Instituto Politécnico Nacional
Escuela superior de Cómputo

Sistema de estacionamiento haciendo uso de Raspberry & Mosquitto MQTT

Por:
Anguiano Rodea Michelle
Ayala Fuentes Sunem Gizeht
Torres Covarrubias Rubén

Grupo: 6CM3

¿Qué se hizo?

Se diseñó e implementó un sistema de control para un estacionamiento inteligente utilizando una placa de desarrollo ESP32 y una Raspberry Pi como servidor central. El sistema permite:

- Detectar la presencia o ausencia de vehículos en tiempo real.
- Visualizar el estado del estacionamiento (lugares disponibles u ocupados) mediante una interfaz gráfica.

¿Qué se utilizó?

- Microcontrolador ESP32
- 4 sensores ultrasónicos HC-SR04
- Raspberry Pi 4
- Monitor (para configuración inicial)

¿Qué software se utilizaron?

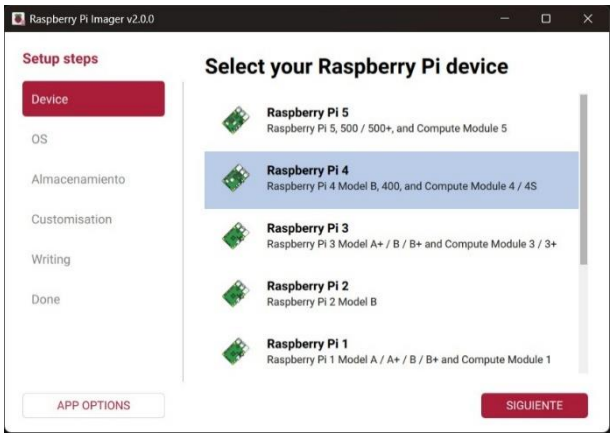
- Arduino IDE
- Node-RED
- Eclipse Mosquitto
- Raspberry Pi Imager
- RealVNC Viewer & VNC Server

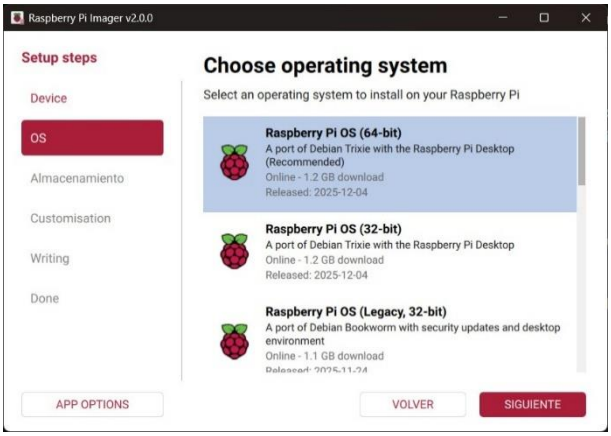
Desarrollo de lo que se hizo

1. Configuración del Entorno *(Raspberry Pi Imager)*

1.1. Selección del Sistema Operativo

Se seleccionó la arquitectura de hardware **Raspberry Pi 4** y se instaló el sistema operativo **Raspberry Pi OS (64-bit)**. Se optó por la versión de 64 bits para maximizar la compatibilidad y el rendimiento con los servicios de middleware (Mosquitto y Node-RED) a implementar.

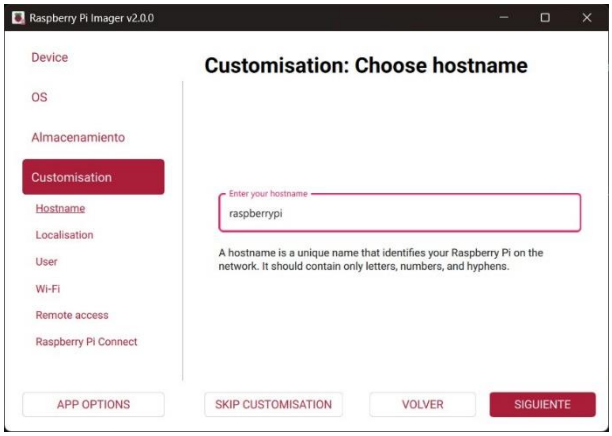


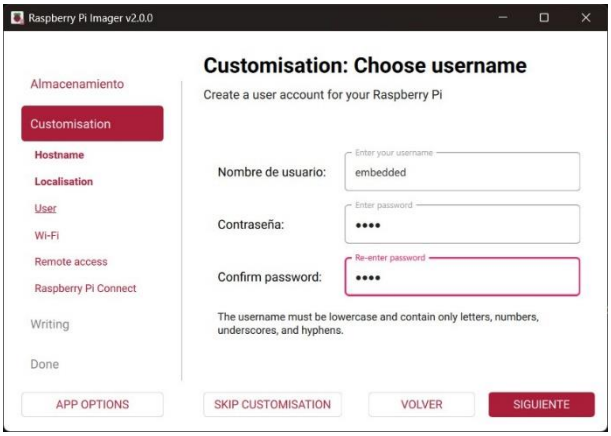


1.2. Configuración Avanzada

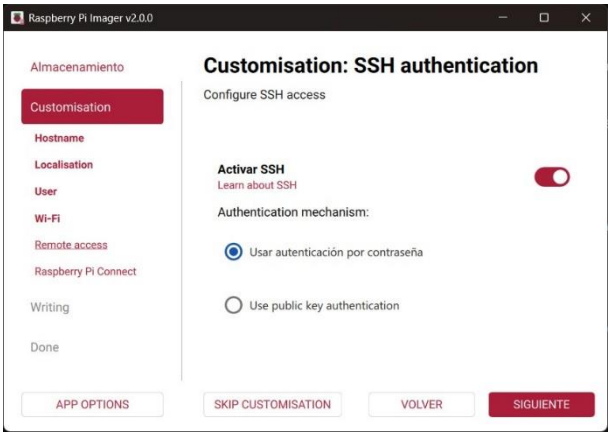
Durante el proceso de creación de la imagen, se aplicaron las siguientes configuraciones para facilitar el despliegue:

- **Identificación y Seguridad:** Se asignó el *hostname* raspberrypi para su fácil localización en la red local. También, se creó el *usuario* dedicado llamado embedded con privilegios de administrador.

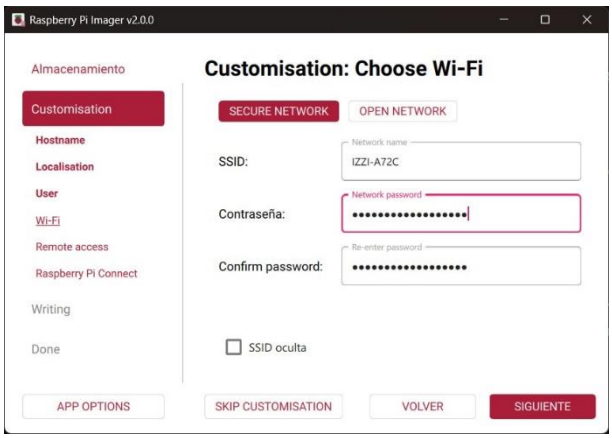


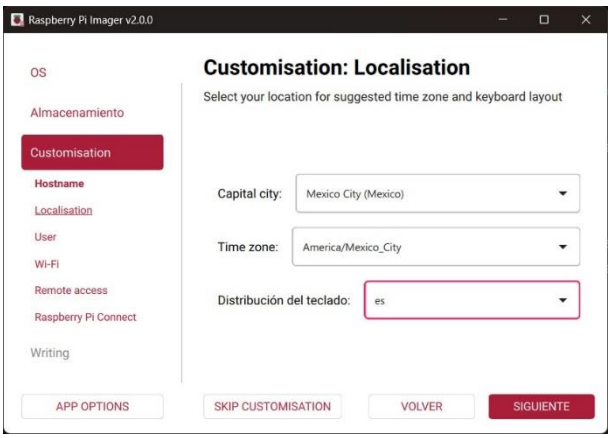


- **Habilitación de SSH (Secure Shell):** Se habilitó el servidor SSH y la autenticación por contraseña para permitir el acceso remoto a la terminal desde el primer arranque.



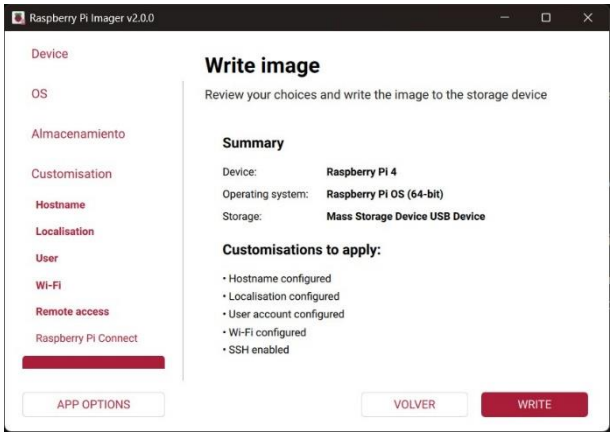
- **Conectividad:** Se preconfiguraron las credenciales de la red Wi-Fi local y la zona horaria (Mexico City) para asegurar la sincronización automática del sistema.





1.3. Resultados

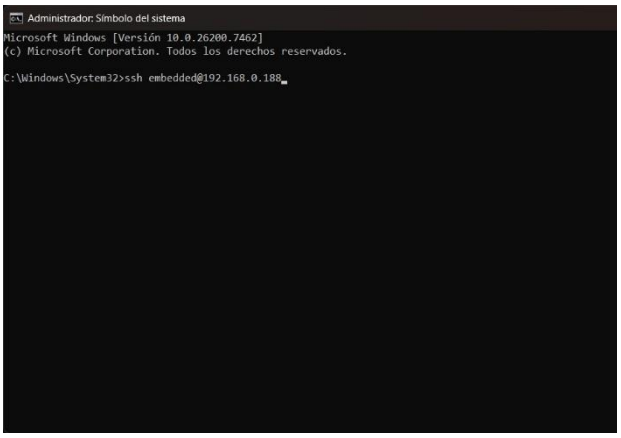
Se nos presenta al final de la configuración en forma de resumen todos y cada uno de los puntos mencionadas previamente, esto con la finalidad de que estemos de acuerdo con las características que se usaran.

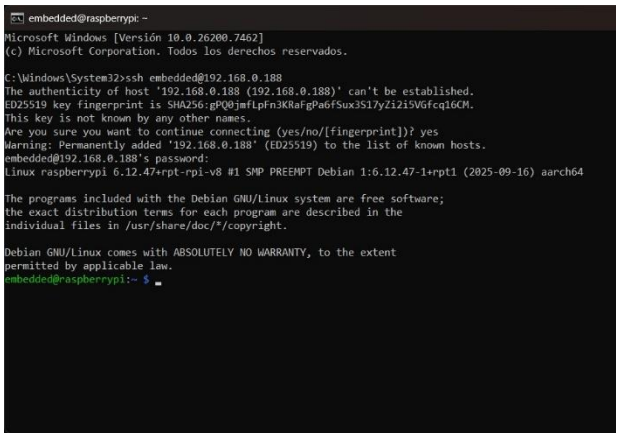


2. Establecimiento de Comunicación Remota (SSH)

Tras el primer arranque, se estableció comunicación mediante el protocolo SSH. Esto permite operar la Raspberry Pi en modo *Headless* (sin monitor).

- **Nota 1:** En el primer inicio, puede ser necesario conectar un monitor vía HDMI para asegurar que el entorno gráfico cargue correctamente si se planea usar VNC posteriormente, o forzar la salida HDMI desde la configuración.
- **Acceso:** Desde la terminal del equipo anfitrión, se realizó la conexión apuntando a la dirección IP asignada por el router (preferentemente IPv4) o mediante el *hostname*.



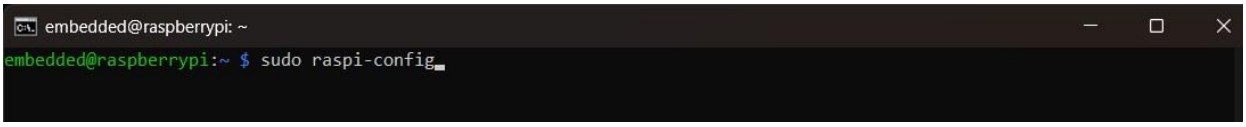


3. Habilitación de Escritorio Remoto (VNC Server)

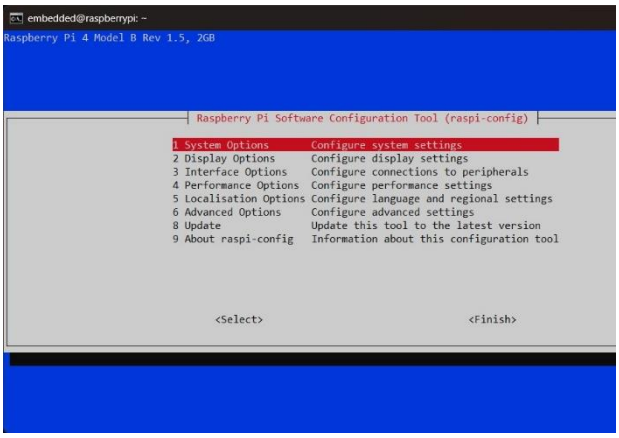
Dado que el proyecto requiere visualizar el Dashboard localmente y gestionar herramientas gráficas, se procedió a habilitar el servidor VNC (*Virtual Network Computing*). Esto permite visualizar el escritorio completo de la Raspberry PI en la computadora de desarrollo, eliminando definitivamente la necesidad de un monitor físico HDMI.

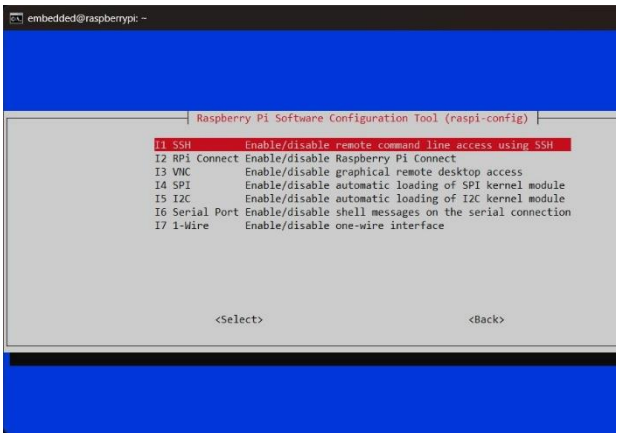
Para realizar esta configuración sin interfaz gráfica previa, se utilizó la herramienta de administración por consola:

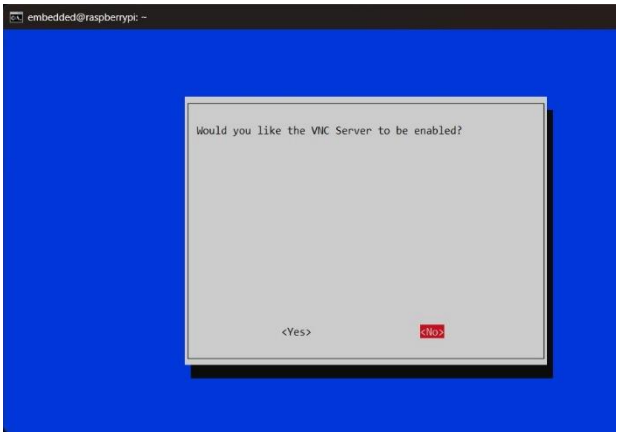
- **Ejecución de Herramienta de Configuración:** A través de la conexión SSH establecida en el paso anterior, se ejecutó el comando sudo raspi-config. Esto desplegó el menú de configuración del sistema a bajo nivel.

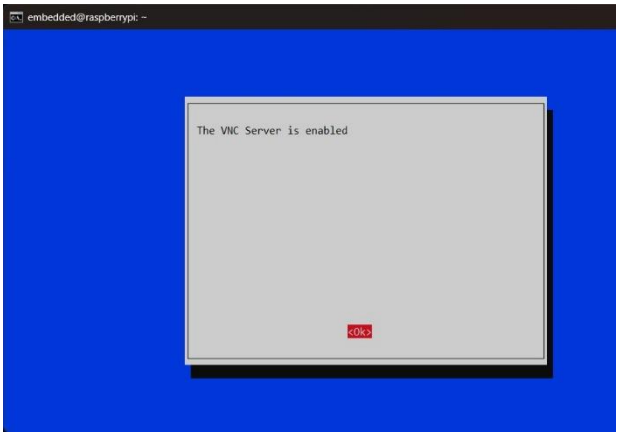


- **Activación del Servicio:** Dentro del menú *2 Interface Options*, se habilitó el servicio **I3 VNC**. Esto inicia el servidor gráfico virtual automáticamente en cada arranque. Al final nos preguntara si queremos habilitar **VNC Server** y le decimos que sí, confirmándonos que quedo habilitado.

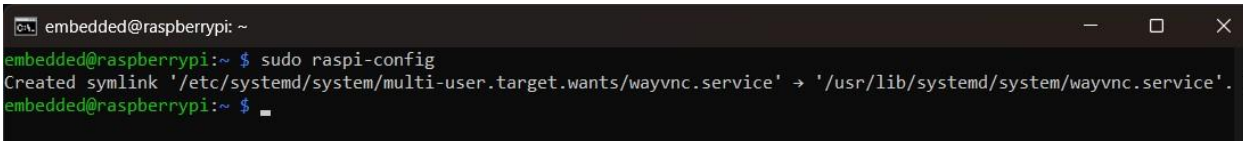








- **Resultado:** Con esto ahora podemos conectarnos de forma remota a la interfaz gráfica (GUI) de la Raspberry PI, sin necesidad de algún monitor o pantalla externo.



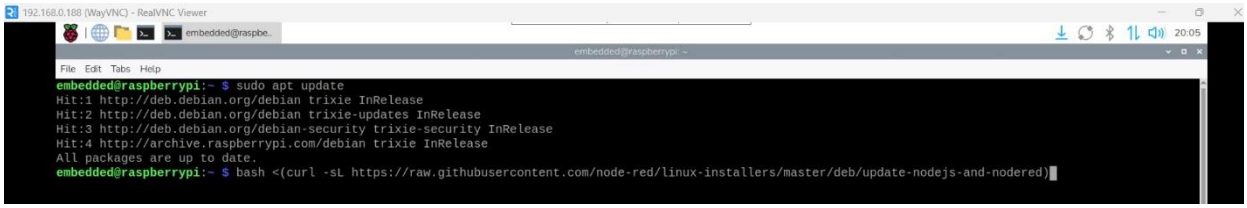
Nota 2: *Se debe tener instalado en el equipo anfitrión [Real VNC Viewer](), para acceder después a la interfaz gráfica (GUI).*

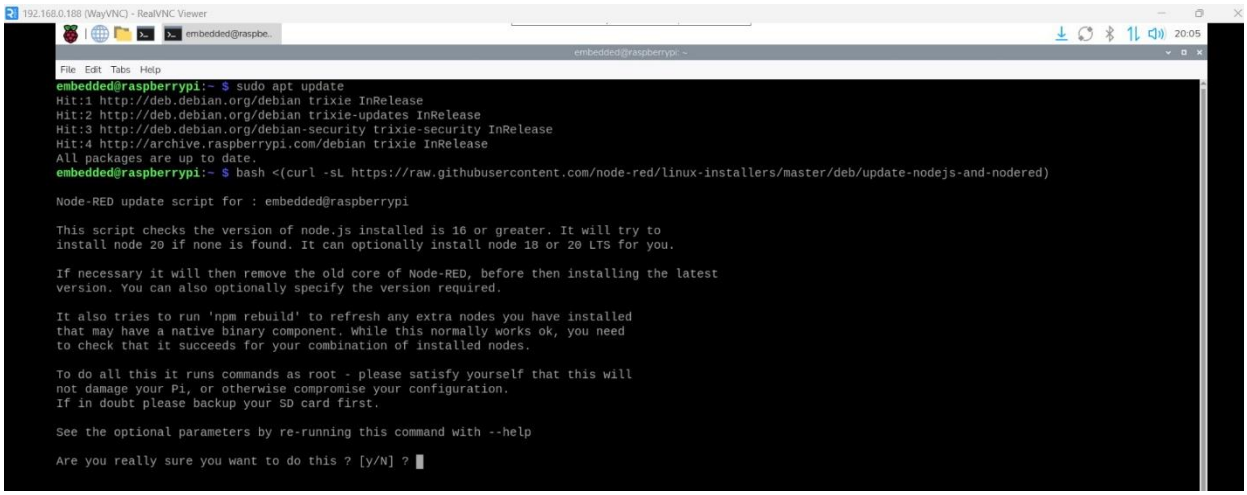
4. Despliegue de la Plataforma (Node-RED)

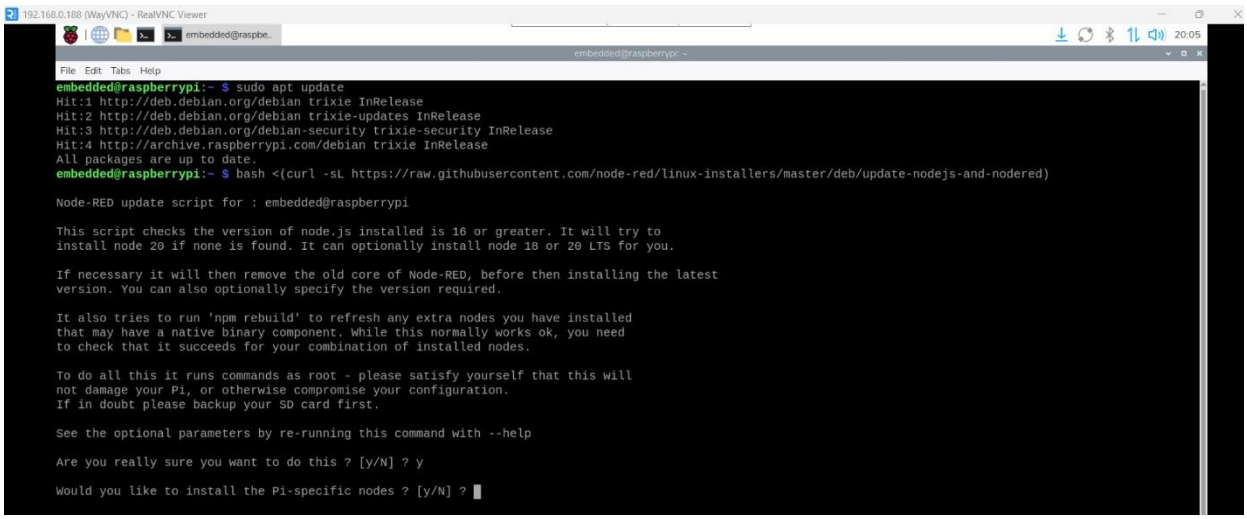
Node-RED se seleccionó como el núcleo de integración *(Middleware)* para procesar los mensajes MQTT y presentar el Dashboard.

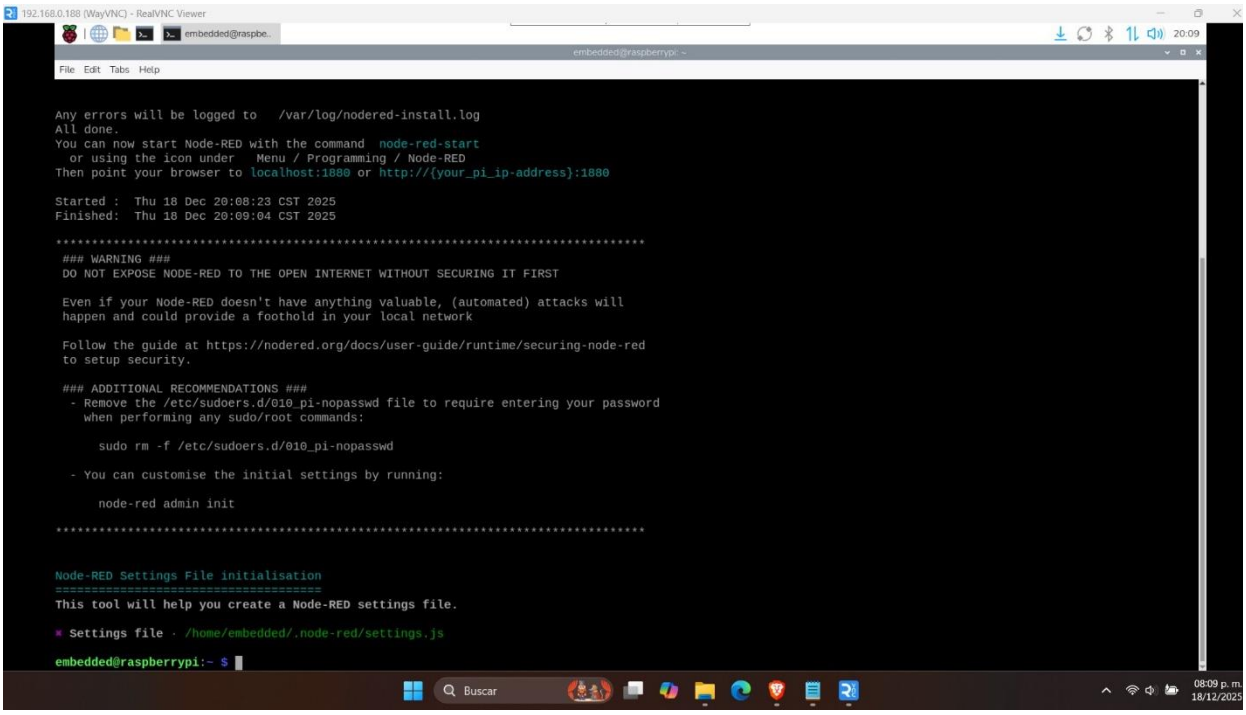
- **Instalación Automatizada:** Se utilizó el script oficial de la comunidad para garantizar la instalación de las versiones más estables de Node.js y npm:
  - **Comando:** bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
  - **Justificación:** Este método garantiza la instalación de la última versión estable de **Node.js** y **npm**, además de configurar automáticamente el servicio de sistema (systemd) para que Node-RED se

inicie automáticamente tras un reinicio o fallo de energía.





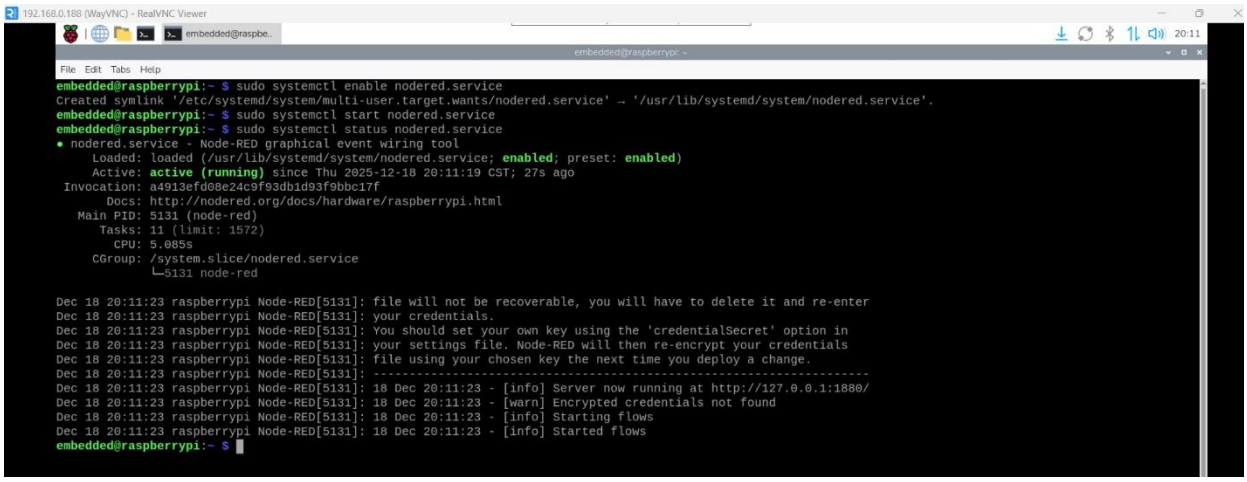




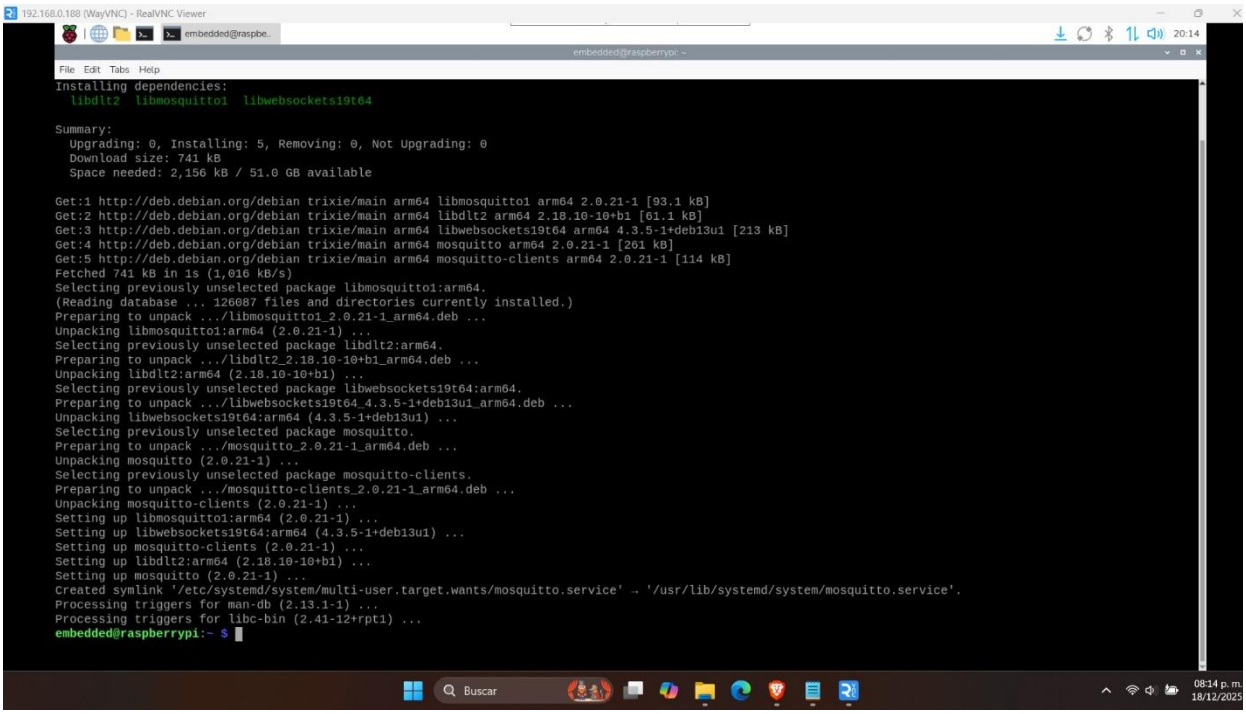
5. Habilitación y Verificación del Servicio (Node-RED)

Una vez finalizada la descarga de los paquetes, fue necesario configurar el sistema operativo para que trate a Node-RED como un servicio crítico de fondo (*Daemon*), asegurando su disponibilidad continua sin necesidad de iniciar sesión manualmente.

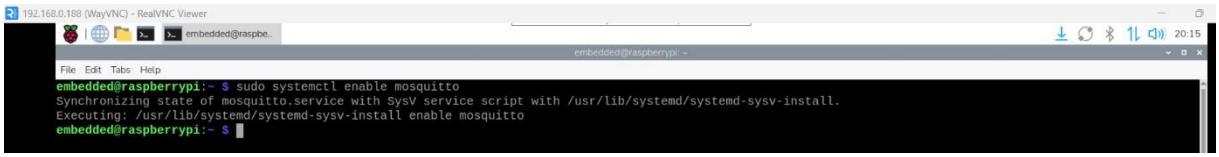
- **Persistencia del Servicio:** Se ejecutó el comando *sudo systemctl enable nodered.service*. Esta instrucción vincula la aplicación a la secuencia de arranque del sistema operativo, garantizando que, ante un corte de energía o reinicio inesperado, el servidor se restablezca automáticamente sin intervención humana.
- **Inicialización:** Se procedió a iniciar el servicio manualmente para la sesión actual mediante el comando *sudo systemctl start nodered.service*.
- **Verificación:** Con el comando *sudo systemctl status nodered.service*, se comprueba que está corriendo y escuchando.



6. **Implementación del Broker de Mensajería (Mosquitto MQTT)**
   Para gestionar la comunicación asíncrona entre los sensores (ESP32) y la interfaz de usuario, se implementó el broker **Eclipse Mosquitto**. Este software actúa como el intermediario central de la arquitectura IoT, recibiendo las publicaciones de los sensores y distribuyéndolas a los clientes suscritos.
   - **Instalación de Paquetes:** Se instaló el servidor y las herramientas de cliente mediante el gestor de paquetes del sistema: *sudo apt install -y mosquitto mosquitto-clients*.



- **Habilitación Temporal:** Se procedió a iniciar el servicio manualmente para la sesión actual mediante el comando *sudo systemctl enable mosquitto*.

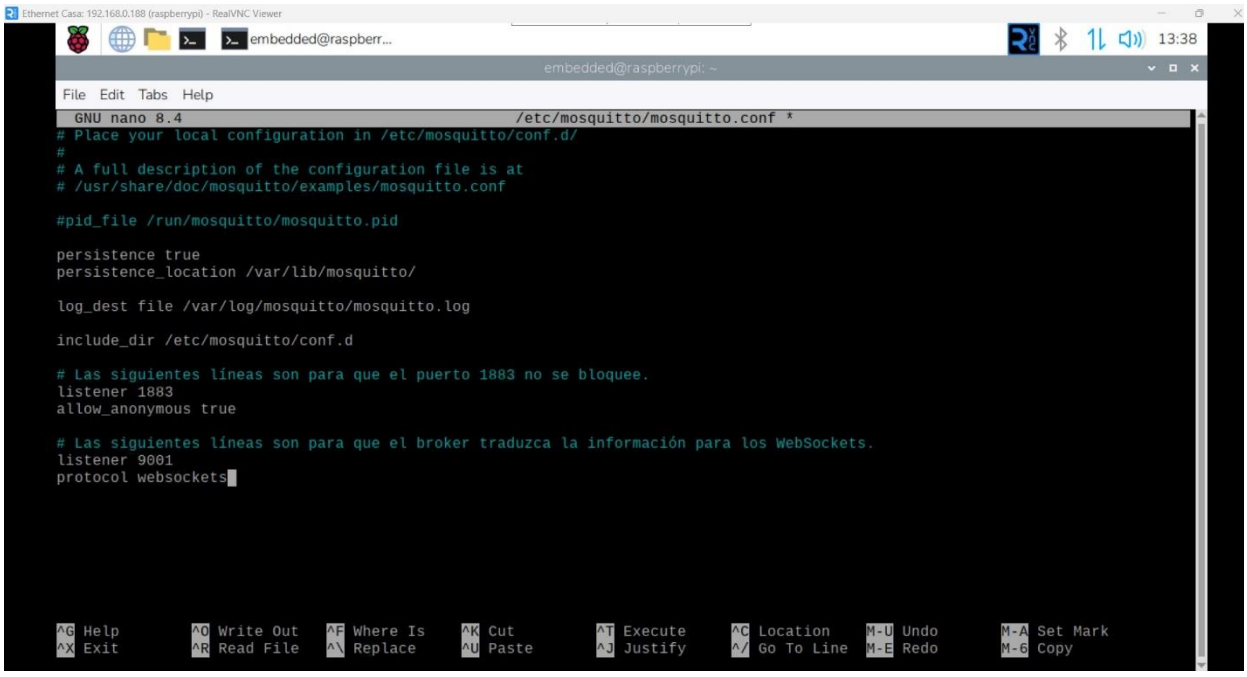


Nota 3: *Es temporal por que vamos a modificar el archivo de mosquitto para que escuche los puertos 1880 y 9001.*

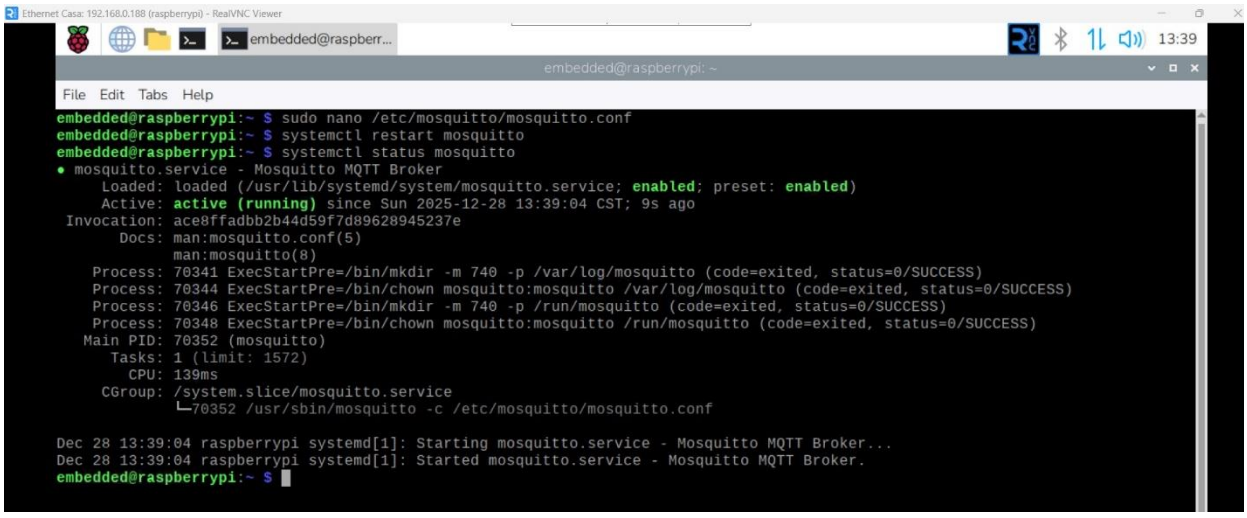
7. Configuración y Habilitación del Broker (Mosquitto MQTT)

Por defecto, la instalación de Mosquitto está bloqueada a otros puertos que por temas de seguridad solo escucha en localhost. Para permitir que la ESP32 y el Dashboard se conecten fue necesario modificar el archivo de *mosquitto.conf*.

- **Edición del Archivo de Configuración:** Utilizando el editor de texto nano, se modificó el archivo */etc/mosquitto/mosquitto.conf*.
    - **Habilitación de Puertos:** Se definieron dos "listeners" simultáneos:
        - **listener 1883:** Habilita el protocolo MQTT nativo sobre TCP, optimizado para la comunicación ligera con el microcontrolador ESP32.
        - **listener 9001 protocol WebSockets:** Habilita el protocolo WebSockets, creando un puente para que las aplicaciones web (Frontend) puedan consumir datos MQTT directamente sin necesidad de servidores intermedios.
    - **Políticas de Acceso:** Se configuró la directiva allow_anonymous true para permitir conexiones ágiles dentro de la red local aislada del prototipo.



- **Reinicio y Persistencia:** Tras guardar la configuración, se reinició el servicio *sudo systemctl restart mosquitto* para aplicar los cambios en la tabla de enrutamiento del servidor.



8. **Configuración de Red (Prioridad de Hotspot)**

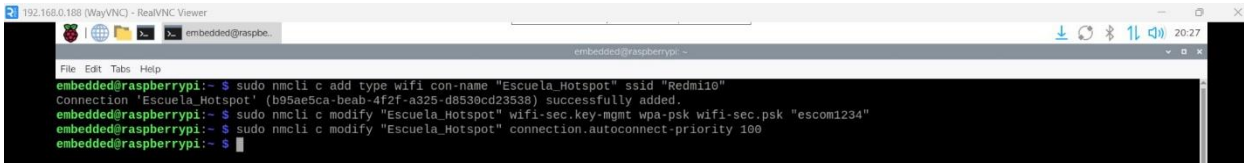
   Para garantizar la operatividad del sistema en cualquier entorno físico (laboratorios o exteriores) sin depender de infraestructura institucional, se configuró la Raspberry Pi como Punto de Acceso Inalámbrico *(Hotspot)*.

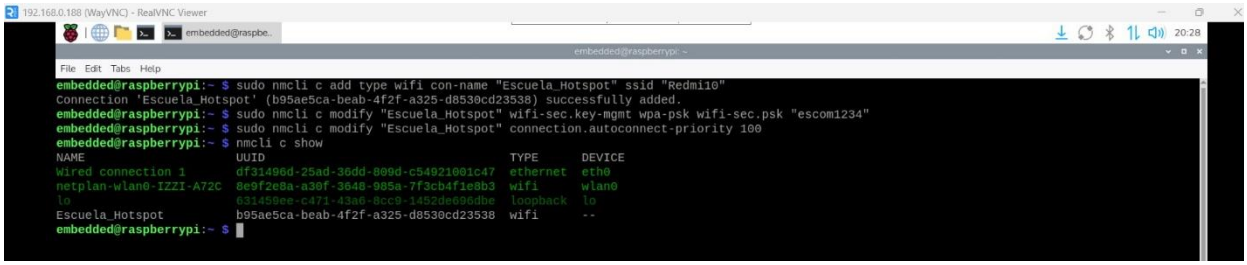
   - **Creación del Punto de Acceso:** Utilizando el gestor de redes *Network Manager* del sistema operativo, se creó una nueva conexión en modo "Hotspot". Esto convierte a la tarjeta de red de la Raspberry Pi en un emisor de señal Wi-Fi, actuando como el *Gateway* predeterminado para los sensores. Usando el comando *sudo nmcli c add type wifi con-name "Escuela_Hotspot ssid Redmi10"*
   - **Seguridad y SSID:** Se definió un nombre de red (SSID) específico para el proyecto y una contraseña WPA2, asegurando que solo nuestros microcontroladores (ESP32) y dispositivos de monitoreo autorizados puedan acceder a la red de control. Usando el comando *sudo nmcli c modify "Escuela_Hotspot" wifi-sec.key-mgmt wpa-psk wifi-sec.psk "escom1234"*

   - **Política de Prioridad de Conexión:** Un aspecto crítico fue modificar la **Prioridad de Conexión** en el gestor de redes.
     - **Problema:** Por defecto, la Raspberry PI intenta conectarse a redes por medio del puerto Ethernet y

en caso de no haber lo hará a través de Wi-Fi, pero normalmente se tratará de conectar a la primera red Wi-Fi que brinde mejor velocidad (ejemplo: 5 GHz). Y nuestro ESP32 necesita de 2.4 GHz para funcionar, por lo que nuestro Hotspot también tiene que estar a la misma frecuencia.

- **Solución:** Se elevó la prioridad del perfil "Hotspot" sobre las conexiones cliente estándar. Esto asegura que, tras un reinicio, el sistema siempre intente levantar su propia red primero, garantizando que el servidor MQTT esté disponible inmediatamente para los sensores. Usando el comando *sudo nmcli c modify "Escuela_Hospot" connection.autoconnect-priority 100* y finalmente comprobamos con el comando *nmcli c show* si se guardo correctamente nuestro Hotspot.





9. Integración del Sistema y Despliegue

Una vez configurada la infraestructura del servidor (Raspberry Pi), se procedió a la integración final de los componentes de hardware (Sensores y ESP32) y la validación del flujo de datos completo.

9.1. Lógica de Firmware (ESP32): Se programó el microcontrolador ESP32 utilizando el *framework* de Arduino. Se implementó una lógica de **"Envío por**

Excepción" para optimizar el ancho de banda de la red IoT:

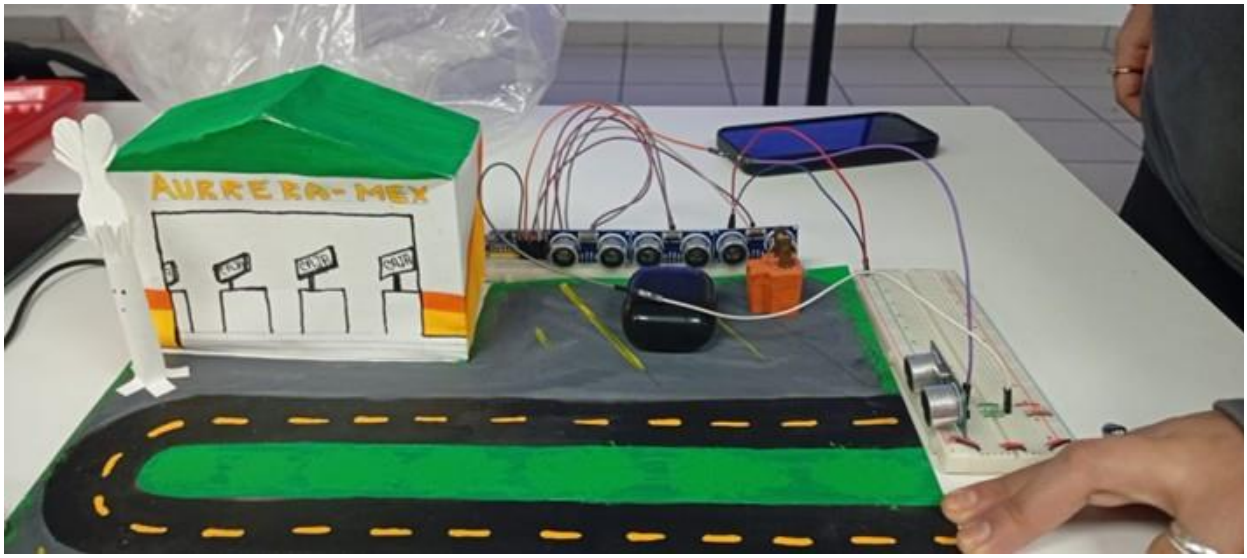
- El sistema realiza un sondeo de los 4 sensores ultrasónicos cada 200ms.
- Si la distancia medida es menor a 10 cm, se determina el estado como "OCUPADO".
- El mensaje MQTT solo se publica al *Broker* si el estado actual es diferente al estado anterior. Esto evita saturar la red con mensajes redundantes ("LIBRE", "LIBRE", "LIBRE"...) y reduce el consumo energético del procesador.

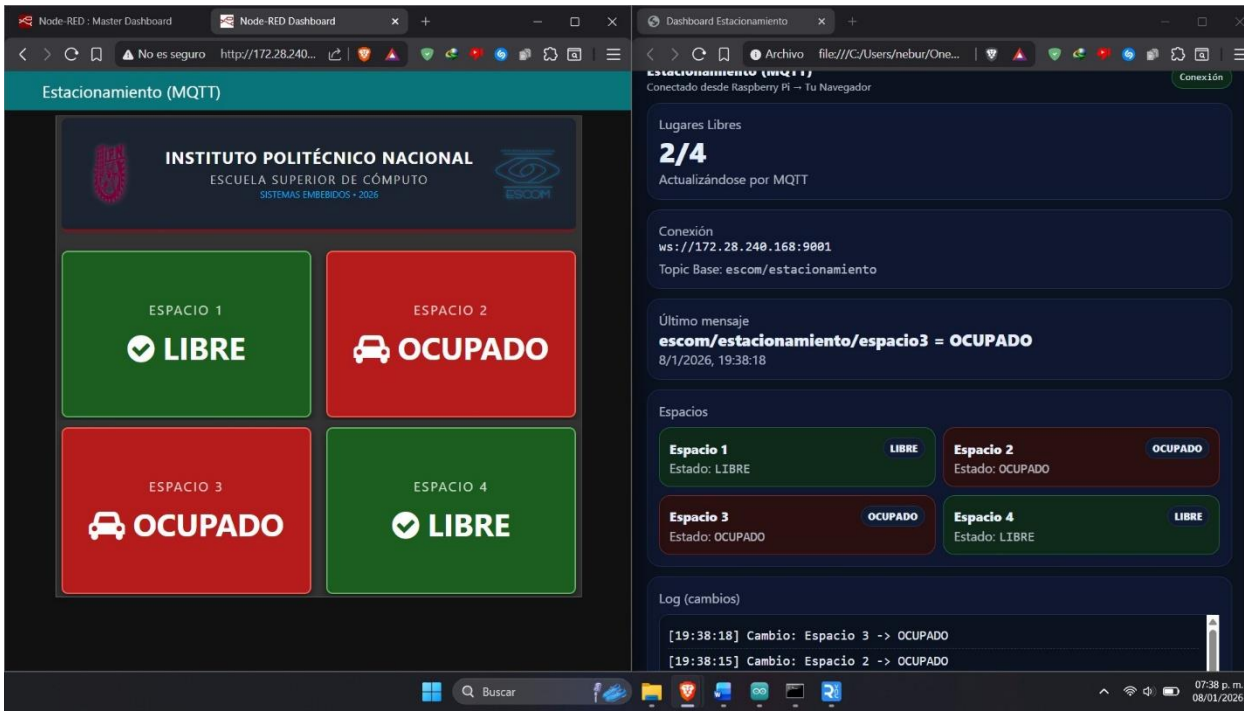
9.2. Despliegue del Dashboard: En Node-RED, se diseñó un dashboard que actúa como suscriptor de los tópicos MQTT.

- **Procesamiento Visual:** Se implementaron funciones JavaScript intermedias para interpretar los mensajes de texto ("LIBRE"/"OCUPADO") y traducirlos en atributos visuales dinámicos (Colores Verde/Rojo e íconos).
- **Offline:** Para garantizar que los logotipos institucionales (IPN/ESCOM) se visualicen incluso sin conexión a Internet, se codificaron las imágenes en formato **Base64** directamente dentro del código del Dashboard, asegurando la portabilidad del sistema.

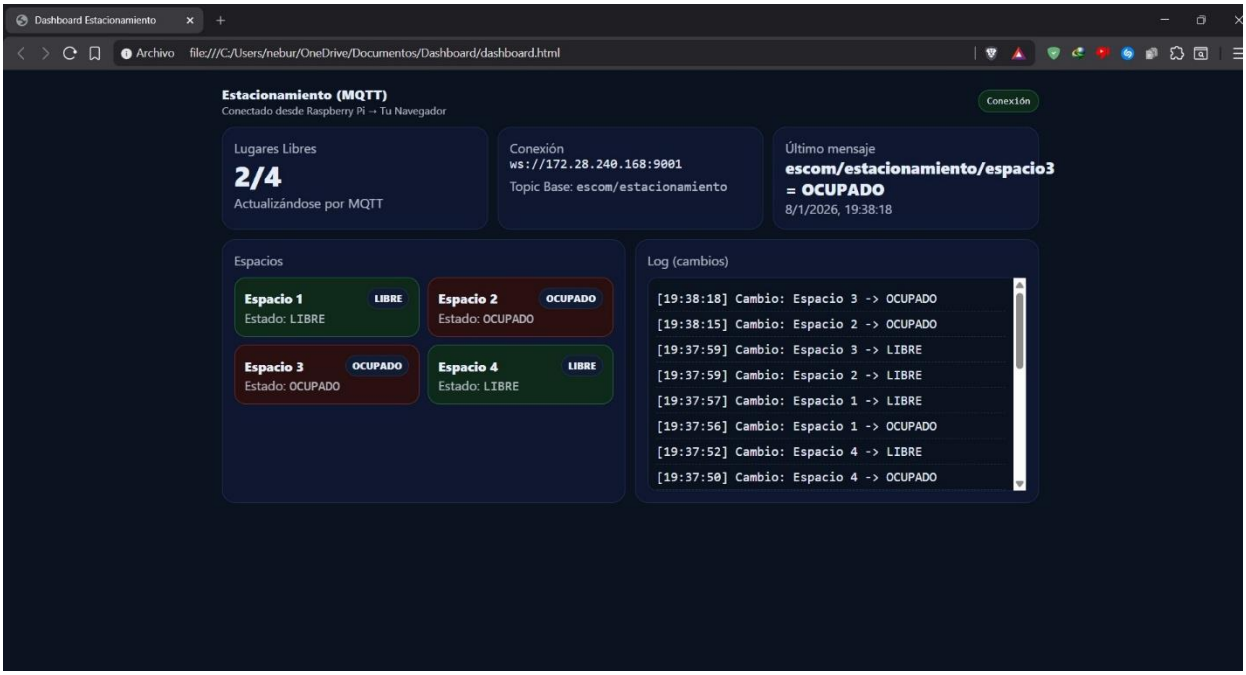
9.3. Funcionamiento: Se realizaron pruebas físicas colocando vehículos a escala frente a los sensores.

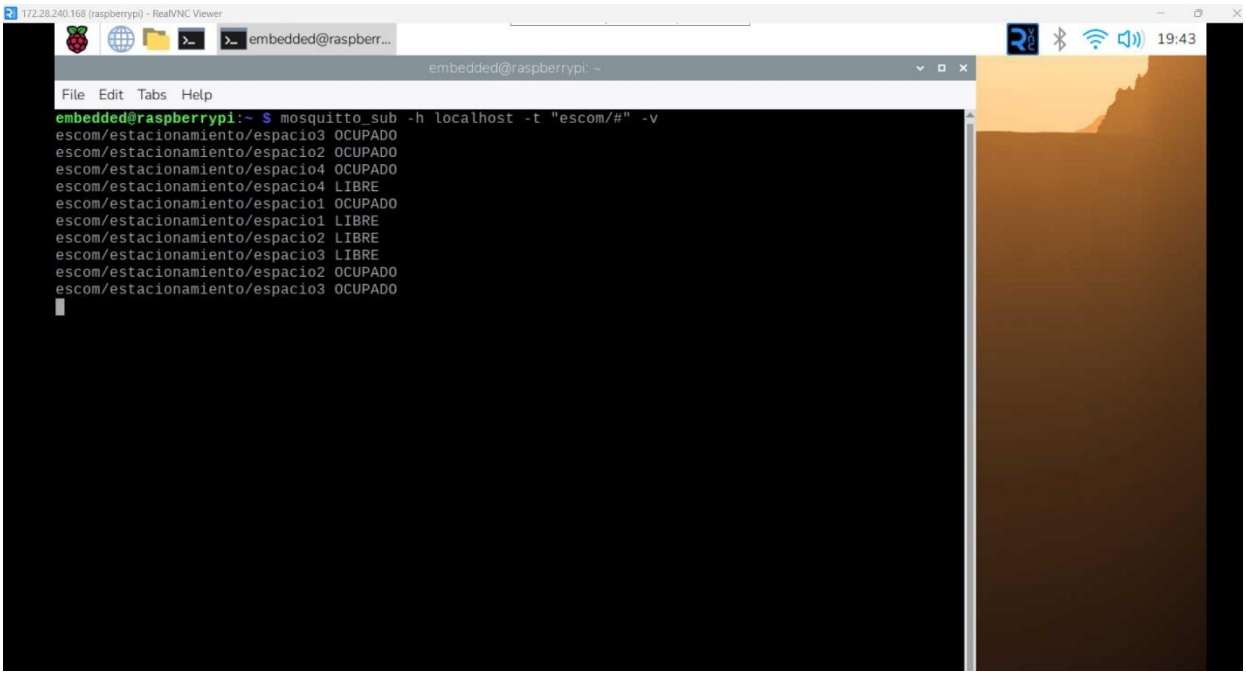
- **Resultado:** Se observó una latencia mínima entre la detección física y la actualización en la pantalla del Dashboard.
- **Validación de Red:** El sistema operó exitosamente conectado exclusivamente al *Hotspot* de la Raspberry Pi.


AURRERA-MEX

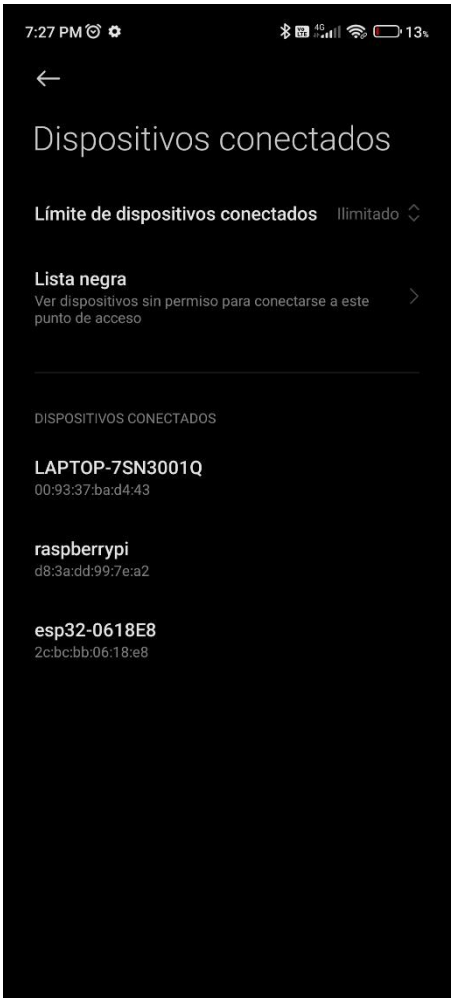

Node-RED : Master Dashboard
Node-RED Dashboard
No es seguro
http://172.28.240...
Estacionamiento (MQTT)
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SISTEMAS EMBEBIDOS • 2026
ESPACIO 1
LIBRE
ESPACIO 2
OCUPADO
ESPACIO 3
OCUPADO
ESPACIO 4
LIBRE
Dashboard Estacionamiento
Archivo
file:///C:/Users/nebur/One...
Conectado desde Raspberry Pi → Tu Navegador
Conexión
Lugares Libres
2/4
Actualizándose por MQTT
Conexión
ws://172.28.240.168:9001
Topic Base: escom/estacionamiento
Último mensaje
escom/estacionamiento/espacio3 = OCUPADO
8/1/2026, 19:38:18
Espacios
Espacio 1
LIBRE
Estado: LIBRE
Espacio 2
OCUPADO
Estado: OCUPADO
Espacio 3
OCUPADO
Estado: OCUPADO
Espacio 4
LIBRE
Estado: LIBRE
Log (cambios)
[19:38:18] Cambio: Espacio 3 -> OCUPADO
[19:38:15] Cambio: Espacio 2 -> OCUPADO
Buscar
07:38 p. m.
08/01/2026


Node-RED : Master Dashboard
Node-RED Dashboard
No es seguro
http://172.28.240.168:1880/ui/#!/0?socketid=WLgYg3B6L-OY4TaoAAAF
Estacionamiento (MQTT)
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SISTEMAS EMBEBIDOS • 2026
ESPACIO 1
LIBRE
ESPACIO 2
OCUPADO
ESPACIO 3
OCUPADO
ESPACIO 4
LIBRE


Dashboard Estacionamiento
Archivo
file:///C:/Users/nebur/OneDrive/Documentos/Dashboard/dashboard.html
Estacionamiento (MQTT)
Conectado desde Raspberry Pi → Tu Navegador
Conexión
Lugares Libres
2/4
Actualizándose por MQTT
Conexión
ws://172.28.240.168:9001
Topic Base: escom/estacionamiento
Último mensaje
escom/estacionamiento/espacio3 = OCUPADO
8/1/2026, 19:38:18
Espacios
Espacio 1
LIBRE
Estado: LIBRE
Espacio 2
OCUPADO
Estado: OCUPADO
Espacio 3
OCUPADO
Estado: OCUPADO
Espacio 4
LIBRE
Estado: LIBRE
Log (cambios)
[19:38:18] Cambio: Espacio 3 -> OCUPADO
[19:38:15] Cambio: Espacio 2 -> OCUPADO
[19:37:59] Cambio: Espacio 3 -> LIBRE
[19:37:59] Cambio: Espacio 2 -> LIBRE
[19:37:57] Cambio: Espacio 1 -> LIBRE
[19:37:56] Cambio: Espacio 1 -> OCUPADO
[19:37:52] Cambio: Espacio 4 -> LIBRE
[19:37:50] Cambio: Espacio 4 -> OCUPADO


172.28.240.168 (raspberrypi) - RealVNC Viewer
embedded@raspberr...
19:43
embedded@raspberrypi: ~
File Edit Tabs Help
embedded@raspberrypi:~ $ mosquitto_sub -h localhost -t "escom/#" -v
escom/estacionamiento/espacio3 OCUPADO
escom/estacionamiento/espacio2 OCUPADO
escom/estacionamiento/espacio4 OCUPADO
escom/estacionamiento/espacio4 LIBRE
escom/estacionamiento/espacio1 OCUPADO
escom/estacionamiento/espacio1 LIBRE
escom/estacionamiento/espacio2 LIBRE
escom/estacionamiento/espacio3 LIBRE
escom/estacionamiento/espacio2 OCUPADO
escom/estacionamiento/espacio3 OCUPADO


7:27 PM
13%
Establecer punto de acceso móvil
Nombre de la red
Redmi10
Contraseña
escom1234
Seguridad
WPA2-Personal
Seleccionar banda AP
2.4 GHz
Ocultar SSID
Desactivado


7:27 PM
13%
Dispositivos conectados
Límite de dispositivos conectados
Ilimitado
Lista negra
Ver dispositivos sin permiso para conectarse a este punto de acceso
DISPOSITIVOS CONECTADOS
LAPTOP-7SN3001Q
00:93:37:ba:d4:43
raspberrypi
d8:3a:dd:99:7e:a2
esp32-0618E8
2c:bc:bb:06:18:e8