ESP32 IOT SOLUTIONS

ESP32 IoT Car Controller

Presentado por Axel Ariza, Juan Pablo Benítez, Juan Montes

Problema y requisitos

Detección de obstáculos segura

Carro sin protección

El carro 2WD carece de sistemas de seguridad, lo que puede resultar en choques y daños. Es vital implementar mecanismos de detección de obstáculos para garantizar un funcionamiento seguro.

Detección ultrasonido

Se requiere un sensor ultrasónico para medir distancias y detectar obstáculos en el camino del carro. Este sensor debe ser integrado para enviar datos a través de MQTT cifrado.

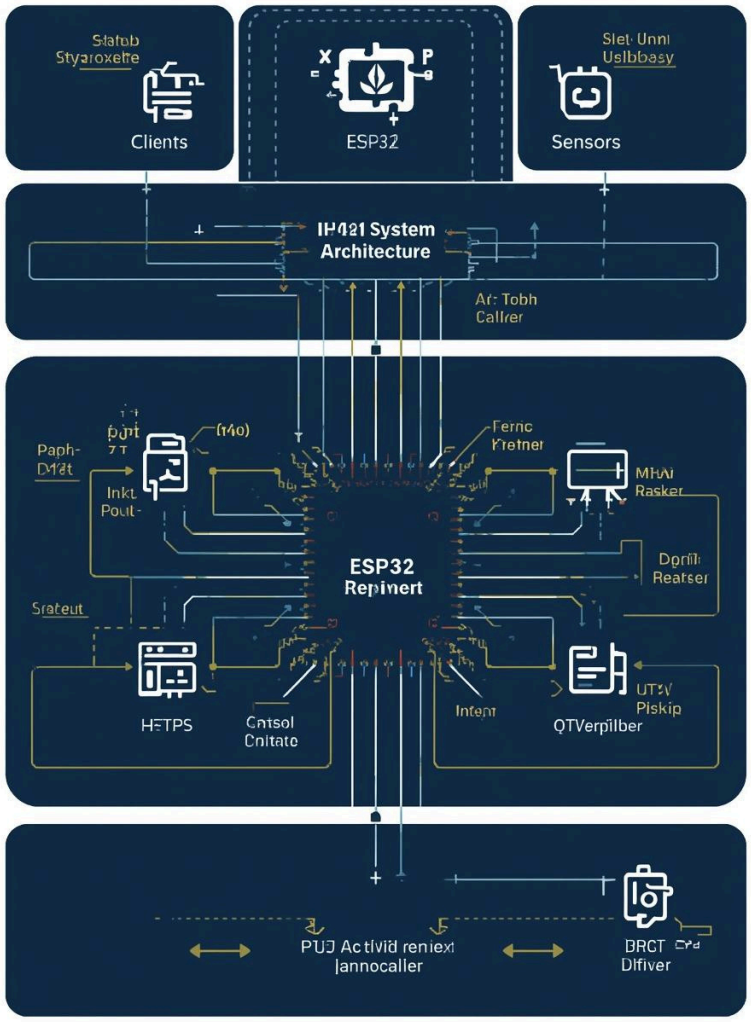
Control seguro

La comunicación remota se realiza mediante una API REST que permite a los usuarios controlar el carro. Además, se garantiza la seguridad en las transmisiones a través de métodos de cifrado.

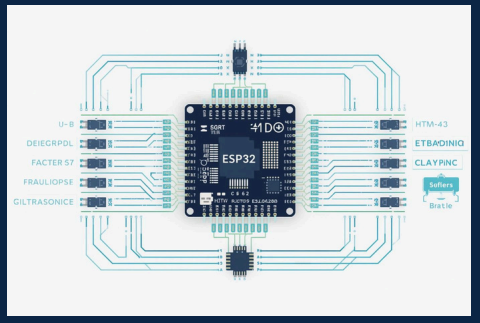
Arquitectura del Sistema

Conexión de ESP32, sensores y comunicación

La arquitectura del sistema se centra en el **ESP32** como nodo principal, integrando sensores y protocolos HTTP y **MQTT** para una comunicación eficiente y segura entre el carro y los clientes.

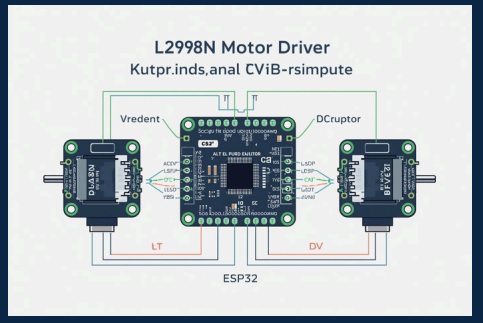


Hardware Connections



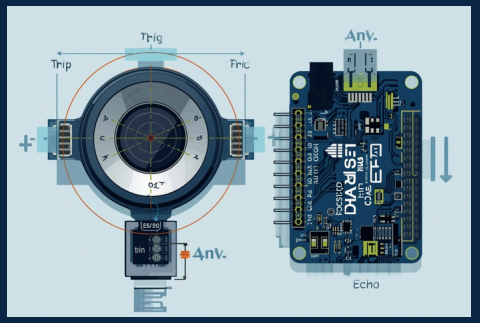
ESP32 Module

The ESP32 connects sensors and controls motor movements.



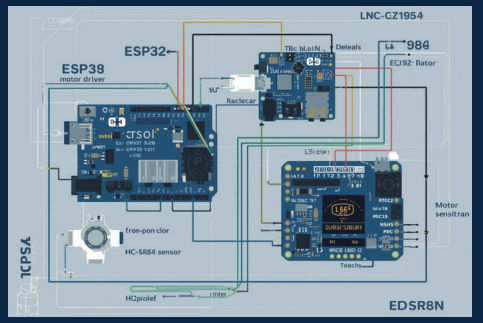
L298N Driver

L298N manages dual motor functionality with power efficiency.



Ultrasonic Sensor

This sensor detects obstacles, facilitating safe navigation.



Power Supply

Separate power sources ensure reliable motor operation.

Lógica de Control de Motores

Detección de Obstáculos y Seguridad

Control Seguro

La lógica de control asegura que el carro **evite colisiones** al combinar comandos de movimiento con la lectura de distancia, protegiendo sus componentes y el entorno.

Funciones de Movimiento

Las funciones como moveForward y stopMotors son esenciales para gestionar el movimiento del 2WD, asegurando que cada acción se base en la detección precisa de obstáculos.

Umbral de Obstáculos

Se establece un umbral de distancia, que permite al sistema **detectar obstáculos** cercanos y bloquear automáticamente comandos de movimiento que podrían resultar en colisiones.

Endpoints de la API

Control y estado del carro

Endpoint Healthcheck

El endpoint **/api/v1/healthcheck** proporciona información sobre el estado actual del carro, incluyendo métricas como RSSI, heap y distancia, asegurando que el sistema esté operativo y accesible.

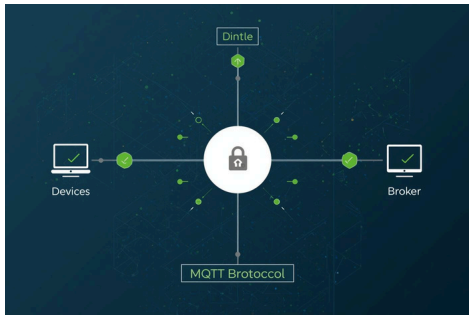
Endpoint Move

El endpoint **/api/v1/move** permite el control del movimiento del carro al recibir comandos como dirección, velocidad y duración, facilitando la interacción remota con el robot en tiempo real.

Respuestas de Error

Este endpoint gestiona respuestas claras ante errores, como el código 400 para solicitudes incorrectas y 409 para conflictos por obstáculos, garantizando una comunicación robusta y confiable.

Telemetría MQTT Segura

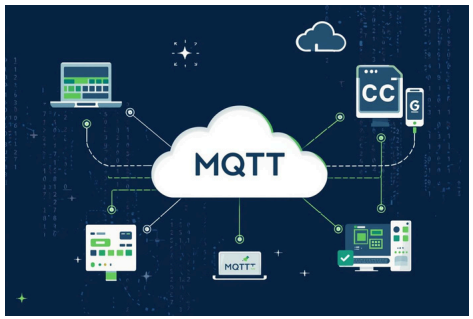


Conexión Segura

MQTT utiliza cifrado TLS para proteger los datos transmitidos.

Telemetría en Tiempo Real

Los datos del sensor se publican cada segundo, asegurando actualizaciones.



Datos Sensibles

Se implementa una conexión encriptada para proteger la información.

Actualización Continua

Se asegura que los datos se envían de manera eficiente y rápida.

Configuración del proyecto

Uso de preprocesador para parámetros

Credenciales y Pines

En config.h, se definen **WIFI_SSID** y **WIFI_PASS**, permitiendo la personalización de la conexión WiFi sin modificar el código principal. Esto simplifica la gestión de credenciales en el proyecto.

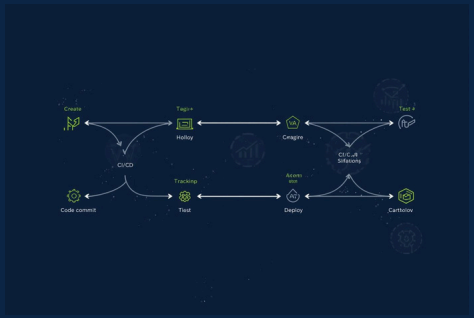
Parámetros de Diseño

Se configuran macros para **MQTT_SERVER**, **MQTT_PORT** y tópicos. Esta modularidad permite ajustar los parámetros de comunicación sin afectar el resto del sistema, favoreciendo la escalabilidad futura.

Fácil Mantenimiento

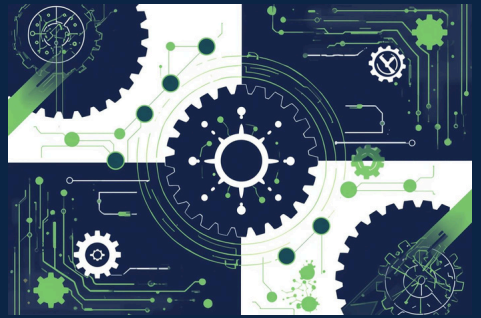
La centralización de configuraciones en config.h facilita la recompilación con diferentes valores. Esto permite una integración más eficiente con herramientas como **arduino-cli** y favorece el uso de **CI/CD**.

CI/CD Pipeline Automation



Automated Builds

Every push triggers a build in the pipeline.



Quality Assurance

Static analysis and testing ensure code reliability.

Release Management

Automated releases facilitate version tracking and deployment.

Continuous Improvement

Feedback loops enhance and refine project outcomes.

Limitaciones y conclusiones

Mejoras y aprendizajes futuros

Interfaz gráfica

Aún no se ha implementado una **interfaz gráfica completa**, solo se proporciona una API lista para usar, lo que limita la interacción directa del usuario con el carro 2WD.

Sensores limitados

Actualmente, el sistema utiliza **un único sensor frontal**, lo que impide la implementación de técnicas como SLAM o mapeo 2D, limitando la capacidad del robot para navegar en entornos complejos.

Aprendizajes clave

Este proyecto ha permitido **integrar conceptos** fundamentales como HTTP, MQTT TLS, detección de obstáculos y CI/CD, proporcionando una base sólida para futuras mejoras y extensiones del sistema.