UNIVERSIDAD TÉCNICA DE MACHALA

Documentación del Proyecto

Nombre del Proyecto:
Dashboard para Sistema de Parqueadero Inteligente

Nombre de los Estudiantes:
Castillo Ortega Brandon Estefano
Peña Paladines Martin Alexander

Asignatura:
Internet de las Cosas

6 de junio de 2025

Índice

1. Introducción

La gestión de estacionamientos es un componente crítico de la infraestructura urbana moderna. Este proyecto aborda este desafío mediante un sistema dual: un dashboard web para el monitoreo visual y un bot de Telegram para la interacción y control remotos. Utilizando Node-RED como plataforma central, el sistema integra sensores de 12 espacios de parqueo, presenta su estado en tiempo real en una interfaz gráfica y, adicionalmente, permite a los usuarios consultar y simular estados a través de comandos enviados a un bot de Telegram, creando una solución de gestión completa, accesible y eficiente.

2. Objetivo

Diseñar y desarrollar un sistema integral para la gestión de un parqueadero de 12 slots, que combine un dashboard de monitoreo visual en tiempo real con un bot de Telegram interactivo para consultas y control.

2.1. Objetivos Específicos

- Desarrollar un flujo en Node-RED para recibir y centralizar los datos del estado de 12 sensores de parqueo vía MQTT.
- Implementar un dashboard web que visualice el estado (Libre/Ocupado) de cada slot y muestre un contador de disponibilidad.
- Construir un bot de Telegram capaz de procesar comandos de los usuarios.
- Implementar la lógica para que el bot responda a consultas sobre el estado general del parqueadero (`/status`) y el estado de slots individuales (`/slot<N>`).
- Añadir una funcionalidad de simulación que permita cambiar el estado de un slot mediante un comando en Telegram (`/set<N><estado>`), actualizando tanto el dashboard como el estado interno del sistema.

3. Requerimientos Técnicos

3.1. Hardware Requerido

- 12 Sensores de ultrasonido o infrarrojos para detectar vehículos.
- 12 Microcontroladores (ej. ESP8266/ESP32) para publicar datos vía MQTT.
- Dispositivo controlador (Computadora o Raspberry Pi) para ejecutar Node-RED.

3.2. Software y Plataformas

- Node-RED para el desarrollo del flujo.
- Broker MQTT (ej. `broker.hivemq.com`) para la comunicación.
- Aplicación de Telegram para la interacción con el bot.
- Navegador web para visualizar el dashboard.

3.3. Dependencias de Node-RED

- `node-red-dashboard`: Para la creación de la interfaz gráfica web.
- `node-red-contrib-telegrambot`: Para la integración con la API de Telegram.
- Nodos base: `mqtt in`, `function`, `switch`, `inject`, `ui_template`.

4. Metodología Empleada

1. **Configuración del Entorno:** Instalación de Node.js, Node-RED y las dependencias necesarias (`dashboard` y `telegrambot`).
2. **Diseño del Flujo de Datos (MQTT):** Creación de 12 nodos `mqtt in` para recibir los datos de los sensores. Se centralizó la lógica en un único nodo de función para procesar todos los mensajes.
3. **Desarrollo del Dashboard:** Implementación del nodo `ui_template` con HTML, CSS y JavaScript para la visualización del estado de los 12 slots.
4. **Integración del Bot de Telegram:** Registro de un nuevo bot en Telegram a través de BotFather para obtener el token de la API. Configuración de los nodos `telegram receiver` y `sender`.
5. **Lógica de Comandos:** Desarrollo de un nodo `switch` para enrutar los comandos del bot y nodos `function` para manejar la lógica de cada comando (`/status`, `/slot`, `/set`).
6. **Gestión de Estado:** Implementación de un sistema de reinicio con un nodo `inject` para limpiar el estado del parqueadero al desplegar el flujo, asegurando consistencia.
7. **Pruebas Integrales:** Verificación del correcto funcionamiento del dashboard, la respuesta del bot a los comandos y la sincronización entre las actualizaciones simuladas por el bot y la interfaz visual.

5. Descripción Detallada del Flujo

El sistema se divide en tres componentes lógicos principales que operan de forma interconectada: el monitoreo visual, la interacción por Telegram y la gestión de estado.

5.1. Monitoreo y Dashboard Visual

Esta parte del flujo se encarga de recibir los datos de los sensores y presentarlos en la web.

- **Nodos `mqtt in`:** 12 nodos se suscriben a los tópicos `parking/slot1` a `parking/slot12`.
- **Nodo `function` (Actualizar Estado y UI):** Es el cerebro del sistema. Todos los mensajes MQTT pasan por este único nodo. Realiza dos tareas:
  1. Actualiza una variable de contexto de flujo (`flow.parkingStatus`) que almacena el estado actual de todos los slots.
  2. Formatea el `msg.payload` en un objeto JSON estandarizado para que el nodo `ui_template` pueda procesarlo.
- **Nodo `ui_template`:** Renderiza la interfaz gráfica del parqueadero, actualizando dinámicamente el estado de cada slot cuando recibe un nuevo mensaje.

5.2. Interacción mediante Bot de Telegram

Esta sección dota al sistema de una interfaz de control conversacional.

- **Nodo `telegram receiver`:** Recibe todos los mensajes enviados al bot.
- **Nodo `switch` (Enrutador de Comandos):** Analiza el contenido del mensaje y lo dirige a la función correspondiente según el comando detectado (`/status`, `/slot`, o `/set`).
- **Nodos `function` (Lógica de Comandos):**
  - *Reporte General:* Consulta la variable `flow.parkingStatus` y genera un resumen de slots libres y ocupados.
  - *Reporte Slot Individual:* Extrae el número de slot del comando y reporta su estado específico.
  - *Simular Sensor / Set Estado:* Permite a un administrador forzar un cambio de estado. Tiene dos salidas: una que publica un mensaje MQTT (simulando un sensor real) para actualizar todo el sistema, y otra que envía un mensaje de confirmación al usuario de Telegram.
- **Nodo `telegram sender`:** Envía las respuestas generadas por los nodos de función de vuelta al usuario a través de Telegram.

5.3. Gestión de Estado del Sistema

Este pequeño pero importante componente asegura que el sistema inicie en un estado limpio.

- **Nodo `inject`:** Se activa una sola vez al desplegar el flujo.
- **Nodo `function` (Limpiar Memoria de Estado):** Reinicia la variable `flow.parkingStatus` a un objeto vacío, evitando que el sistema retenga datos de una sesión anterior.

6. Capturas del Sistema

A continuación se presentan las interfaces y componentes clave del proyecto.

6.1. Flujo General en Node-RED

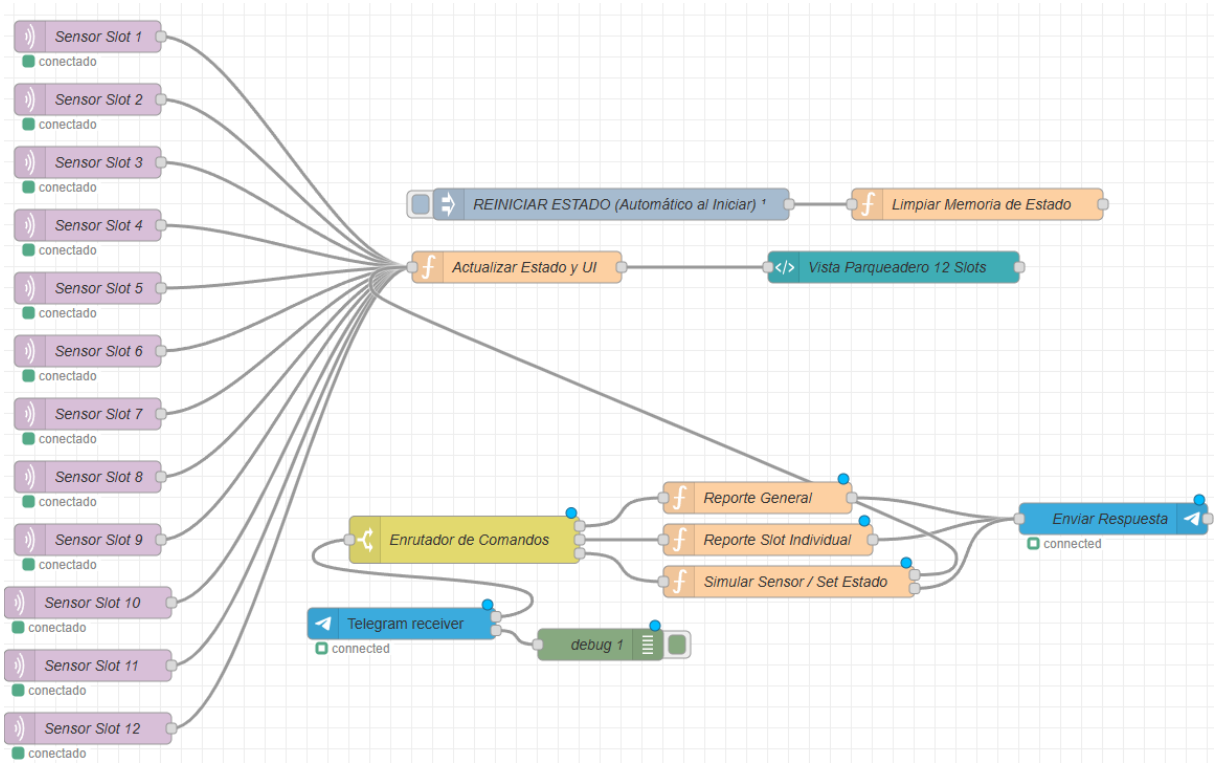

Figura 1: Vista completa del flujo en Node-RED, mostrando la integración de MQTT, el dashboard y el bot de Telegram.

6.2. Dashboard del Parqueadero

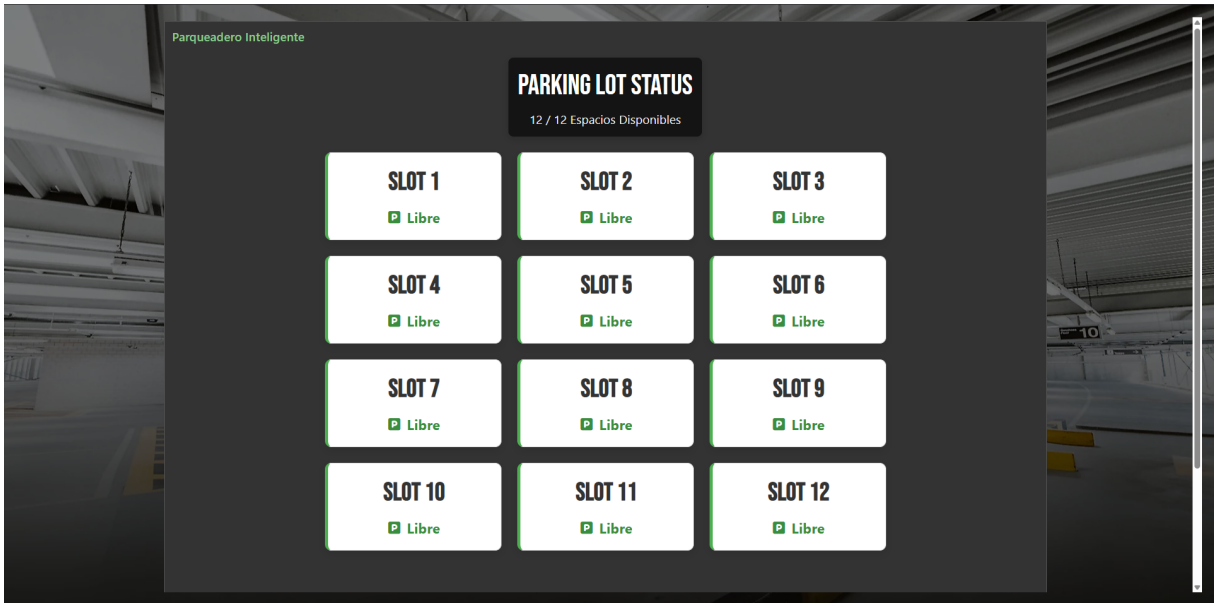

Figura 2: Interfaz web del dashboard, mostrando el estado en tiempo real de los 12 slots.

6.3. Nodo Principal de Lógica

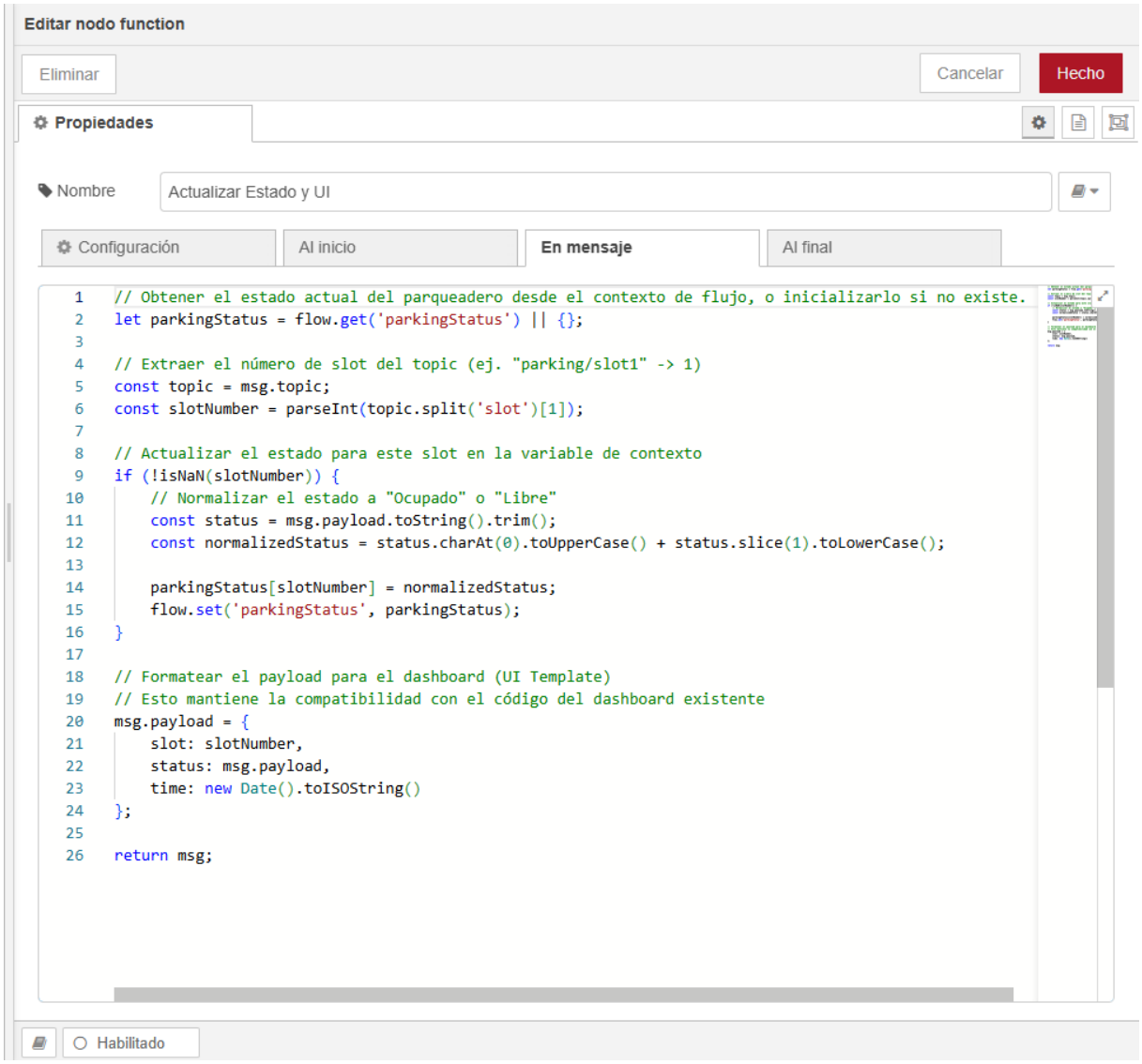

Figura 3: Detalle del código dentro del nodo de función .ªctualizar Estado y UI", que centraliza la lógica de procesamiento de datos.

6.4. Interacción con el Bot de Telegram

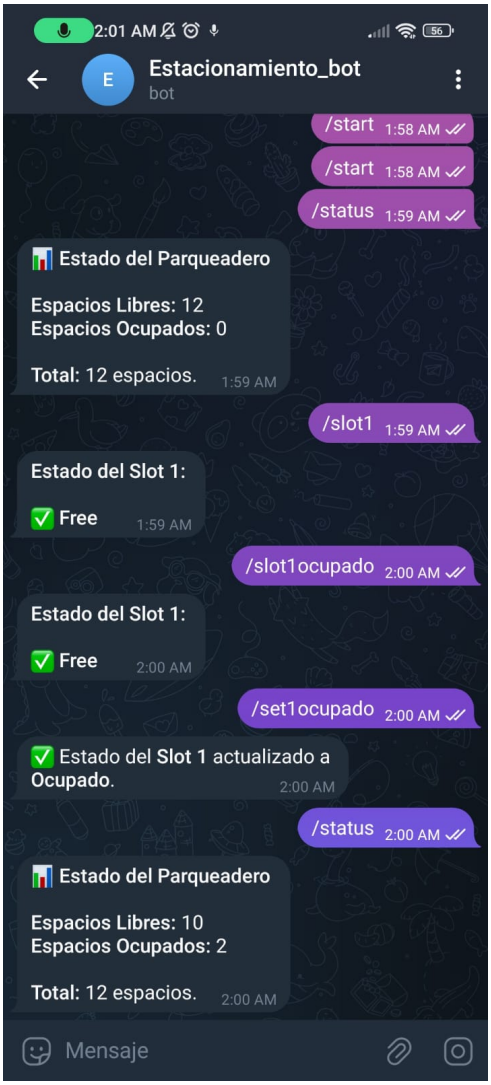

Figura 4: Ejemplos de interacción con el bot para consultar estados y simular cambios.

7. Conclusiones

Este proyecto culminó con éxito en la creación de un sistema de gestión de parqueadero robusto y multifacético. La combinación de un dashboard visual intuitivo con un bot de Telegram interactivo ofrece una solución completa que satisface tanto las necesidades de monitoreo pasivo como de control activo.

La arquitectura centralizada, donde un único nodo de función gestiona el estado del sistema, demostró ser eficiente y fácil de mantener. La integración del bot de Telegram no solo añade una capa de conveniencia para el usuario final, sino que también proporciona una potente herramienta de administración para simular eventos y verificar el estado del sistema de forma remota.

El proyecto destaca la capacidad de Node-RED para orquestar rápidamente sistemas complejos de IoT, integrando diferentes protocolos y servicios (MQTT, HTTP, API de Telegram) en un único entorno de desarrollo visual.

8. Bibliografía

Referencias

[1] "Node-red-dashboard," *Nodered.org*. [En línea]. Disponible: https://flows.nodered.org/node/node-red-dashboard. [Consultado: 06-jun-2025].

[2] "node-red-contrib-telegrambot," *Nodered.org*. [En línea]. Disponible: https://flows.nodered.org/node/node-red-contrib-telegrambot. [Consultado: 06-jun-2025].

[3] "HiveMQ Public MQTT Broker," *Hivemq.com*. [En línea]. Disponible: https://www.hivemq.com/public-mqtt-broker/. [Consultado: 06-jun-2025].

[4] "Node-RED Documentation," *Nodered.org*. [En línea]. Disponible: https://nodered.org/docs/. [Consultado: 06-jun-2025].