Documentación evidencias y endpoints

Juan Esteban Rodriguez

Sebastían Castellanos

Caren Piñeros

Universidad de La Sabana

Octubre, 2025

Figura1

La figura demuestra la salida del comando ipconfig ejecutado en el computador cliente. Se observa la interfaz inalámbrica Wi-Fi con la dirección IPv4 10.217.41.109, la misma que aparece en los registros del ESP32 al recibir comandos. Esto confirma que ambos dispositivos el computador y el ESP32 se encuentran conectados a la misma red local, requisito esencial para que las peticiones HTTP sean recibidas correctamente por el servidor del carro.

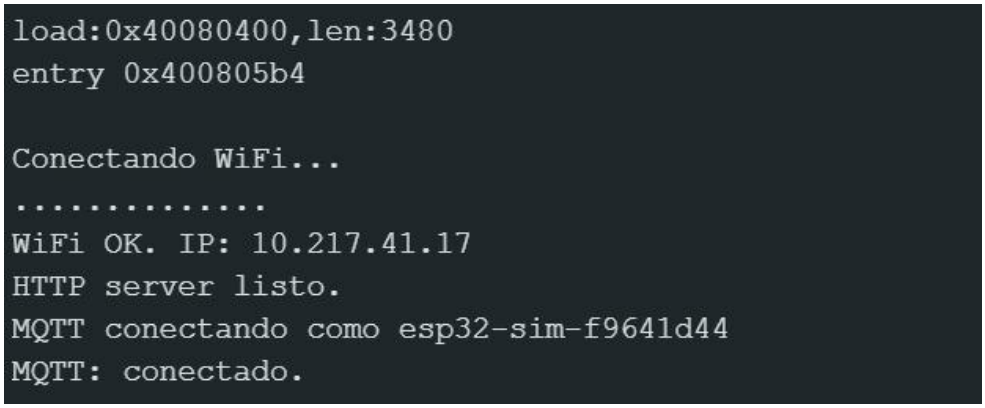

Figura2

Se muestra la salida por consola del ESP32 al iniciar el programa. Primero, el dispositivo establece la conexión WiFi y obtiene una dirección IP asignada (10.217.41.17), lo que confirma el acceso a la red. Posteriormente, se indica que el servidor HTTP ha sido inicializado correctamente, permitiendo recibir peticiones para controlar el carro. Finalmente, se observa la conexión al broker MQTT bajo el identificador *esp32-sim-f9641d44*, confirmando la comunicación activa mediante el mensaje "MQTT: conectado".

```
==== COMANDO RECIBIDO ====
DIR: forward | SPEED: 60% | DURATION: 2000 ms | CLIENT_IP: 10.217.41.109
MQTT publish: OK
==========================
>>> FIN DE ACCION (auto-stop) <<<
```

Figura3

En esta figura se muestra la respuesta del ESP32 tras recibir una instrucción enviada mediante una petición HTTP. El comando recibido indica la dirección del movimiento ("forward"), una velocidad del 60 % y una duración de 2000 ms (2 s), junto con la dirección IP del cliente que realizó la solicitud. El mensaje "MQTT publish: OK" confirma que la acción fue notificada correctamente mediante el protocolo MQTT. Finalmente, se ejecuta el mecanismo de seguridad "auto-stop", que detiene el movimiento al finalizar el tiempo indicado, evitando desplazamientos indefinidos.

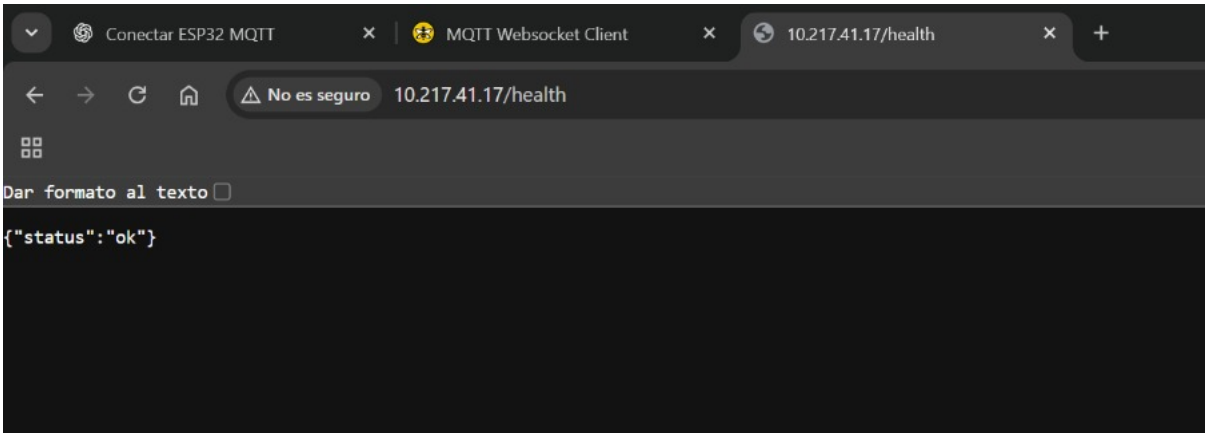

Figura4

En esta captura se visualiza el resultado de acceder al endpoint /health del servidor HTTP implementado en el ESP32. La respuesta en formato JSON {"status":"ok"} confirma que el servidor está en funcionamiento y responde correctamente a las peticiones. Este endpoint de verificación permite comprobar el estado del sistema sin ejecutar movimientos, garantizando que el carro esté disponible y conectado a la red antes de enviar comandos de control.

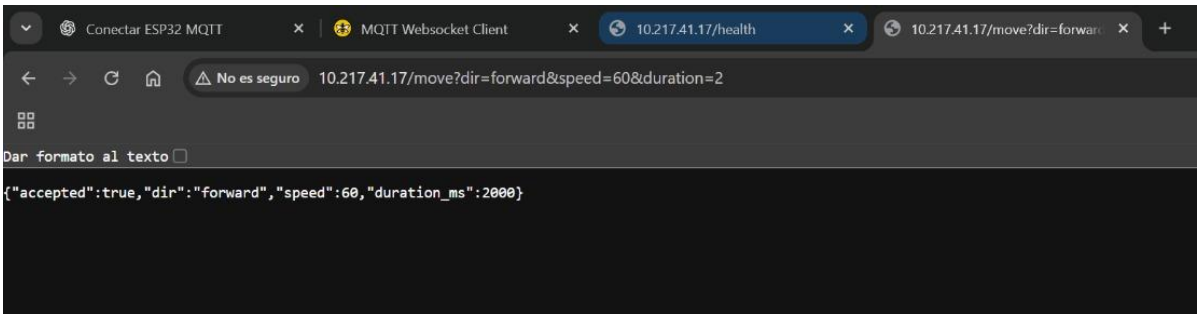

Figura5

En esta cptura podemos observar el uso del endpoint principal del servidor HTTP para controlar el carro. La solicitud enviada a la URL incluye los parámetros de dirección, velocidad y duración del movimiento. La respuesta JSON {"accepted":true,"dir":"forward","speed":60,"duration_ms":2000} confirma que el comando fue recibido y procesado correctamente, convirtiendo la duración expresada en segundos a milisegundos para su ejecución. Este endpoint centraliza las instrucciones de control, reemplazando el uso de Bluetooth por comunicación vía HTTP.

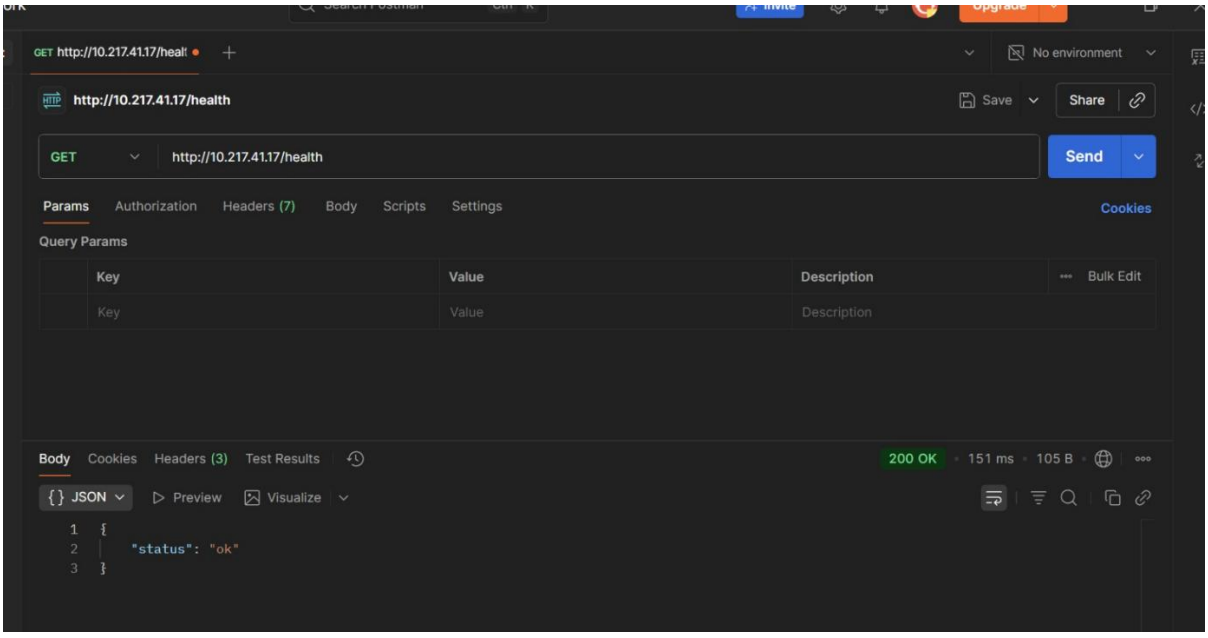

Figura6

En esta figura se muestra la verificación del endpoint de *healthcheck* a través de la herramienta Postman, utilizando una petición HTTP de tipo **GET** a la dirección http://10.217.41.17/health. La respuesta obtenida, con código **200 OK**, devuelve el mensaje JSON {"status":"ok"}, lo que confirma que el servidor HTTP implementado en el ESP32 se

encuentra operativo y disponible para recibir comandos. Esta prueba permite validar la correcta configuración del servidor y la conectividad entre el cliente y el dispositivo.

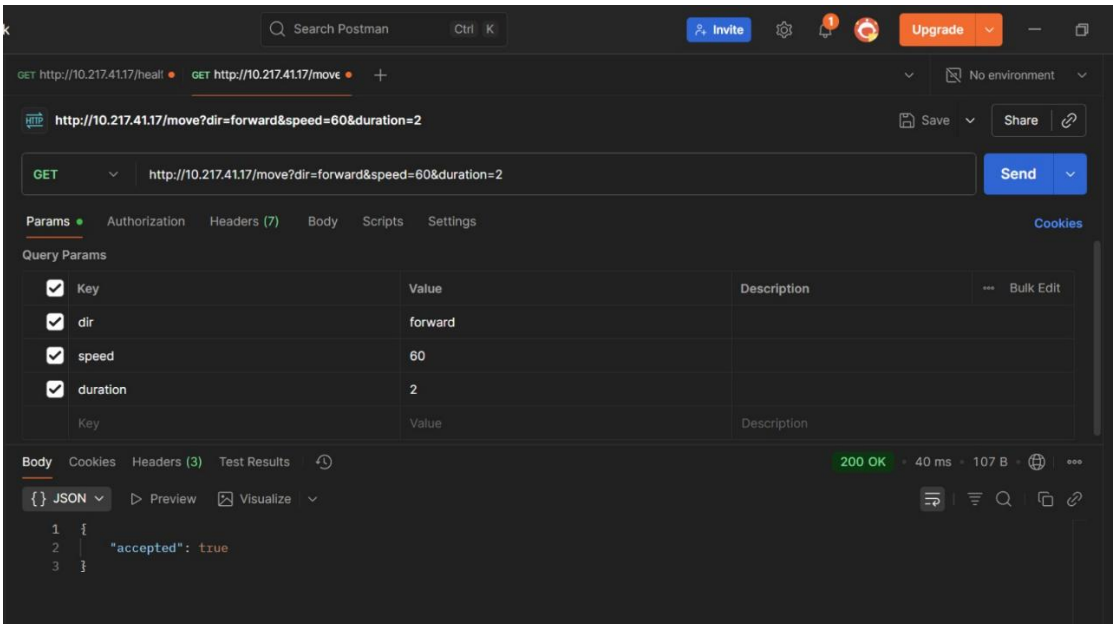

Figura7

La figura evidencia la ejecución de una petición HTTP de tipo **GET** desde Postman hacia el endpoint http://10.217.41.17/move, con los parámetros dir=forward, speed=60 y duration=2. Estos valores indican que el carro debe avanzar hacia adelante a un 60 % de velocidad durante 2 segundos. La respuesta JSON {"accepted": true} y el código **200 OK** confirman que la instrucción fue recibida correctamente por el servidor del ESP32, validando la funcionalidad del control de movimiento vía HTTP.