Descripción General

Este proyecto implementa un vehículo controlado por un **ESP32** conectado a **AWS IoT Core** usando **MQTT sobre TLS (Certificados X.509)**.

El ESP32:

- Controla motores mediante un L298N
- Lee distancia con un HC-SR04
- Lee IMU (acelerómetro y giroscopio) MPU6050
- Publica telemetría
- Recibe comandos para moverse

No usa HTTP (ni GET, ni POST).
Toda la comunicación es **MQTT**.

MQTT API — Topics, Payload y Dirección

Suscripción (ESP32 escucha):

1️⃣ Topic: `sebas/car/cmd`

El ESP32 espera comandos JSON.

Payloads permitidos:

A. Movimiento

```
{
  "dir": "forward",
  "speed": 50,
  "duration": 2
}
```

dir: `"forward" | "backward" | "left" | "right" | "stop"`
speed: 0–100 (porcentaje → convertido a PWM 0–255)
duration: segundos (máx 5s)

B. Stop

```
{ "stop": true }
```

C. Cambiar distancia segura

```
{ "safe_stop_cm": 30 }
```

📤 Publicación (ESP32 envía):

1️⃣ sebas/car/status

```
{
  "moving": true,
  "dir": "forward",
  "speed": 120
}
```

2️⃣ sebas/car/distance

```
{
  "cm": 25.4,
  "obstacle": false
}
```

3️⃣ sebas/car/imu

```
{
  "ax": 0.12,
  "ay": 9.80,
  "az": 0.03,
  "gx": 1.22,
  "gy": 0.03,
  "gz": -0.51
}
```

4️⃣ sebas/car/alert

```
{
  "type": "obstacle",
  "cm": 12.5
}
```

5️⃣ sebas/car/lwt

LWT al conectarse:

```
{ "status": "online" }
```

Limitaciones actuales

1. Sin reconexión robusta WiFi

Solo reintenta, no hay backoff exponencial ni manejo avanzado.

2. El parsing JSON es manual

Algo frágil. Sería mejor usar **ArduinoJson**.

3. No hay control PID para motores

El giro no es totalmente preciso.

4. Movimiento máximo: 5 segundos

Limitado por constante MAX_MOVE_MS.

5. Sin validación profunda de payload

Cualquier texto mal formateado puede romper lógica.

Librerías utilizadas

- `<Arduino.h>`
- `<WiFi.h>`
- `<WiFiClientSecure.h>`
- `<PubSubClient.h>`
- `<Wire.h>` (I2C)
- Drivers internos (PWM, GPIO)

Uso de Memoria

Generado por el IDE cuando se compila.

```
Sketch uses 1346 bytes (76%) of program storage space.
```

Estructura Interna del Firmware

- **`setup()`**
  - Inicialización de pines
  - WiFi STA
  - Configurar TLS + certificados
  - Conectar a AWS IoT
  - Suscripción a comandos
  - Publica estado inicial
- **`loop()`**
  - Mantener MQTT
  - Control de movimiento con timeout
  - Detección de obstáculos
  - Telemetría periódica