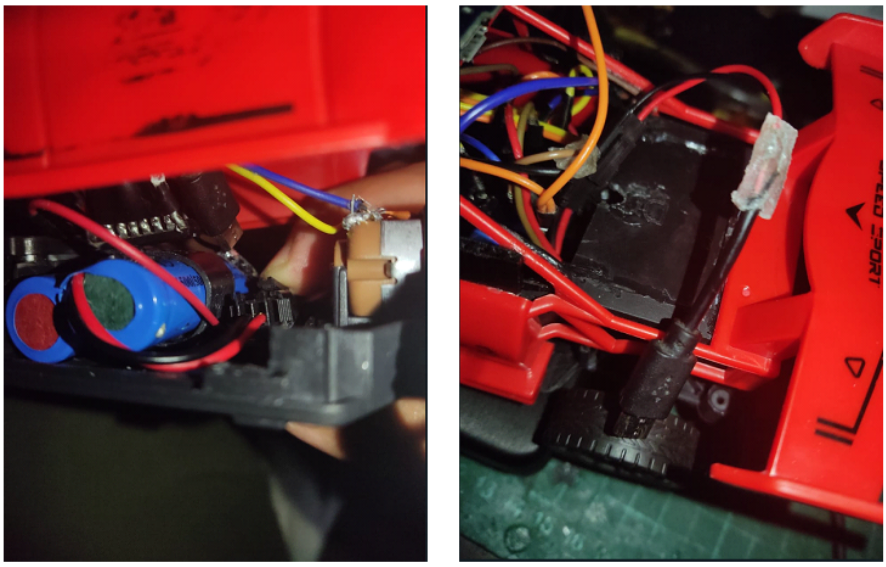


Figura 5: Guia de desmontaje.

7. Recomendaciones Técnicas

7.1. Soldadura de Componentes

Durante el desarrollo del proyecto, se identificaron desafíos significativos en la soldadura de componentes pequeños, específicamente con el L293D. Para una soldadura exitosa, se recomienda:

1. **Preparación de la Superficie**
   - Limpiar los puntos de soldadura con malla de alambre
   - Eliminar residuos de estaño previos
2. **Técnica de Soldadura**
   - Empastar adecuadamente el cautín
   - Acercar el estaño a la punta del cautín
   - Realizar movimientos precisos y rápidos
   - Mantener un tiempo mínimo de contacto

3. **Consideraciones Importantes**

   - Evitar el sobrecalentamiento de los componentes
   - Verificar la calidad de la soldadura (aspecto de "puntito")
   - Realizar pruebas de continuidad después de cada soldadura

8. Problemas y Soluciones

8.1. Dificultades Encontradas

Durante el desarrollo del proyecto se enfrentaron varios desafíos:

- Pérdida de dos controladores L293D durante el proceso de soldadura
- Complejidad en el manejo de componentes de tamaño reducido
- Necesidad de precisión extrema en las conexiones

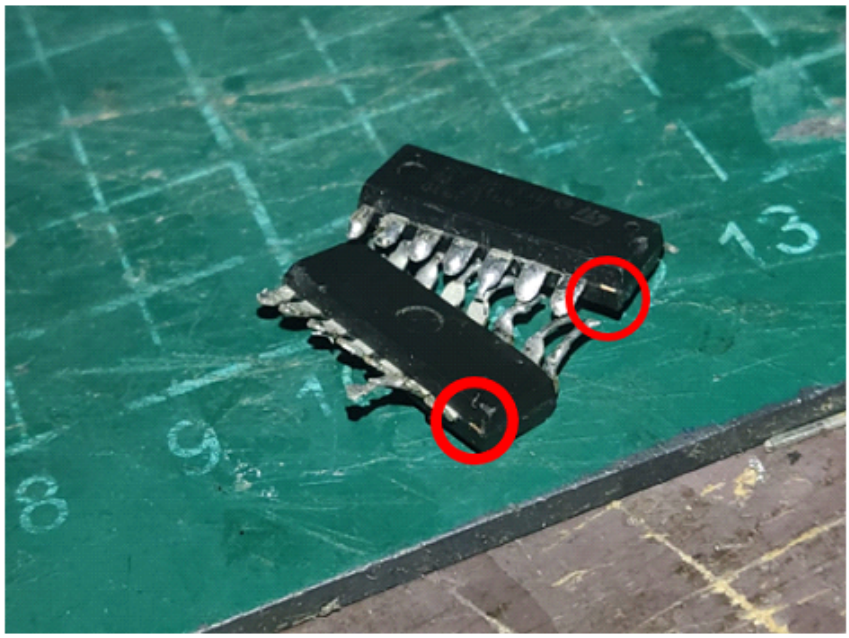


Figura 6: L293D dañados.

8.2. Soluciones Implementadas

- Desarrollo de un protocolo detallado de soldadura
- Implementación de un proceso sistemático de ensamblaje
- Documentación exhaustiva del proceso de mantenimiento

9. Conclusiones

El objetivo principal de este proyecto, que ha sido diseñar e implementar un prototipo de vehículo a escala para que pudiera ser controlado de forma remota a través de un dispositivo mediante red wif, se puede afirmar que ha sido logrado de manera satisfactoria. Para alcanzar este objetivo, se han tenido que cumplir previamente otros objetivos, definidos previamente: Se ha conseguido diseñar y construir una estructura similar a la de un vehículo, en la que poder añadir toda la circuitería para que se mueva sin tener que depender de la conexion de un ordenador o una fuente de energia externa. Se ha desarrollado una aplicación web para su control en la que se tienen en cuenta tanto los posibles movimientos del vehículo como los dos modos de manejo, cuya finalidad principal es manejar el vehículo. Gracias al diseño sencillo e intuitivo, uno de los principales objetivos del diseño de la aplicación web, cualquier persona independientemente de la experiencia que tenga utilizando aplicaciones móviles, puede manejar el vehículo.

9.1. Logros del Proyecto

- Se logró diseñar e implementar un robot móvil funcional y autónomo
- Se solucionaron los desafíos de gestión energética
- La integración del interruptor lógico añadió flexibilidad al control

9.2. Innovación y Creatividad

- El diseño destaca por su enfoque modular
- La interfaz web simplifica la interacción y la curva de aprendizaje

9.3. Recomendaciones

- Integrar sensores de proximidad para evitar colisiones
- Optimizar el software para reducir la latencia
- Explorar el uso de tecnologías IoT para análisis en tiempo real

Referencias

[1] N. Londoño, *Metodologías de Desarrollo de Software, un enfoque a Robots Móviles*, Revista Politécnica, ISSN 1900-2351, Número 8, pp. 74-83, 2009.

[2] J. Pastor, F. J. Rodríguez, *La robótica como elemento de motivación del aprendizaje en los alumnos de ingeniería y potenciación de habilidades profesionales*, Departamento de Electrónica, Universidad de Alcalá, España, 15 de julio 2015.

[3] F. Lewis, *Robotics*, Mechanical Engineering Handbook, Frank Kreith, CRC Press LLC, 1999.

[4] R. R. Murphy, *Introduction to AI Robotics*, MIT Press, 2000.

[5] P. Rusu, E. M. Petriu, T. E. Whalen, A. Cornell, H. J. Spoelder, *Behavior-Based Neuro-Fuzzy Controller for Mobile Robot Navigation*, IEEE Transactions on Instrumentation and Measurement, vol. 52, no. 4, pp. 1335-1340, August 2003.

[6] S. Samuel, V. Kavati, N. Uddin, *Trajectory Planning of a Mobile Robot*, National Conference on Technological Advancements in Mechanical Engineering, 2016.

[7] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed., MIT Press, London, England, 2011.

A. Anexo: Código cargado al Wemos

En esta sección se presentan los detalles del código que se ha cargado al Wemos. A continuación, se incluye el código correspondiente:

```
#include <ESP8266WiFi.h>
#include <ESPAsyncWebServer.h>
#include <WebSocketsServer.h>

const char* ssid = "RUBENS 2";
const char* password = "Val2321eraB1111";

AsyncWebServer server(80);
WebSocketsServer webSocket(81);

int velocidad = 0; // Velocidad actual
int velocidadMax = 255; // Velocidad m xima del motor
int velocidadMin = 150; // Velocidad m nima para superar inercia
bool enMovimiento = false; // Estado del motor (movi ndose o detenido)
bool retrocediendo = false; // Estado de retroceso

// Pines del motor y luz
int m1_sta1 = D1; // Direcci n
int m1_sta2 = D2;

int m2_pwm = D5; // Motor principal (velocidad)
int m2_sta1 = D7;
int m2_sta2 = D6;

int luzPin = D3; // Luces

void setup() {
Serial.begin(115200);
Serial.println("Iniciando programa...");

pinMode(m1_sta1, OUTPUT);
pinMode(m1_sta2, OUTPUT);
pinMode(m2_pwm, OUTPUT);
pinMode(m2_sta1, OUTPUT);
pinMode(m2_sta2, OUTPUT);
pinMode(luzPin, OUTPUT);

apagarLuces();
detenerDireccion();
detenerArranque();

WiFi.begin(ssid, password);
Serial.print("Conectando a WiFi...");
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.print(".");
}
Serial.println("\nConectado a WiFi");
Serial.println(WiFi.localIP());

webSocket.begin();
webSocket.onEvent(onWebSocketEvent);
```

```
53
54 server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
55 request->send(200, "text/plain", "Servidor WebSocket para carrito");
56 });
57 server.begin();
58 }
59
60 void loop() {
61 webSocket.loop();
62 }
63
64 void onWebSocketEvent(uint8_t num, WStype_t type, uint8_t *payload, size_t length) {
65     if (type == WStype_TEXT) {
66     String command = String((char *)payload);
67     Serial.println("Comando recibido: " + command);
68     if (command == "luces_on") encenderLuces();
69     else if (command == "luces_off") apagarLuces();
70     else if (command == "left") izquierda();
71     else if (command == "right") derecha();
72     else if (command == "stop_dir") detenerDireccion();
73     else if (command == "accelerate") acelerar();
74     else if (command == "decelerate") desacelerar();
75     else if (command == "stop_motor") detenerArranque();
76
77 }
78 }
79
80 void izquierda() {
81 Serial.println("Girando a la izquierda");
82 digitalWrite(m1_sta1, HIGH);
83 digitalWrite(m1_sta2, LOW);
84 }
85
86 void derecha() {
87 Serial.println("Girando a la derecha");
88 digitalWrite(m1_sta1, LOW);
89 digitalWrite(m1_sta2, HIGH);
90 }
91
92 void detenerDireccion() {
93 Serial.println("Deteniendo direcci n");
94 digitalWrite(m1_sta1, LOW);
95 digitalWrite(m1_sta2, LOW);
96 }
97
98 void acelerar() {
99 if (retrocediendo) {
100 Serial.println("No se puede acelerar mientras retrocede. Det n el motor primero.");
101 return;
102 }
103
104 Serial.println("Acelerando...");
105 digitalWrite(m2_sta1, HIGH); // Direcci n hacia adelante
106 digitalWrite(m2_sta2, LOW);
107
108 if (velocidad < velocidadMin) {
109 velocidad = velocidadMin; // Valor m nimo para superar la inercia
```

```
}

analogWrite(m2_pwm, velocidadMax); // Velocidad m xima
enMovimiento = true;
retrocediendo = false;
}

void desacelerar() {
if (enMovimiento) {
Serial.println("Det n el motor antes de retroceder.");
detenerArranque(); // Detener primero
delay(500); // Peque a pausa para evitar conflictos
}

Serial.println("Retrocediendo...");
digitalWrite(m2_sta1, LOW); // Direcci n hacia atr s
digitalWrite(m2_sta2, HIGH);

if (velocidad < velocidadMin) {
velocidad = velocidadMin; // Valor m nimo para superar la inercia al retroceder
}

analogWrite(m2_pwm, velocidadMax); // Velocidad m xima al retroceder
retrocediendo = true;
enMovimiento = false;
}

void detenerArranque() {
Serial.println("Deteniendo motor...");
velocidad = 0;
analogWrite(m2_pwm, velocidad); // Detener el motor
digitalWrite(m2_sta1, LOW);
digitalWrite(m2_sta2, LOW);

enMovimiento = false;
retrocediendo = false;
}

void encenderLuces() {
Serial.println("Encendiendo luces");
digitalWrite(luzPin, HIGH);
}

void apagarLuces() {
Serial.println("Apagando luces");
digitalWrite(luzPin, LOW);
}
```

Listing 1: Código para el Wemos

B. Anexo: Código de la Interfaz HTML

En esta sección se presentan los detalles del código de la interfaz HTML. A continuación, se incluye el código correspondiente:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Control de Carrito</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      color: #333;
      text-align: center;
      padding: 20px;
    }

    h1 {
      margin-bottom: 20px;
      font-size: 2rem;
      color: #007BFF;
    }

    .container {
      max-width: 600px;
      margin: 0 auto;
      background: #fff;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    .button-group {
      margin: 20px 0;
      display: flex;
      justify-content: center;
      align-items: center;
    }

    .button-group.left-right {
      justify-content: space-between;
    }

    button {
      font-size: 16px;
      margin: 10px;
      padding: 15px 25px;
      border: none;
```

```
      border-radius: 5px;
      background-color: #007BFF;
      color: #fff;
      cursor: pointer;
      transition: background-color 0.3s, transform 0.2s;
    }

    button:hover {
      background-color: #0056b3;
      transform: scale(1.05);
    }

    button:active {
      transform: scale(0.95);
    }

    button:disabled {
      background-color: #ccc;
      cursor: not-allowed;
    }

    .status {
      margin-top: 20px;
      font-size: 1rem;
      color: #555;
    }
  </style>
</head>
<body>
  <h1>Control de Carrito - Need for Speed</h1>
  <div class="container">
    <div class="button-group">
      <button id="luces_on" onclick="sendCommand('luces_on')">Encender Luces</button>
      <button id="luces_off" onclick="sendCommand('luces_off')">Apagar Luces</button>
    </div>

    <div class="button-group left-right">
      <button id="left" onmousedown="startCommand('left')" onmouseup="stopCommand('
    stop_dir')">Izquierda</button>
      <button id="right" onmousedown="startCommand('right')" onmouseup="stopCommand('
    stop_dir')">Derecha</button>
    </div>

    <div class="button-group">
      <button id="accelerate" onclick="sendCommand('accelerate')">Acelerar</button>
      <button id="decelerate" onclick="sendCommand('decelerate')">Desacelerar</button
    >
      <button id="stop_motor" onclick="sendCommand('stop_motor')">Detener Motor</
    button>
    </div>

    <div class="status" id="status">
      Estado: Conectando...
    </div>
  </div>

  <script>
```

```
    const socket = new WebSocket('ws://192.168.100.110:81');
    const statusElement = document.getElementById('status');
    const buttons = {
      accelerate: document.getElementById('accelerate'),
      decelerate: document.getElementById('decelerate'),
      stop_motor: document.getElementById('stop_motor'),
    };

    let enMovimiento = false; // Estado del motor
    let retrocediendo = false; // Estado de retroceso

    // Actualizar estado del WebSocket
    socket.onopen = () => {
      console.log('WebSocket conectado');
      statusElement.textContent = 'Estado: Conectado';
      statusElement.style.color = 'green';

      // Habilitar botones al conectar
      for (const button in buttons) {
        buttons[button].disabled = false;
      }
    };

    socket.onclose = () => {
      console.log('WebSocket desconectado');
      statusElement.textContent = 'Estado: Desconectado';
      statusElement.style.color = 'red';

      // Deshabilitar botones al desconectar
      for (const button in buttons) {
        buttons[button].disabled = true;
      }
    };

    socket.onerror = (error) => {
      console.error('WebSocket error:', error);
      statusElement.textContent = 'Estado: Error en conexi n';
      statusElement.style.color = 'orange';
    };

    // Enviar comando al servidor
    function sendCommand(command) {
      if (socket.readyState === WebSocket.OPEN) {
        if (command === 'decelerate' && enMovimiento) {
          enMovimiento = false;
          retrocediendo = true;
        } else if (command === 'accelerate') {
          enMovimiento = true;
          retrocediendo = false;
        }

        socket.send(command);
        console.log(Comando enviado: ${command});
      } else {
        console.error('WebSocket no est  conectado');
        alert('No se puede enviar el comando: WebSocket no est  conectado');
      }
```

```
    }

    // Comenzar un comando continuo (para botones de direcci n)
    function startCommand(command) {
      if (socket.readyState === WebSocket.OPEN) {
        socket.send(command);
        console.log(Comando enviado: ${command});
      } else {
        console.error('WebSocket no est  conectado');
      }
    }

    // Detener un comando continuo (para botones de direcci n)
    function stopCommand(command) {
      if (socket.readyState === WebSocket.OPEN) {
        socket.send(command);
        console.log(Comando enviado: ${command});
      } else {
        console.error('WebSocket no est  conectado');
      }
    }
  </script>
</body>
</html>
```

Listing 2: Código de la interfaz HTML