Carrito exploraror

Ricardo Antonio Gil Garcia 239536

11 de abril de 2025

1. Codigo en ARDUINO IDE

```
#include <WiFi.h>
#include <WebServer.h>

// WiFi
const char* ssid = "Gilst";
const char* password = "1605ragg";

WebServer server(80);

// Pines del puente H
const int ENA = 7;  // PWM Motor 1
const int IN1 = 6;
const int IN2 = 5;
const int ENB = 2;  // PWM Motor 2
const int IN3 = 4;
const int IN4 = 3;

// Pines del sensor ultras nico
const int triggerPin = 8;
const int echoPin = 9;

void setup() {
  Serial.begin(115200);

  // Pines motores
  pinMode(ENA, OUTPUT);
```

```
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(ENB, OUTPUT);
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);

// Pines sensor ultras nico
pinMode(triggerPin, OUTPUT);
pinMode(echoPin, INPUT);

// Conexi n WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

Serial.println("\WiFi conectado");
Serial.print("IP: ");
Serial.println(WiFi.localIP());

// Movimiento
server.on("/adelante", []() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, 200);
  analogWrite(ENB, 200);
  server.send(200, "text/plain", "Motores avanzando");
});

server.on("/atras", []() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 200);
  analogWrite(ENB, 200);
```

```
  server.send(200, "text/plain", "Motores␣
     retrocediendo");
});

server.on("/girar_derecha", []() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, 125);
  analogWrite(ENB, 125);
  server.send(200, "text/plain", "Giro␣a␣la␣derecha");
});

server.on("/girar_izquierda", []() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 125);
  analogWrite(ENB, 125);
  server.send(200, "text/plain", "Giro␣a␣la␣izquierda"
     );
});

server.on("/detener", []() {
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  server.send(200, "text/plain", "Motores␣detenidos");
});

// Endpoint para la distancia
server.on("/distancia", []() {
  long duracion, distancia;
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
```

```
    duracion = pulseIn(echoPin, HIGH);
    distancia = duracion * 0.034 / 2;

    String mensaje = "Distancia: " + String(distancia) +
        " cm";
    server.send(200, "text/plain", mensaje);
  });

  server.begin();
  Serial.println("Servidor iniciado");
}

void loop() {
  server.handleClient();
}
```

2. Codigo en ARDUINO IDE

```
import tkinter as tk
import requests
from pyfirmata import Arduino, util

# === Configuraci n de Arduino ===
arduino_port = "COM3"
board = Arduino(arduino_port)
it = util.Iterator(board)
it.start()

# Pines del joystick
x_pin = board.get_pin('a:3:i')  # Eje Y
y_pin = board.get_pin('a:2:i')  # Eje X
x_pin.enable_reporting()
y_pin.enable_reporting()

# Direcci n IP de la Raspberry Pi Pico W
pico_ip = "192.168.94.40"
ultimo_comando = None
```

```
# === L gica de env o de comandos ===
def enviar_comando(comando):
    global ultimo_comando
    if comando != ultimo_comando:
        try:
            url = f"http://{pico_ip}/{comando}"
            response = requests.get(url, timeout=1)
            if response.status_code == 200:
                print(f"Comando␣{comando}␣enviado")
            else:
                print(f"Error␣al␣enviar␣{comando}:␣{
                   response.status_code}")
        except Exception as e:
            print(f"Error␣de␣conexi n:␣{e}")
        ultimo_comando = comando

# === Zona muerta y movimiento ===
zona_muerta = 0.3
umbral_mov = 0.7
bloqueo_movimiento = False
limite_seguridad = 20  # cm

# === Actualizaci n por joystick ===
def actualizar_joystick():
    x_val = x_pin.read()
    y_val = y_pin.read()

    if x_val is not None and y_val is not None:
        x = x_val * 2 - 1
        y = y_val * 2 - 1

        if abs(x) < zona_muerta and abs(y) < zona_muerta
           :
            enviar_comando("detener")
        else:
            if abs(y) > abs(x):
                if y > umbral_mov:
                    if not bloqueo_movimiento:
                        enviar_comando("adelante")
                    else:
```

```
                        enviar_comando("detener")
                elif y < -umbral_mov:
                    enviar_comando("atras")
                else:
                    enviar_comando("detener")
            else:
                if not bloqueo_movimiento:
                    if x > umbral_mov:
                        enviar_comando("girar_derecha")
                    elif x < -umbral_mov:
                        enviar_comando("girar_izquierda"
                            )
                    else:
                        enviar_comando("detener")
                else:
                    enviar_comando("detener")

    root.after(100, actualizar_joystick)

# === Verificaci n de distancia ===
def verificar_distancia():
    global bloqueo_movimiento
    try:
        url = f"http://{pico_ip}/distancia"
        response = requests.get(url, timeout=1)
        if response.status_code == 200:
            data = response.text
            if "Distancia" in data:
                distancia = int(data.split(":")[1].
                    replace("cm", "").strip())
                print(f"Distancia detectada: {distancia}
                    cm")

                if distancia < limite_seguridad:
                    bloqueo_movimiento = True
                    mostrar_alerta_canvas()
                else:
                    bloqueo_movimiento = False
                    ocultar_alerta_canvas()
    except Exception as e:
```

```
        print(f"Error leyendo distancia: {e}")

    root.after(500, verificar_distancia)

# === Ventana de alerta canvas ===
alerta_canvas = tk.Canvas(width=500, height=100, bg="#
    fa0202", highlightthickness=4, highlightbackground="
    #000000")
alerta_texto = alerta_canvas.create_text(250, 50, text="
     PELIGRO !Objeto muy cerca", fill="#ffffff", font=("
    Arial", 20, "bold"))

def mostrar_alerta_canvas():
    if not alerta_canvas.winfo_ismapped():
        alerta_canvas.place(x=50, y=50)

def ocultar_alerta_canvas():
    if alerta_canvas.winfo_ismapped():
        alerta_canvas.place_forget()

# === Control por teclado ===
def manejar_tecla(event):
    tecla = event.keysym.lower()
    if tecla == "w":
        if not bloqueo_movimiento:
            enviar_comando("adelante")
        else:
            enviar_comando("detener")
    elif tecla == "s":
        enviar_comando("atras")
    elif tecla == "a":
        if not bloqueo_movimiento:
            enviar_comando("girar_izquierda")
        else:
            enviar_comando("detener")
    elif tecla == "d":
        if not bloqueo_movimiento:
            enviar_comando("girar_derecha")
        else:
            enviar_comando("detener")
```

```
    elif tecla == "space":
        enviar_comando("detener")

# === Salida segura ===
def cerrar():
    board.exit()
    root.destroy()

# === Interfaz principal ===
root = tk.Tk()
root.title("Control del Carrito (Joystick + WASD)")
root.geometry("600x300")
root.configure(bg="#f0f0f0")

titulo = tk.Label(
    root,
    text="Control del Carrito",
    font=("Comic Sans MS", 20, "italic bold underline"),
    fg="#800080", bg="#D3D3D3",
    relief="groove", bd=3, width=30
)
titulo.pack(pady=20)

btn_cerrar = tk.Button(root, text="Cerrar", font=("Arial
   ", 14), command=cerrar, bg="#aa2929", fg="white")
btn_cerrar.pack(pady=10)

# Vincular eventos de teclado
root.bind("<KeyPress>", manejar_tecla)

# Iniciar bucles
actualizar_joystick()
verificar_distancia()
root.mainloop()
```

3. Explicación de su funcionamiento

Para este proyecto se construyó un carrito explorador con la opción de ser controlado mediante un joystick o utilizando las teclas WASD del teclado. Además, cuenta con un sensor ultrasónico que mide la distancia a objetos cercanos y envía una señal para detener el carrito automáticamente antes de una posible colisión.

Se utilizaron dos programas: el primero, escrito en Arduino, se ejecuta en la Raspberry Pi Pico W y controla los dos motoreductores mediante un puente H, así como el sensor ultrasónico. Este código crea un servidor web que define rutas como `/adelante"`, `/atras"` y `/distancia"`, las cuales permiten enviar comandos de movimiento al carrito.

El segundo programa fue desarrollado en Python usando Spyder. Este código crea una interfaz con `tkinter` y utiliza `pyfirmata` para leer los valores del joystick desde una placa Arduino. Además, consulta constantemente el valor del sensor ultrasónico para mostrar una ventana emergente si un objeto se encuentra a menos de 20 cm de distancia, impidiendo que el carrito avance.

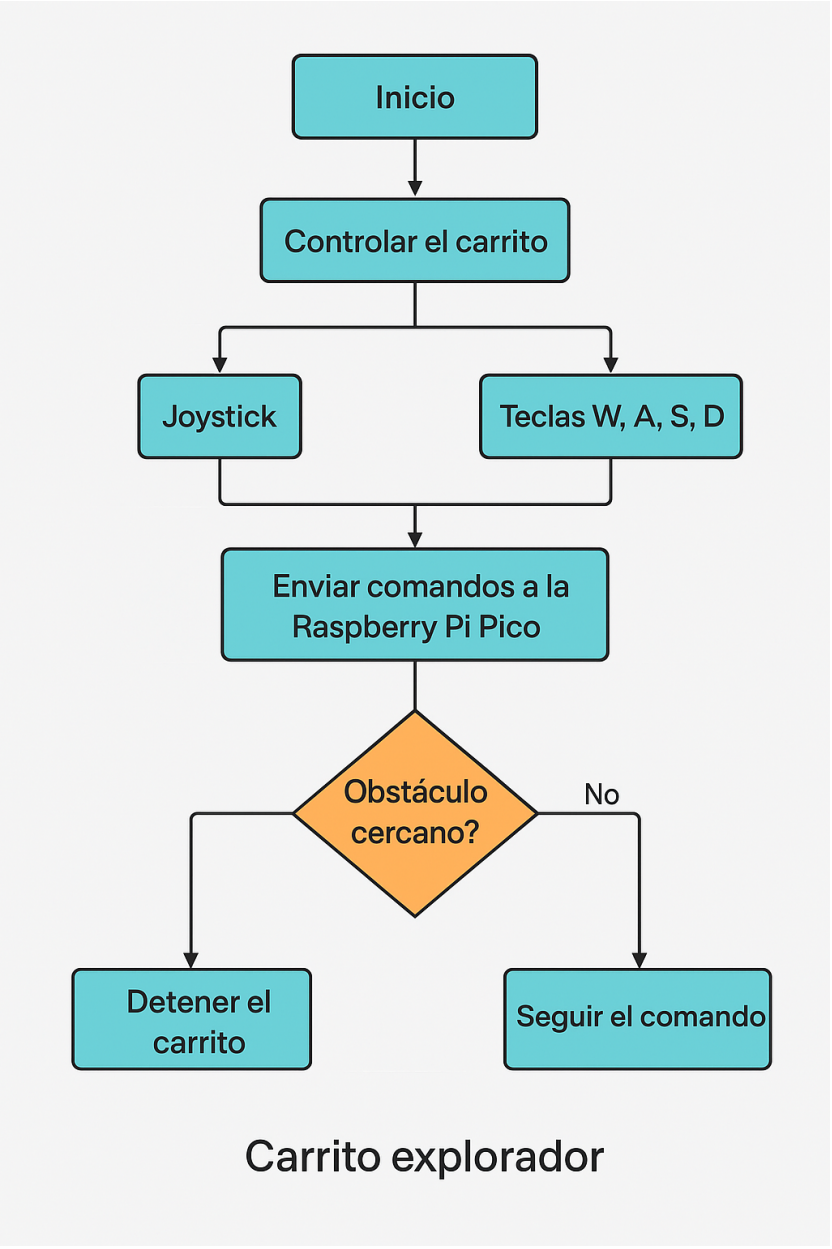

Figura 1: Diagrama de flujo