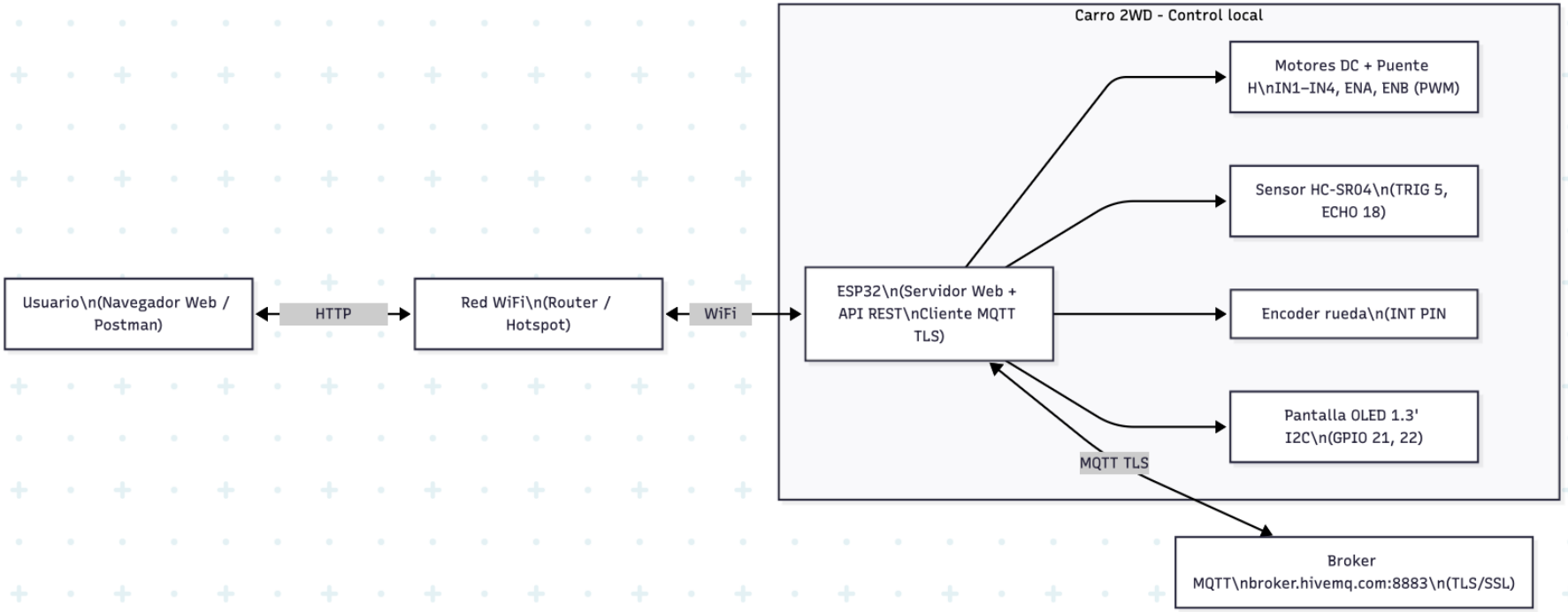
Carro 2WD con ESP32, Web UI, MQTT con TLS y API REST

Autores: Camilo Otálora · Juan Diego Martínez · Nicolás Rodríguez

Objetivo y requisitos del taller

- Partir del robot 2WD ya construido
  - Añadir detección de obstáculos con sensor ultrasónico (HC-SR04)
  - Bloquear automáticamente solo el movimiento hacia adelante
  - Publicar datos en un servidor MQTT
  - Usar MQTT con cifrado TLS (puerto 8883)
  - Exponer API REST: /api/v1/healthcheck y /api/v1/move

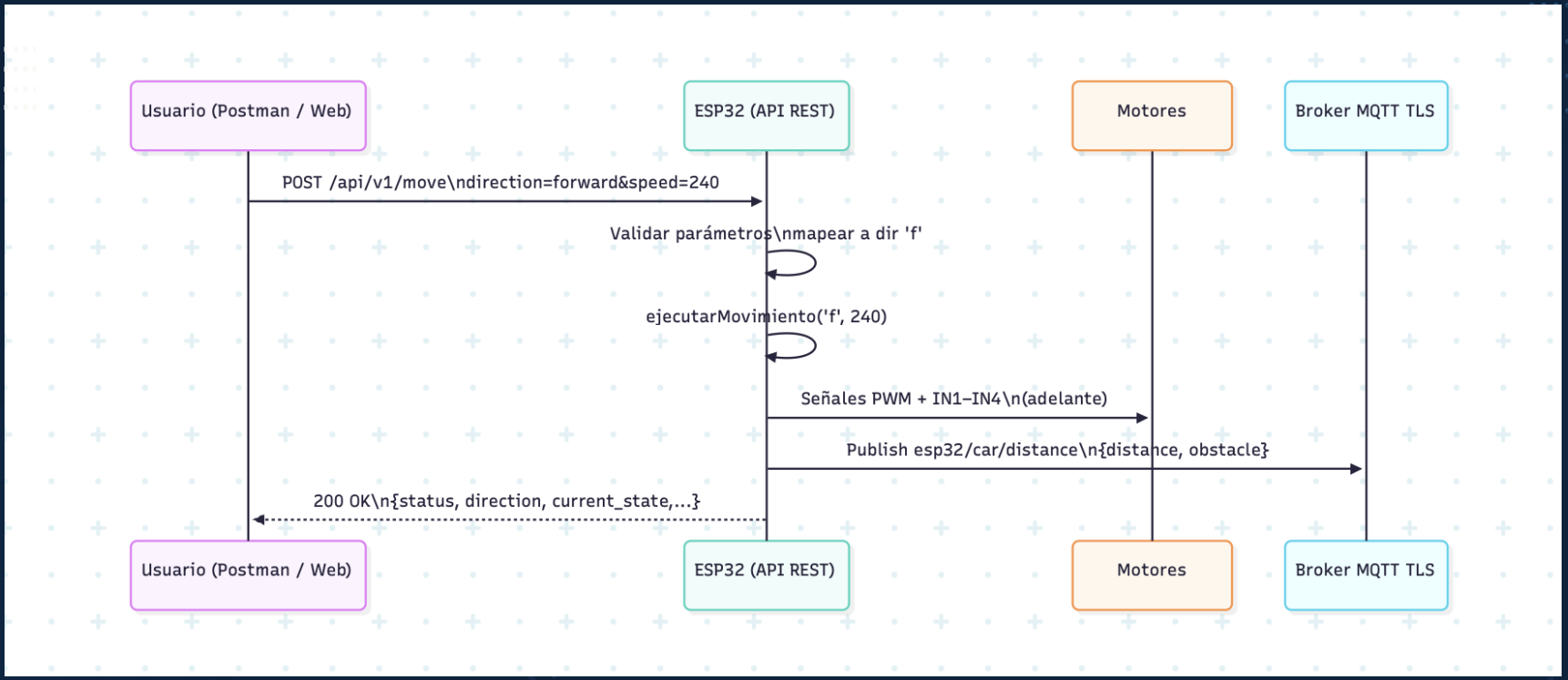


Arquitectura general del sistema

- ESP32 como cerebro principal del robot
  - Motores DC + puente H controlados por PWM (IN1–IN4, ENA, ENB)
  - Sensor ultrasónico frontal HC-SR04 y encoder en la rueda
  - Pantalla OLED 1.3" I2C (estado, velocidad, odometría, IP)
  - Servidor web + API REST embebidos en el ESP32
  - Cliente MQTT seguro hacia broker.hivemq.com:8883 (TLS)

Seguridad de comunicaciones: MQTT con TLS/SSL

- Cambio de WiFiClient a WiFiClientSecure
  - MQTT_PORT 8883 en lugar de 1883 (canal cifrado)
  - Cliente MQTT creado sobre secureClient (PubSubClient mqttClient(secureClient))
  - Configuración TLS con secureClient.setInsecure() para simplificar pruebas
  - Opción de mayor seguridad usando secureClient.setCACert(ca_cert)
  - Función reconnectMQTT() mantiene la conexión TLS activa

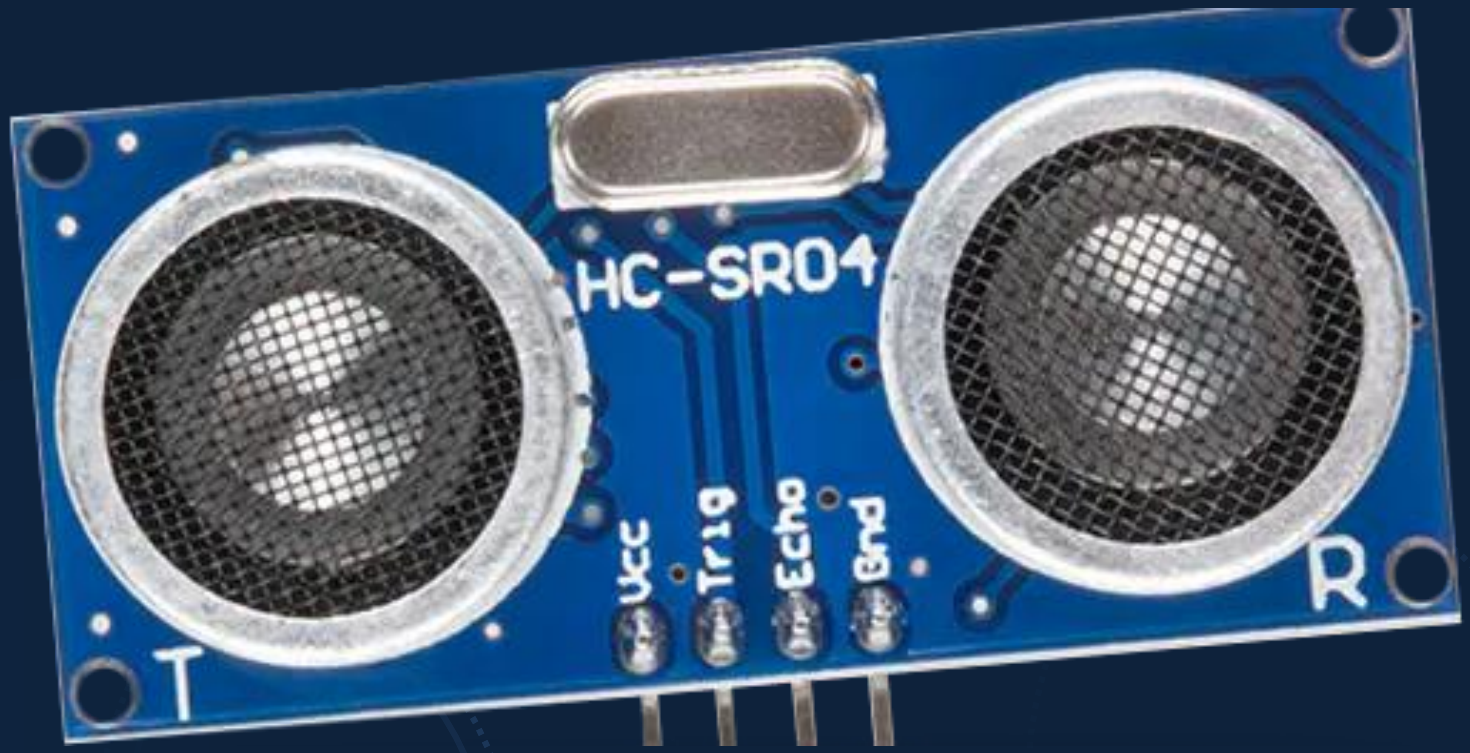


API REST v1: endpoints principales

- GET /api/v1/healthcheck
  - • Responde JSON con: wifi_connected, mqtt_connected, mqtt_tls, ip, uptime_ms
  - • Incluye current_direction, obstacle_ahead y distance_cm
  - POST /api/v1/move (también GET)
  - • Parámetros: direction (forward/backward/left/right/stop), speed (220–255)
  - • Devuelve status, direction, speed, current_state, obstacle_ahead, timestamp

Lógica de movimiento y bloqueo solo adelante

- Funciones de movimiento: adelante, atras, izquierda, derecha, stopMotors
  - ejecutarMovimiento(dir, speed) centraliza las órdenes de movimiento
  - Variable obstacleAhead indica si hay obstáculo en la trayectoria frontal
  - En adelante(): si obstacleAhead == true se frena el carro y se pasa a 'BLOQUEADO'
  - Movimientos hacia atrás y giros siguen permitidos aunque haya obstáculo



Sensor ultrasónico y seguridad con MQTT

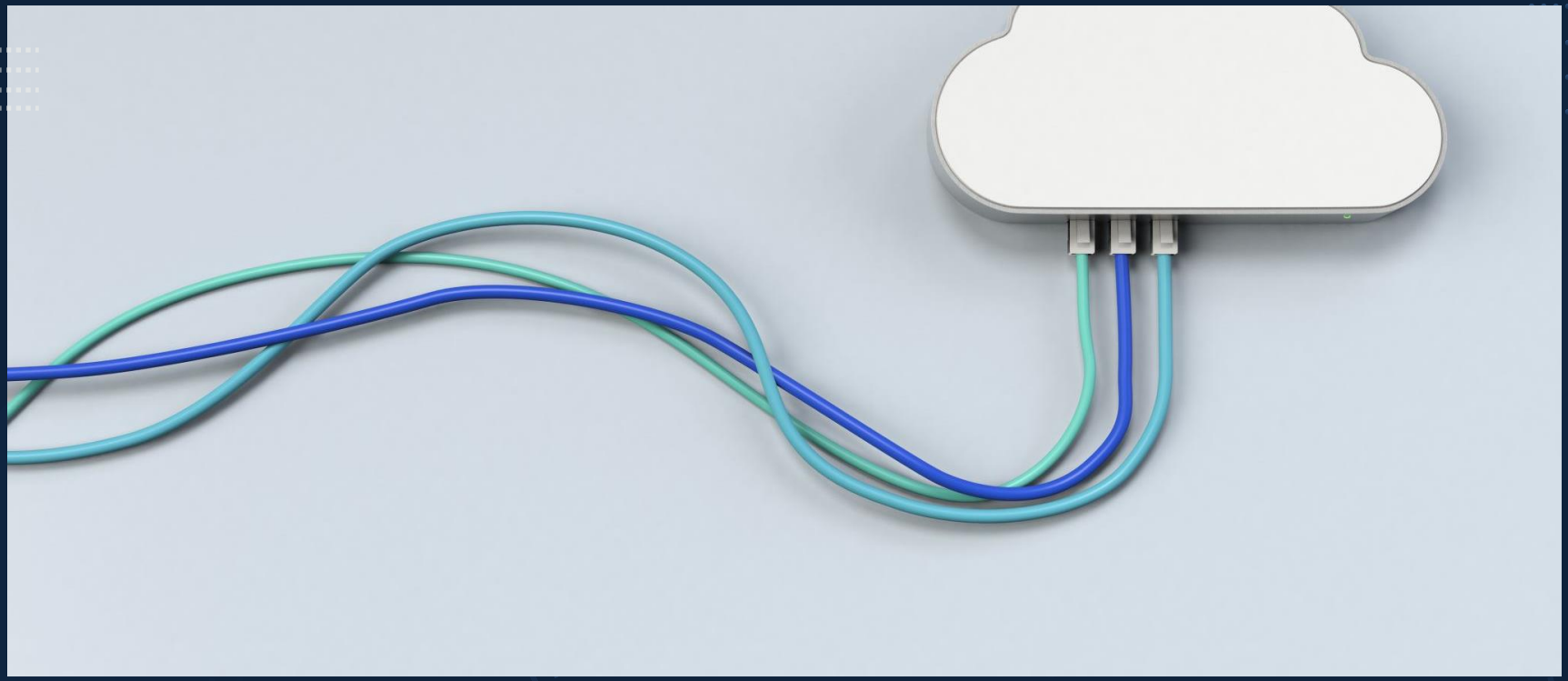
- Lectura periódica del HC-SR04 en checkProximitySafety()
  - Umbral de seguridad: SAFETY_DISTANCE = 20 cm
  - Actualización de lastUltrasonicDistance y flag obstacleAhead
  - Si cambia el estado del obstáculo se decide frenar o permitir avance
  - Publicación en MQTT (esp32/car/distance) con JSON {distance, obstacle}

Odometría, velocidad y display OLED

- Encoder en la rueda conectado a interrupción (encoderISR)
  - Cálculo de distancePerPulse con diámetro de la rueda y PULSES_PER_REVOLUTION
  - calculateDistanceMeters() obtiene los metros recorridos acumulados
  - calculateVelocity() estima la velocidad actual en km/h
  - Pantalla OLED muestra dirección, velocidad, distancia, odometría e IP
  - Modo alerta: 'OBSTÁCULO', distancia y banda 'DETENER!' parpadeando

Interfaz web y experiencia de usuario

- HTML, CSS y JavaScript embebidos en el ESP32 (PROGMEM)
  - Diseño futurista: fuente Orbitron, gradientes y efecto glass
  - Control del carro con slider de velocidad y botones de dirección
  - Consultas periódicas a /distance y /odometry para actualizar telemetría
  - Log de comandos enviados y badge de 'Conexión MQTT cifrada (TLS/SSL)'



Limitaciones, mejoras y conclusiones

- Limitaciones actuales: setInsecure() sin validación de CA, una sola vista frontal, sin autenticación en web/API
  - Mejoras posibles: añadir más sensores, autenticación y panel en la nube
  - Conclusión: integración de hardware, firmware, web UI, API REST y MQTT con TLS
  - Base sólida para evolucionar hacia un robot más autónomo, conectado y seguro