

# PROYECTO FINAL PROGRAMACION APLICADA

Alejandro Rubiano Alarcón - 20212005159

Universidad Distrital Francisco José de Caldas

Ingeniería Electrónica - Programación Aplicada

enero 2025



UNIVERSIDAD DISTRITAL  
FRANCISCO JOSÉ DE CALDAS

## 1. Introducción

**Introducción:** Este proyecto consiste en desarrollar un robot utilizando un microcontrolador ESP32 programado en MicroPython. El robot se inspirará en el *Boom Bot* de *Valorant*, un robot de exploración que se desplaza por el mapa buscando enemigos. El propósito de este proyecto es aplicar los conocimientos adquiridos en el curso de programación aplicada, combinando la robótica, sensores, y el control de movimiento con un microcontrolador.

## 2. Descripción del problema

**Descripción del problema:** El proyecto busca resolver la necesidad de crear un robot autónomo capaz de moverse en un espacio determinado, detectar obstáculos y evitar colisiones. Este tipo de robots tiene aplicaciones en la automatización de tareas, exploración en áreas peligrosas, o simplemente en la creación de dispositivos interactivos en juegos o entornos virtuales.

## 3. Objetivos

**Objetivos:**

1. Desarrollar un robot móvil con el ESP32 como cerebro central.
2. Programar la lógica de movimiento utilizando sensores para evitar obstáculos.
3. Implementar un sistema de control de motor para que el robot pueda desplazarse de manera autónoma.
4. Integrar un sistema de activación que simula la acción de un “Boom Bot” con la detección de obstáculos.
5. Documentar el proceso y compartir el código fuente.

## 4. Componentes a utilizar

**Componentes:**

1. **ESP32:** Microcontrolador para manejar la programación y el control de los sensores/motores.
2. **Motores DC:** Para el movimiento del robot.
3. **Driver de motores L298N:** Para controlar la dirección y velocidad de los motores.
4. **Sensores ultrasónicos (HC-SR04):** Para detectar obstáculos a una distancia.
5. **Batería recargable (por ejemplo, Li-ion):** Para alimentar el robot.
6. **Ruedas y estructura del robot:** Para montar los componentes.
7. **Cables y conectores:** Para realizar las conexiones entre los componentes.
8. **Placa base (breadboard o PCB):** Para facilitar las conexiones.
9. **Chasis o carcasa:** Para mantener todos los componentes en su lugar.
10. **Cableado de mayor calibre** para evitar sobrecalentamiento en las conexiones donde fluye mayor corriente, como las que alimentan los motores y el L298N.

11. **Transformador** con devanado secundario de 6 V y tierra (6 V tierra, 6 V a tierra), utilizado para proporcionar una alimentación más estable y suficiente para los motores.

## 5. Código fuente

```
from machine import Pin
import time

# Configuración del pulsador
boton = Pin(4, Pin.IN, Pin.PULL_UP) # Pulsador con resistencia
pull-up interna

# Configuración de los pines para el control del puente H (L298N)
motor1_a = Pin(12, Pin.OUT) # IN1 - Motor 1 adelante
motor1_b = Pin(14, Pin.OUT) # IN2 - Motor 1 atrás
motor2_a = Pin(27, Pin.OUT) # IN3 - Motor 2 adelante
motor2_b = Pin(26, Pin.OUT) # IN4 - Motor 2 atrás

# Configuración de los pines para el sensor ultrasónico HC-SR04
trig = Pin(23, Pin.OUT)
echo = Pin(22, Pin.IN)

# Configuración de los LEDs
leds = [Pin(5, Pin.OUT), Pin(18, Pin.OUT), Pin(19, Pin.OUT), Pin(21,
Pin.OUT)]

# Variable para almacenar la dirección del motor
direccion = 0 # 0: adelante, 1: atrás

# Función para medir la distancia con el sensor ultrasónico
def medir_distancia():
    trig.value(0)
    time.sleep_us(2)
    trig.value(1)
    time.sleep_us(10)
    trig.value(0)

    while echo.value() == 0:
        inicio = time.ticks_us()
    while echo.value() == 1:
        fin = time.ticks_us()
```

```

    duracion = time.ticks_diff(fin, inicio)
    distancia = (duracion * 0.0343) / 2
    return distancia

# Funciones para controlar el movimiento del robot
def mover_adelante():
    motor1_a.value(1)
    motor1_b.value(0)
    motor2_a.value(1)
    motor2_b.value(0)

def mover_atras():
    motor1_a.value(0)
    motor1_b.value(1)
    motor2_a.value(0)
    motor2_b.value(1)

def girar_izquierda():
    motor1_a.value(0)
    motor1_b.value(1)
    motor2_a.value(1)
    motor2_b.value(0)

def girar_derecha():
    motor1_a.value(1)
    motor1_b.value(0)
    motor2_a.value(0)
    motor2_b.value(1)

def detenerse():
    motor1_a.value(0)
    motor1_b.value(0)
    motor2_a.value(0)
    motor2_b.value(0)

# Función para cambiar la dirección del motor con el botón
def cambiar_direccion(pin):
    global direccion
    direccion = not direccion # Alternar entre adelante y atrás
    if direccion:

```

```

        mover_atras()
    else:
        mover_adelante()
        print("Dirección cambiada:", "Atrás" if direccion else
"Adelante")

# Asignar interrupción al pulsador
boton.irq(trigger=Pin.IRQ_FALLING, handler=cambiar_direccion)

# Función para encender los LEDs en onda
def leds_en_onda():
    for led in leds:
        led.value(1)
        time.sleep(0.1)
        led.value(0)

# Lógica principal para el control del robot
while True:
    distancia = medir_distancia()
    print("Distancia: ", distancia, "cm")

    if distancia < 20:
        print("Obstáculo detectado. Cambiando dirección.")
        detenerse()
        time.sleep(0.5)
        mover_atras()
        time.sleep(0.5)
        girar_derecha()
        time.sleep(0.5)
    else:
        if direccion == 0:
            mover_adelante()
        else:
            mover_atras()

    leds_en_onda()
    time.sleep(0.1)

```

Este código hace lo siguiente:

- Usa un sensor ultrasónico para medir la distancia.
- Si el robot detecta que hay un obstáculo a menos de 20 cm, se detiene.
- Si no detecta obstáculos, se mueve hacia adelante.
- Leds que encienden y apagan con iluminación de onda
- Botón pulsador que hace que cambie de giro los motores

## 6. Proceso de construcción

### Proceso de construcción:

1. Conectar el ESP32 a la placa base (protoboard) para realizar las conexiones iniciales.
2. Instalar los motores DC en la estructura del robot, asegurándose de que estén bien sujetos para evitar vibraciones.
3. Conectar los motores al driver L298N utilizando cables para garantizar una conexión segura y minimizar riesgos de sobrecalentamiento.
4. Alimentar el puente H L298N utilizando un transformador con un devanado secundario de 6 V y tierra, configurado para proporcionar la energía necesaria a los motores. Esta configuración se eligió debido a que la batería de 9V no era suficiente para soportar el consumo de corriente requerido por los motores.
5. Instalar el sensor ultrasónico en la parte delantera del robot, conectado a los pines correspondientes del ESP32.
6. Programar el ESP32 con MicroPython para gestionar el control de los motores y el sensor ultrasónico.
7. Probar y ajustar el sistema para garantizar que el robot pueda moverse y evitar obstáculos correctamente.

## Conectar los Motores al Driver L298N

### Conexión de motores:

#### Motor 1 (Motor A):

- Motor A + (positivo): Conectar uno de los cables del motor al pin Out 1 del L298N.
- Motor A - (negativo): Conectar el otro cable del motor al pin Out 2 del L298N.

#### Motor 2 (Motor B):

- Motor B + (positivo): Conectar uno de los cables del segundo motor al pin Out 3 del L298N.
- Motor B - (negativo): Conectar el otro cable del motor al pin Out 4 del L298N.

## Conectar el Driver L298N al ESP32

Ahora que los motores están conectados al L298N, conectar los pines de control del L298N al ESP32 para poder controlar la dirección y la velocidad de los motores.

- **IN1 (Motor 1):** Conectar el pin IN1 del L298N al D12 del ESP32.
- **IN2 (Motor 1):** Conectar el pin IN2 del L298N al D14 del ESP32.
- **IN3 (Motor 2):** Conectar el pin IN3 del L298N al D27 del ESP32.
- **IN4 (Motor 2):** Conectar el pin IN4 del L298N al D26 del ESP32.

## Conectar el Sensor Ultrasónico HC-SR04 al ESP32

El sensor ultrasónico sirve para medir la distancia y detectar obstáculos. Este sensor tiene cuatro pines: VCC, Trig, Echo, y GND.

- **VCC:** Se conecta a 3V3 del ESP32.
- **GND:** Se conecta al GND del ESP32.
- **Trig:** Se conecta a un pin digital del ESP32 al D23.
- **Echo:** Se conecta a otro pin digital del ESP32 al D22.

## Conectar el Control de Movimiento

- **Motor 1 (dirección):** Controlado por D12 y D14.
- **Motor 2 (dirección):** Controlado por D27 y D26.
- **Velocidad de Motor 1:** Controlada por D13.
- **Velocidad de Motor 2:** Controlada por D25.
- **Sensor Ultrasónico:**
  - Trig: Conectado a D23.
  - Echo: Conectado a D22.

## Conectar los LEDs al ESP32

Se utilizarán cuatro LEDs conectados a los siguientes pines del ESP32:

- **LED 1:** Conectado al D5.
- **LED 2:** Conectado al D18.
- **LED 3:** Conectado al D9.
- **LED 4:** Conectado al D21.

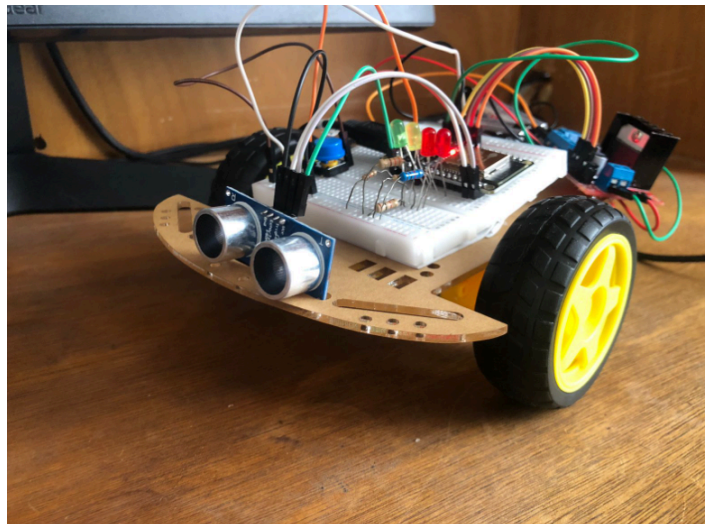
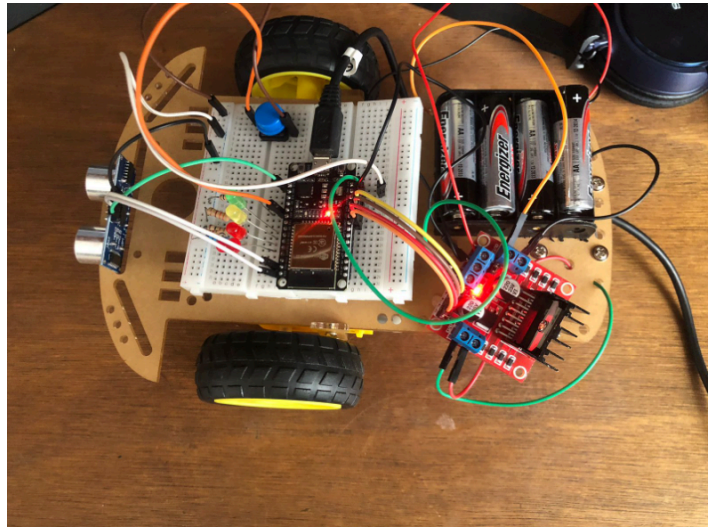
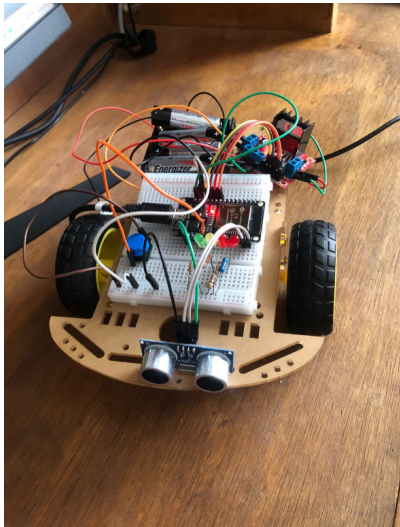
## Conectar el Pulsador al ESP32

El pulsador se conectará al D4

- **Un pin del pulsador:** Conectar a GND del ESP32.
- **Otro pin del pulsador:** Conectar al D4 del ESP32.

Al presionar el botón, el GPIO 4 se pondrá en **LOW (0V)** y se podrá detectar en el código para cambiar la dirección del motor.

## 7. Diseño final imagenes



## 8. Costo Materiales total

El costo total de los materiales utilizados en la construcción del robot fue de **122.500**. Donde ese costo fue dirigido para la compra:

El programa tuvo una última actualización que fue los leds que encendieran continuamente en onda.

1. **Costo de materiales:** 122,500 COP.
2. **Costo de programación (4 horas a 35,000 COP por hora):** 140,000 COP.

El valor total estimado del robot sería:



- **122,500 COP + 140,000 COP = 262,500 COP.**

## **9. Conclusiones**

**Conclusiones:** El proyecto fue una excelente oportunidad para aplicar los conocimientos adquiridos en programación aplicada, especialmente en el uso de MicroPython con el ESP32 para controlar hardware. A través de la implementación de sensores y motores, se logró construir un robot autónomo que es capaz de evitar obstáculos, lo que demuestra la viabilidad de utilizar microcontroladores en proyectos de robótica.