



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?

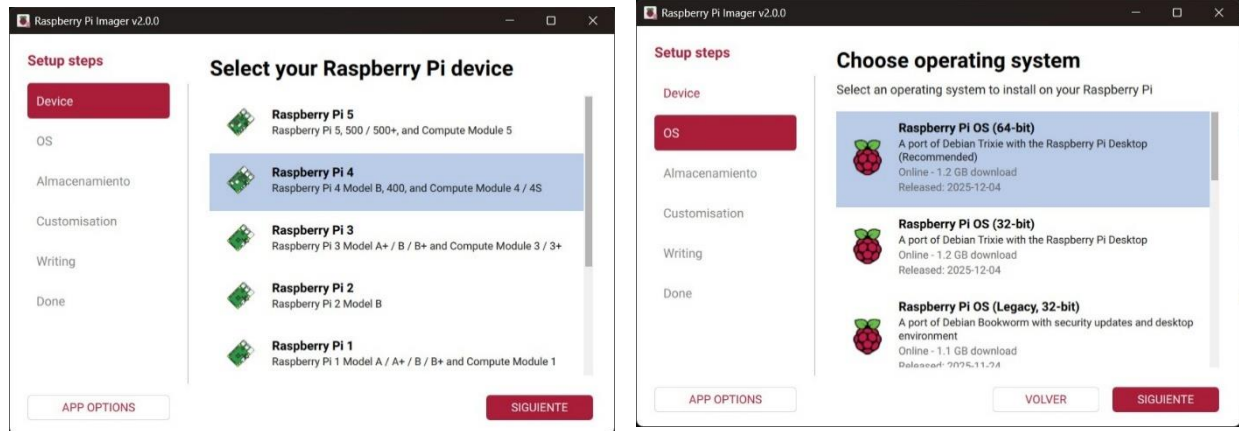
- Arduino IDE
- Node-RED
- Eclipse Mosquitto
- Raspberry Pi Imager
- RealVNC Viewer & VNC Server

## Desarrollo de lo que se hizo

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

#### 1.1. Selección del Sistema Operativo

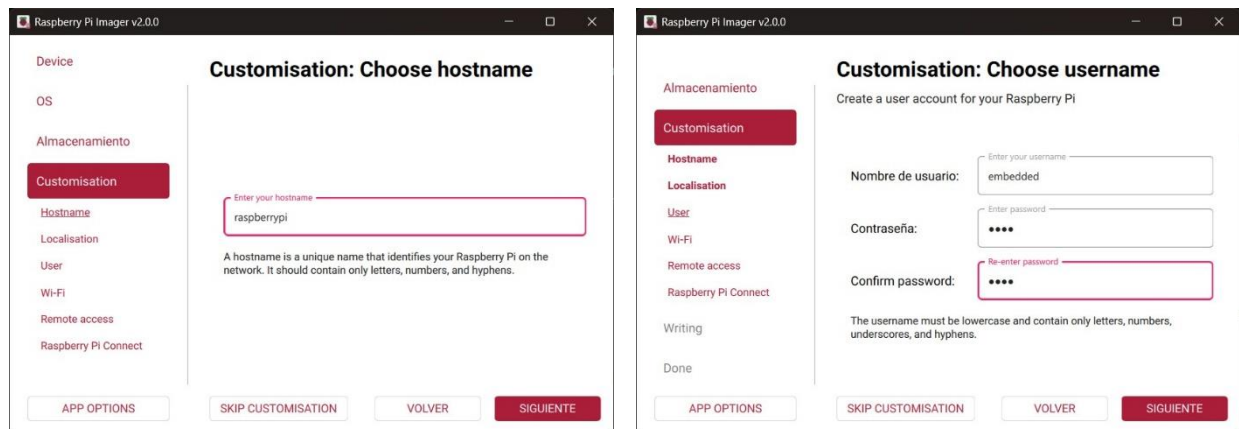
Se seleccionó la arquitectura de hardware **Raspberry Pi 4** y se instaló el sistema operativo **Raspberry Pi OS (64-bit)**. Se optó por la versión de 64 bits para maximizar la compatibilidad y el rendimiento con los servicios de middleware (Mosquitto y Node-RED) a implementar.



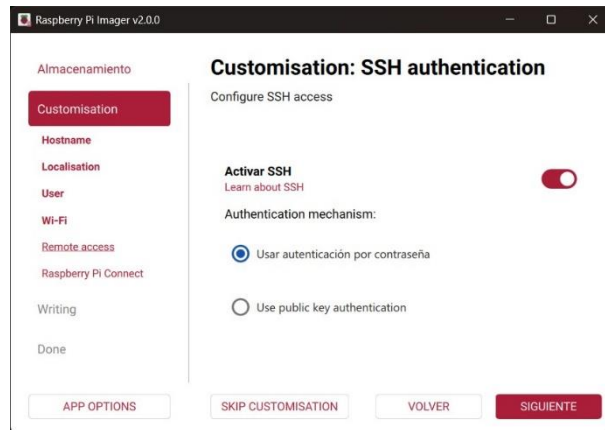
## 1.2. Configuración Avanzada

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

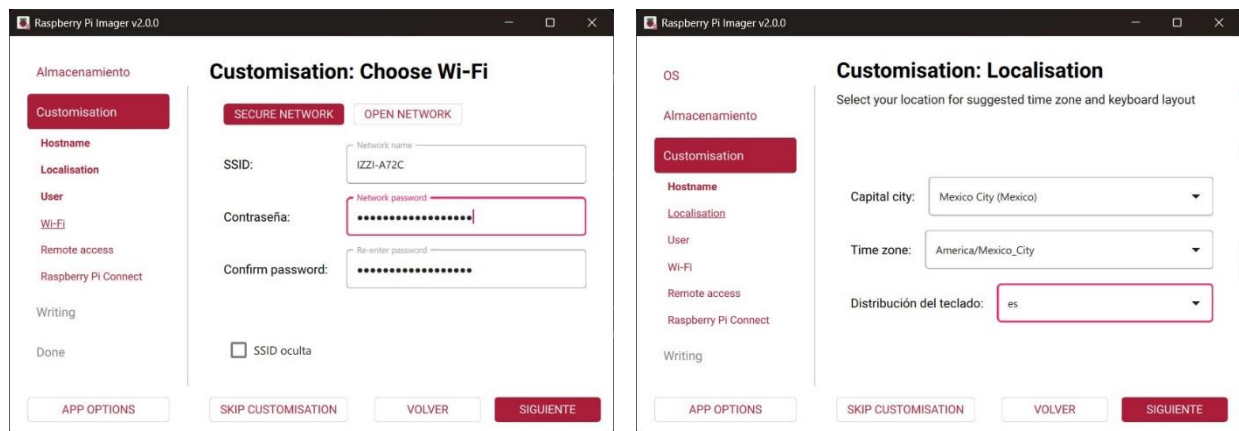
- **Identificación y Seguridad:** Se asignó el *hostname* raspberrypi para su fácil localización en la red local. También, se creó el *usuario* dedicado llamado embedded con privilegios de administrador.



- **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.

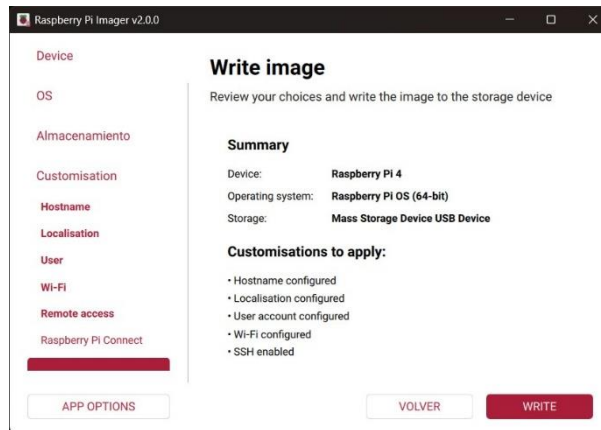


- **Conectividad:** Se preconfiguraron las credenciales de la red Wi-Fi local y la zona horaria (Mexico City) para asegurar la sincronización automática del sistema.



### 1.3. Resultados

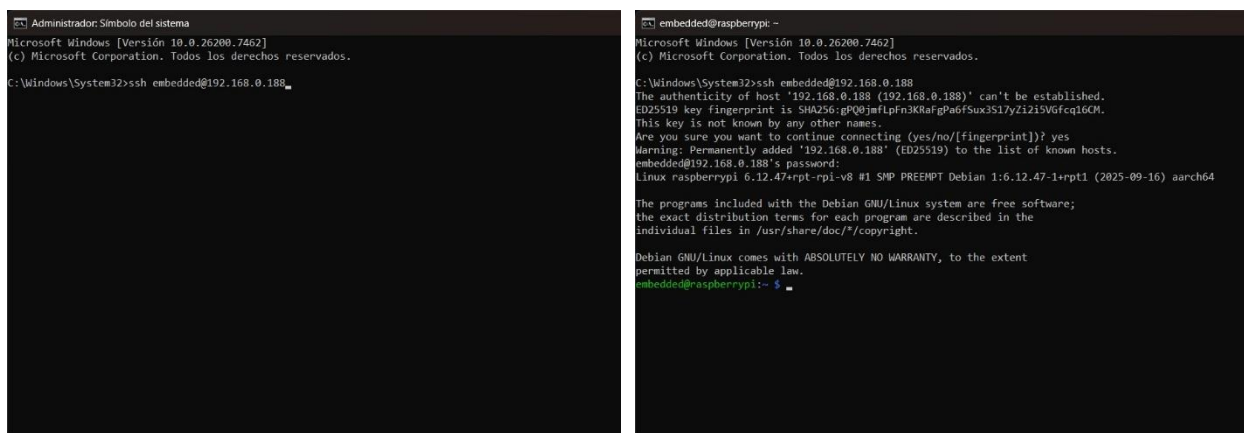
Se nos presenta al final de la configuración en forma de resumen todos y cada uno de los puntos mencionadas previamente, esto con la finalidad de que estemos de acuerdo con las características que se usaran.



## 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)

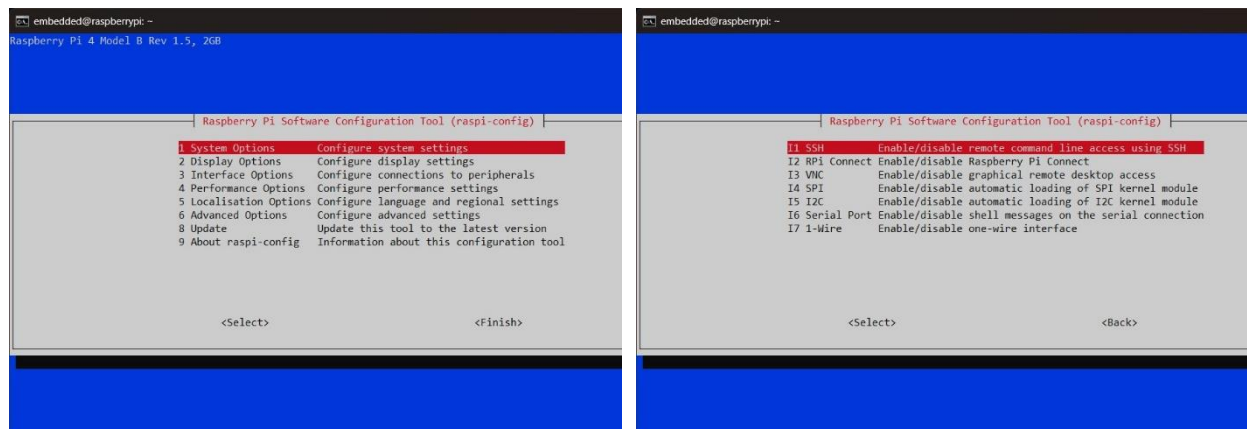
Dado que el proyecto requiere visualizar el Dashboard localmente y gestionar herramientas gráficas, se procedió a habilitar el servidor VNC (*Virtual Network Computing*). Esto permite visualizar el escritorio completo de la Raspberry PI en la computadora de desarrollo, eliminando definitivamente la necesidad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

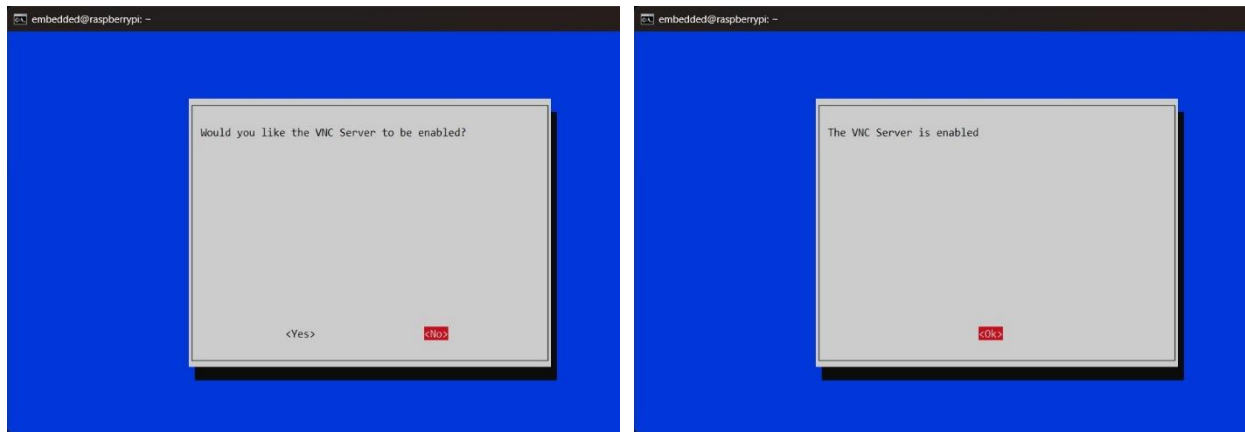
Para realizar esta configuración sin interfaz gráfica previa, se utilizó la herramienta de administración por consola:

- **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.

```
embedded@raspberrypi: ~  
embedded@raspberrypi:~$ sudo raspi-config
```

- **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, confirmandonos 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.

```
embedded@raspberrypi: ~  
embedded@raspberrypi:~$ sudo raspi-config  
Created symlink '/etc/systemd/system/multi-user.target.wants/wayvnc.service' → '/usr/lib/systemd/system/wayvnc.service'.  
embedded@raspberrypi:~$
```

*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.

```
192.168.0.188 (WayVNC) - RealVNC Viewer
embedded@raspberrypi: ~
File Edit Tabs Help
embedded@raspberrypi:~$ sudo apt update
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
Hit:4 http://archive.raspberrypi.com/debian trixie InRelease
All packages are up to date.
embedded@raspberrypi:~$ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

```
192.168.0.188 (WayVNC) - RealVNC Viewer
embedded@raspberrypi: ~
File Edit Tabs Help
embedded@raspberrypi:~$ sudo apt update
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
Hit:4 http://archive.raspberrypi.com/debian trixie InRelease
All packages are up to date.
embedded@raspberrypi:~$ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

Node-RED update script for : embedded@raspberrypi

This script checks the version of node.js installed is 16 or greater. It will try to
install node 20 if none is found. It can optionally install node 18 or 20 LTS for you.

If necessary it will then remove the old core of Node-RED, before then installing the latest
version. You can also optionally specify the version required.

It also tries to run 'npm rebuild' to refresh any extra nodes you have installed
that may have a native binary component. While this normally works ok, you need
to check that it succeeds for your combination of installed nodes.

To do all this it runs commands as root - please satisfy yourself that this will
not damage your Pi, or otherwise compromise your configuration.
If in doubt please backup your SD card first.

See the optional parameters by re-running this command with --help

Are you really sure you want to do this ? [y/N] ?
```

```
192.168.0.188 (WayVNC) - RealVNC Viewer
embedded@raspberrypi: ~
File Edit Tabs Help
embedded@raspberrypi:~$ sudo apt update
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
Hit:4 http://archive.raspberrypi.com/debian trixie InRelease
All packages are up to date.
embedded@raspberrypi:~$ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

Node-RED update script for : embedded@raspberrypi

This script checks the version of node.js installed is 16 or greater. It will try to
install node 20 if none is found. It can optionally install node 18 or 20 LTS for you.

If necessary it will then remove the old core of Node-RED, before then installing the latest
version. You can also optionally specify the version required.

It also tries to run 'npm rebuild' to refresh any extra nodes you have installed
that may have a native binary component. While this normally works ok, you need
to check that it succeeds for your combination of installed nodes.

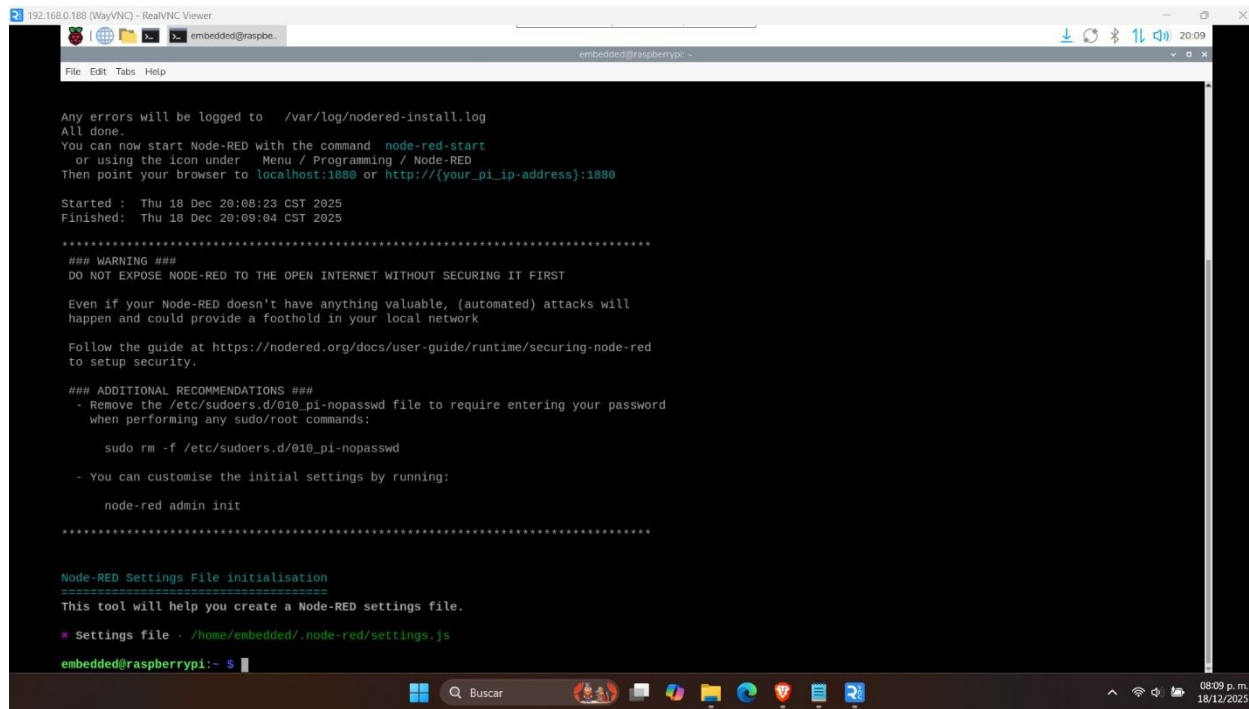
To do all this it runs commands as root - please satisfy yourself that this will
not damage your Pi, or otherwise compromise your configuration.
If in doubt please backup your SD card first.

See the optional parameters by re-running this command with --help

Are you really sure you want to do this ? [y/N] ? y

Would you like to install the Pi-specific nodes ? [y/N] ?
```





```
Any errors will be logged to /var/log/nodered-install.log
All done.
You can now start Node-RED with the command node-red-start
or using the icon under Menu / Programming / Node-RED
Then point your browser to localhost:1880 or http://{your_pi_ip-address}:1880

Started : Thu 18 Dec 20:08:23 CST 2025
Finished: Thu 18 Dec 20:09:04 CST 2025

*****
### WARNING ###
DO NOT EXPOSE NODE-RED TO THE OPEN INTERNET WITHOUT SECURING IT FIRST

Even if your Node-RED doesn't have anything valuable, (automated) attacks will
happen and could provide a foothold in your local network

Follow the guide at https://nodered.org/docs/user-guide/runtime/securing-node-red
to setup security.

### ADDITIONAL RECOMMENDATIONS ###
- Remove the /etc/sudoers.d/010_pi-nopasswd file to require entering your password
  when performing any sudo/root commands:

  sudo rm -f /etc/sudoers.d/010_pi-nopasswd

- You can customise the initial settings by running:

  node-red admin init

*****

Node-RED Settings File Initialisation
*****
This tool will help you create a Node-RED settings file.

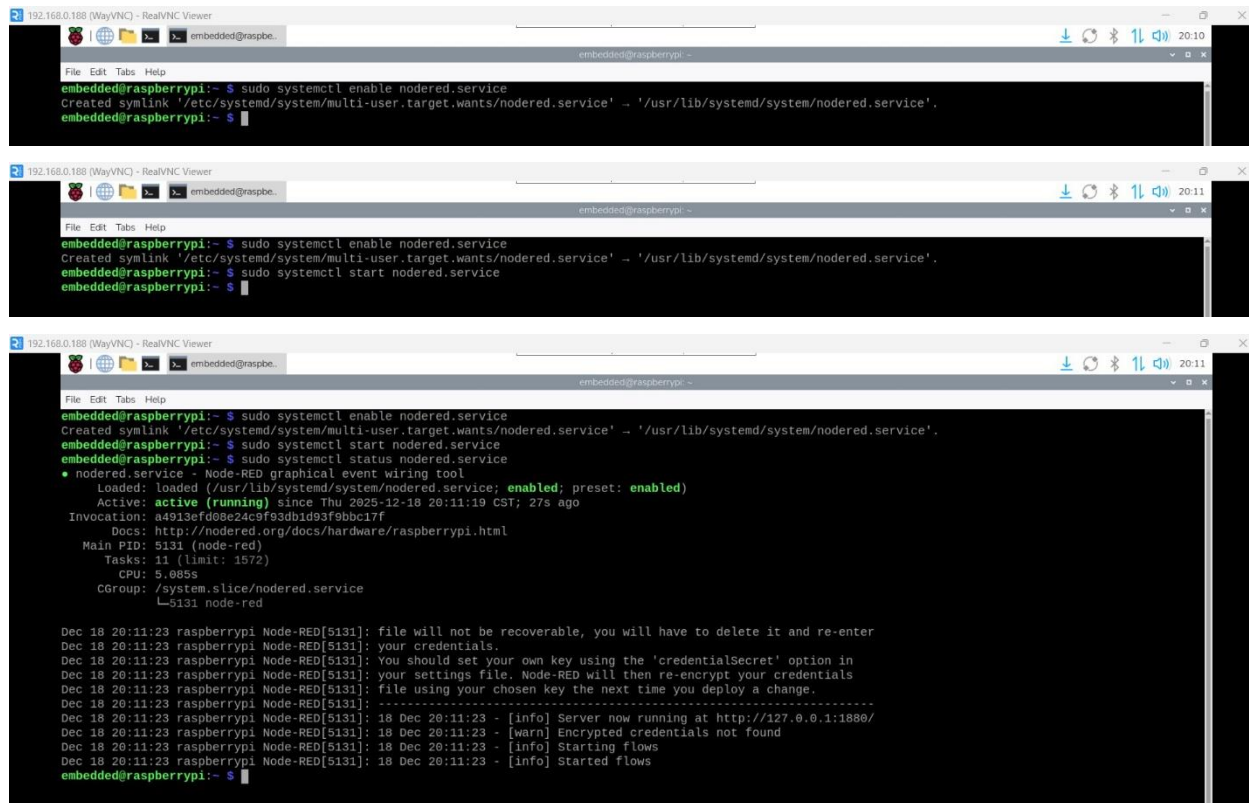
* Settings file : /home/embedded/.node-red/settings.js

embedded@raspberrypi:~$
```

## 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.



```
embedded@raspberrypi:~$ sudo systemctl enable nodered.service
Created symlink '/etc/systemd/system/multi-user.target.wants/nodered.service' -> '/usr/lib/systemd/system/nodered.service'.
embedded@raspberrypi:~$

embedded@raspberrypi:~$ sudo systemctl enable nodered.service
Created symlink '/etc/systemd/system/multi-user.target.wants/nodered.service' -> '/usr/lib/systemd/system/nodered.service'.
embedded@raspberrypi:~$ sudo systemctl start nodered.service
embedded@raspberrypi:~$

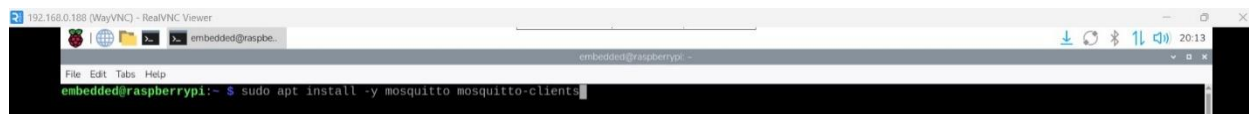
embedded@raspberrypi:~$ sudo systemctl status nodered.service
● nodered.service - Node-RED graphical event wiring tool
   Loaded: loaded (/usr/lib/systemd/system/nodered.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-12-18 20:11:19 CST; 27s ago
     Invocation: a4913efd08e24c9f93db1d93f9bbc17f
       Docs: http://nodered.org/docs/hardware/raspberrypi.html
    Main PID: 5131 (node-red)
      Tasks: 11 (limit: 1572)
         CPU: 5.085s
        CGroup: /system.slice/nodered.service
                └─5131 node-red

Dec 18 20:11:23 raspberrypi Node-RED[5131]: file will not be recoverable, you will have to delete it and re-enter
Dec 18 20:11:23 raspberrypi Node-RED[5131]: your credentials.
Dec 18 20:11:23 raspberrypi Node-RED[5131]: You should set your own key using the 'credentialSecret' option in
Dec 18 20:11:23 raspberrypi Node-RED[5131]: your settings file. Node-RED will then re-encrypt your credentials
Dec 18 20:11:23 raspberrypi Node-RED[5131]: file using your chosen key the next time you deploy a change.
Dec 18 20:11:23 raspberrypi Node-RED[5131]: -----
Dec 18 20:11:23 raspberrypi Node-RED[5131]: 18 Dec 20:11:23 - [info] Server now running at http://127.0.0.1:1880/
Dec 18 20:11:23 raspberrypi Node-RED[5131]: 18 Dec 20:11:23 - [warn] Encrypted credentials not found
Dec 18 20:11:23 raspberrypi Node-RED[5131]: 18 Dec 20:11:23 - [info] Starting flows
Dec 18 20:11:23 raspberrypi Node-RED[5131]: 18 Dec 20:11:23 - [info] Started flows
embedded@raspberrypi:~$
```

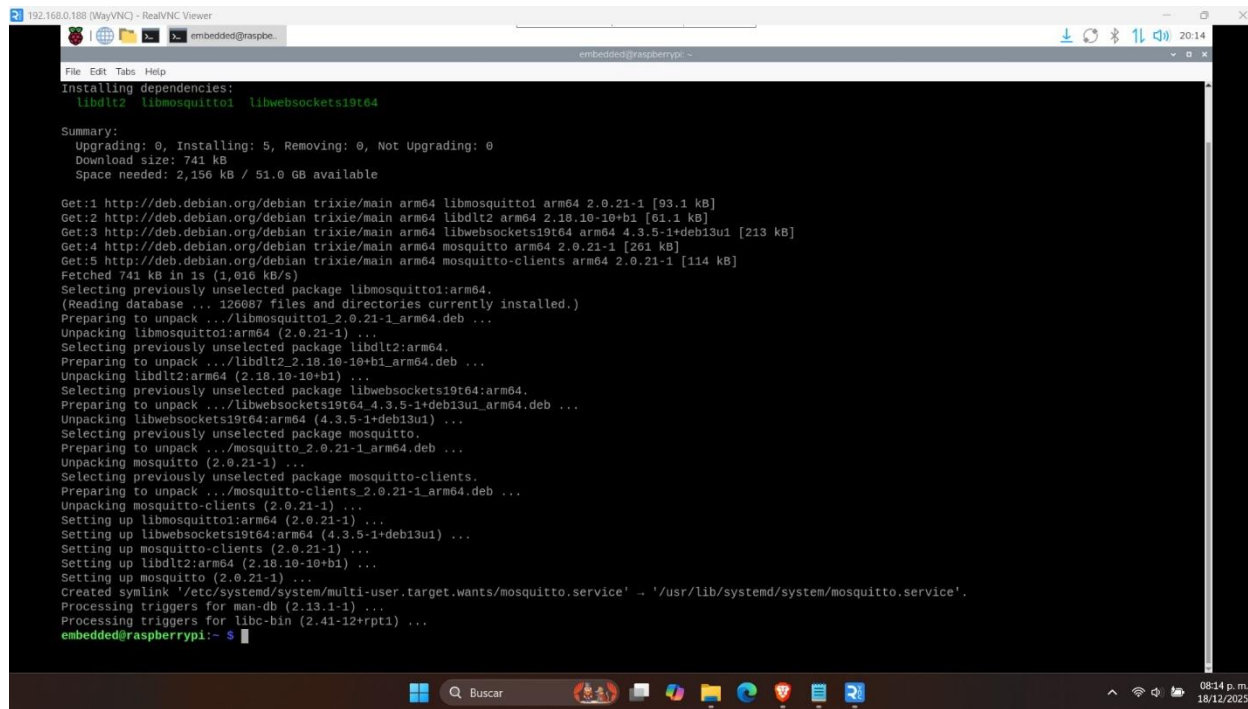
## 6. Implementación del Broker de Mensajería (Mosquitto MQTT)

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*.



```
embedded@raspberrypi:~$ sudo apt install -y mosquitto mosquitto-clients
```

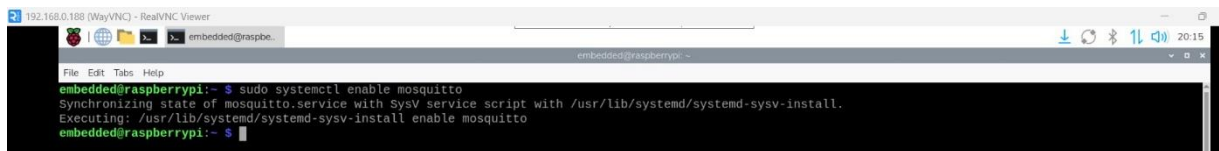


```
192.168.0.188 (WayVNC) - RealVNC Viewer
embedded@raspberrypi: ~
File Edit Tabs Help
Installing dependencies:
libldt2 libmosquitto1 libwebsockets19t64

Summary:
Upgrading: 0, Installing: 5, Removing: 0, Not Upgrading: 0
Download size: 741 kB
Space needed: 2,156 kB / 51.0 GB available

Get:1 http://deb.debian.org/debian trixie/main arm64 libmosquitto1 arm64 2.0.21-1 [93.1 kB]
Get:2 http://deb.debian.org/debian trixie/main arm64 libldt2 arm64 2.18.10-10+b1 [61.1 kB]
Get:3 http://deb.debian.org/debian trixie/main arm64 libwebsockets19t64 arm64 4.3.5-1+deb13u1 [213 kB]
Get:4 http://deb.debian.org/debian trixie/main arm64 mosquitto arm64 2.0.21-1 [261 kB]
Get:5 http://deb.debian.org/debian trixie/main arm64 mosquitto-clients arm64 2.0.21-1 [114 kB]
Fetched 741 kB in 1s (1,010 kB/s)
Selecting previously unselected package libmosquitto1:arm64.
(Reading database ... 126087 files and directories currently installed.)
Preparing to unpack .../libmosquitto1_2.0.21-1_arm64.deb ...
Unpacking libmosquitto1:arm64 (2.0.21-1) ...
Selecting previously unselected package libldt2:arm64.
Preparing to unpack .../libldt2_2.18.10-10+b1_arm64.deb ...
Unpacking libldt2:arm64 (2.18.10-10+b1) ...
Selecting previously unselected package libwebsockets19t64:arm64.
Preparing to unpack .../libwebsockets19t64_4.3.5-1+deb13u1_arm64.deb ...
Unpacking libwebsockets19t64:arm64 (4.3.5-1+deb13u1) ...
Selecting previously unselected package mosquitto.
Preparing to unpack .../mosquitto_2.0.21-1_arm64.deb ...
Unpacking mosquitto (2.0.21-1) ...
Selecting previously unselected package mosquitto-clients.
Preparing to unpack .../mosquitto-clients_2.0.21-1_arm64.deb ...
Unpacking mosquitto-clients (2.0.21-1) ...
Setting up libmosquitto1:arm64 (2.0.21-1) ...
Setting up libwebsockets19t64:arm64 (4.3.5-1+deb13u1) ...
Setting up mosquitto-clients (2.0.21-1) ...
Setting up libldt2:arm64 (2.18.10-10+b1) ...
Setting up mosquitto (2.0.21-1) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/mosquitto.service' -> '/usr/lib/systemd/system/mosquitto.service'.
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for libc-bin (2.41-12+rpt1) ...
embedded@raspberrypi:~$
```

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



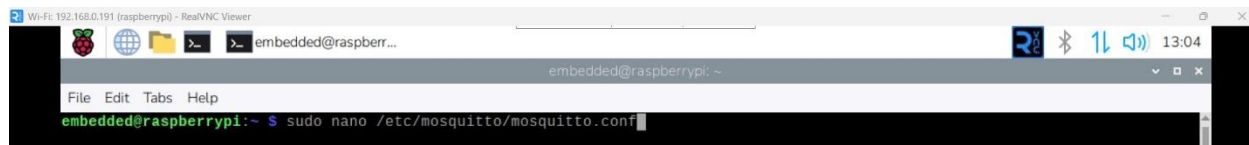
```
192.168.0.188 (WayVNC) - RealVNC Viewer
embedded@raspberrypi: ~
File Edit Tabs Help
embedded@raspberrypi:~$ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mosquitto
embedded@raspberrypi:~$
```

*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.



```
Ethernet Casa: 192.168.0.188 (raspberrypi) - RealVNC Viewer
embedded@raspberrypi: ~
GNU nano 8.4 /etc/mosquitto/mosquitto.conf *
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf

#pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

# Las siguientes líneas son para que el puerto 1883 no se bloquee.
listener 1883
allow_anonymous true

# Las siguientes líneas son para que el broker traduzca la información para los WebSockets.
listener 9001
protocol websockets
```

- **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.

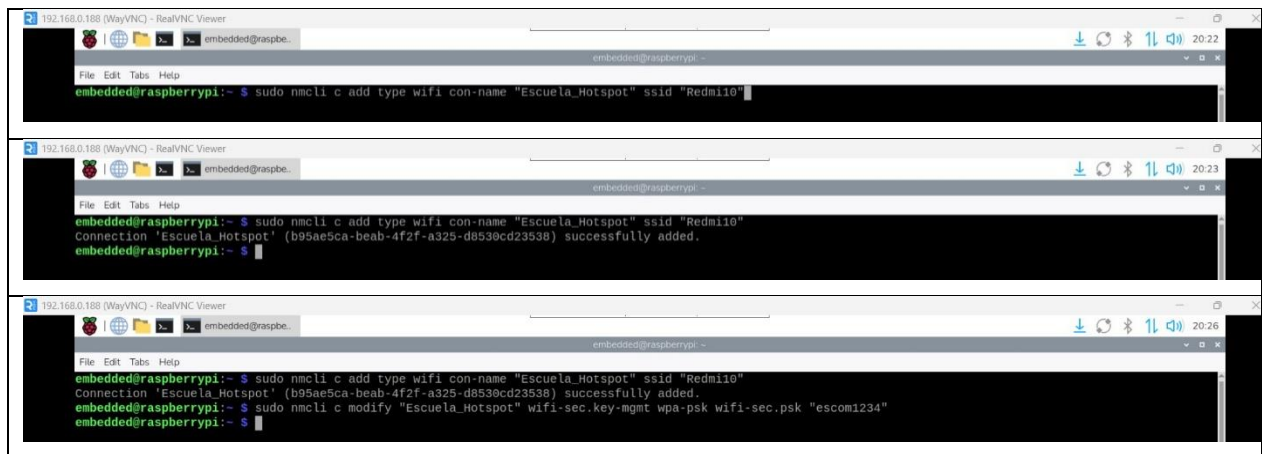
```
Ethernet Casa: 192.168.0.188 (raspberrypi) - RealVNC Viewer
embedded@raspberrypi: ~
File Edit Tabs Help
embedded@raspberrypi:~$ sudo nano /etc/mosquitto/mosquitto.conf
embedded@raspberrypi:~$ systemctl restart mosquitto
embedded@raspberrypi:~$ systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-12-28 13:39:04 CST; 9s ago
   Invocation: ace8ffadbb2b44d59f7d89628945237e
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Process: 70341 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 70344 ExecStartPre=/bin/chown mosquitto:mosquitto /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 70346 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, status=0/SUCCESS)
   Process: 70348 ExecStartPre=/bin/chown mosquitto:mosquitto /run/mosquitto (code=exited, status=0/SUCCESS)
   Main PID: 70352 (mosquitto)
     Tasks: 1 (limit: 1572)
        CPU: 139ms
   CGroup: /system.slice/mosquitto.service
           └─70352 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Dec 28 13:39:04 raspberrypi systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker...
Dec 28 13:39:04 raspberrypi systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.
embedded@raspberrypi:~$
```

## 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"`



The image displays three sequential screenshots of a terminal window running on a Raspberry Pi, connected via VNC. The terminal shows the execution of commands to create and configure a Wi-Fi hotspot named 'Escuela\_Hotspot' with the SSID 'Redmi10'. The first screenshot shows the command to add the connection. The second screenshot shows the command to modify the connection with WPA2 security and the password 'escom1234'. The third screenshot shows the final state of the terminal after the configuration is complete.

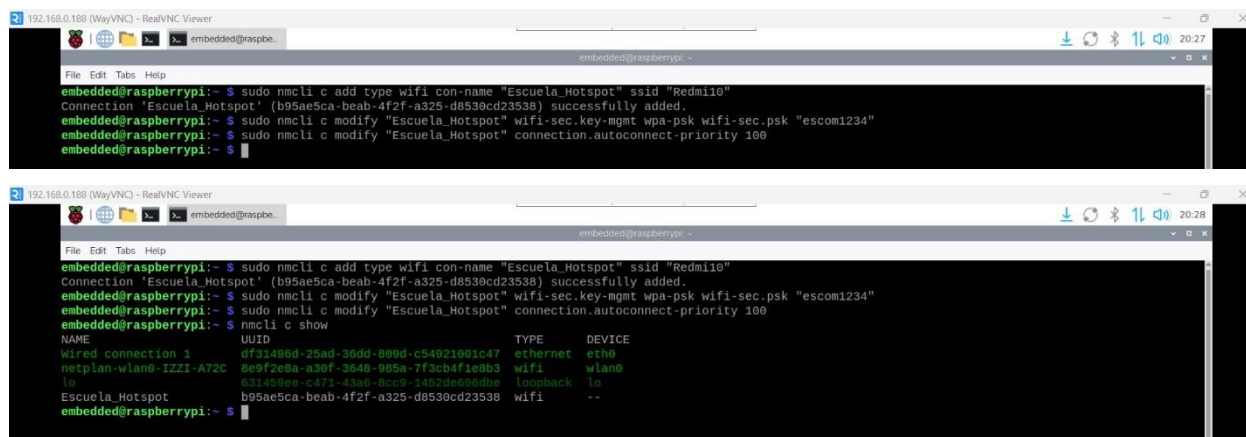
```
192.168.0.188 (WayVNC) - RealVNC Viewer
embedded@raspberrypi:~$ sudo nmcli c add type wifi con-name "Escuela_Hotspot" ssid "Redmi10"
embedded@raspberrypi:~$ sudo nmcli c modify "Escuela_Hotspot" wifi-sec.key-mgmt wpa-psk wifi-sec.psk "escom1234"
```

- **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.



The image shows two screenshots of a terminal window on a Raspberry Pi. The first screenshot shows the commands to create a new Wi-Fi connection named 'Escuela\_Hotspot' with SSID 'Redmi10', and then modify its connection priority to 100. The second screenshot shows the output of the 'nmcli c show' command, which lists the network connections and their properties.

```
embedded@raspberrypi:~$ sudo nmcli c add type wifi con-name "Escuela_Hotspot" ssid "Redmi10"
Connection 'Escuela_Hotspot' (b95ae5ca-beab-4f2f-a325-d8530cd23538) successfully added.
embedded@raspberrypi:~$ sudo nmcli c modify "Escuela_Hotspot" wifi-sec.key-mgmt wpa-psk wifi-sec.psk "escom1234"
embedded@raspberrypi:~$ sudo nmcli c modify "Escuela_Hotspot" connection.autoconnect-priority 100
embedded@raspberrypi:~$

embedded@raspberrypi:~$ sudo nmcli c add type wifi con-name "Escuela_Hotspot" ssid "Redmi10"
Connection 'Escuela_Hotspot' (b95ae5ca-beab-4f2f-a325-d8530cd23538) successfully added.
embedded@raspberrypi:~$ sudo nmcli c modify "Escuela_Hotspot" wifi-sec.key-mgmt wpa-psk wifi-sec.psk "escom1234"
embedded@raspberrypi:~$ sudo nmcli c modify "Escuela_Hotspot" connection.autoconnect-priority 100
embedded@raspberrypi:~$ nmcli c show
NAME                UUID                                  TYPE      DEVICE
wired connection 1  df31496d-25ad-36dd-809d-c54921001c47 ethernet  eth0
netplan-wlan0-IZZI-A72C  8e9f2e0a-a30f-3640-905a-7f3cb4f1e0b3 wifi      wlan0
lo                    631450ee-c471-43a0-8cc9-1452de060db6 loopback  lo
Escuela_Hotspot      b95ae5ca-beab-4f2f-a325-d8530cd23538 wifi      --
embedded@raspberrypi:~$
```

## 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.

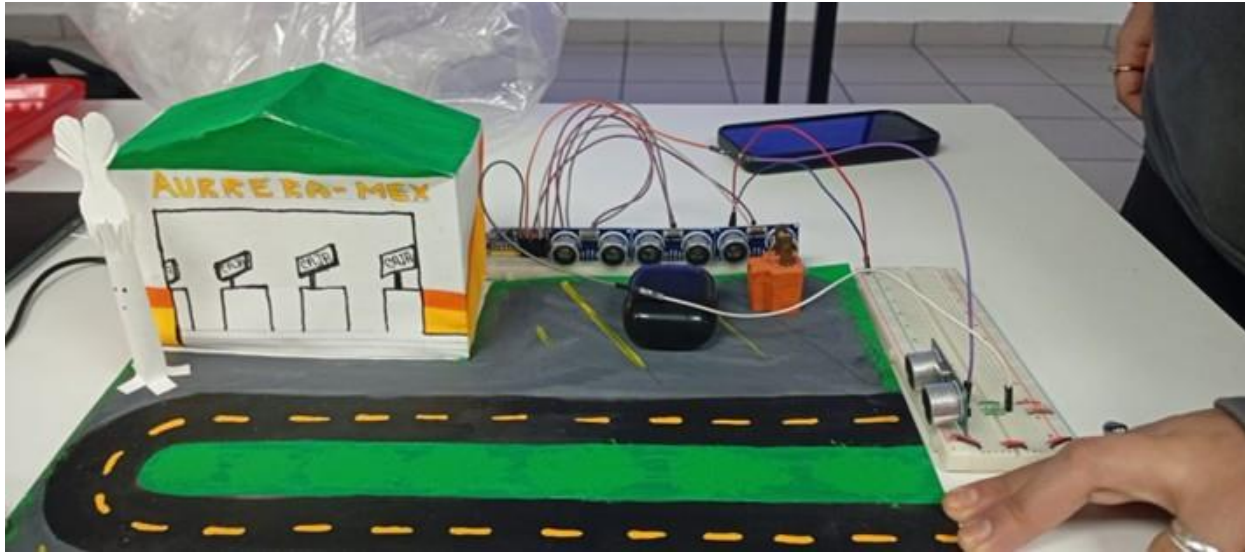
**9.2. Despliegue del Dashboard:** En Node-RED, se diseñó un dashboard que actúa como suscriptor de los tópicos MQTT.

- **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.





Node-RED: Master Dashboard | Node-RED Dashboard | Dashboard Estacionamiento

Estacionamiento (MQTT)

INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO  
SISTEMAS EMBEBIDOS - 2026

ESPACIO 1	ESPACIO 2
✓ LIBRE	🚗 OCUPADO
ESPACIO 3	ESPACIO 4
🚗 OCUPADO	✓ LIBRE

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

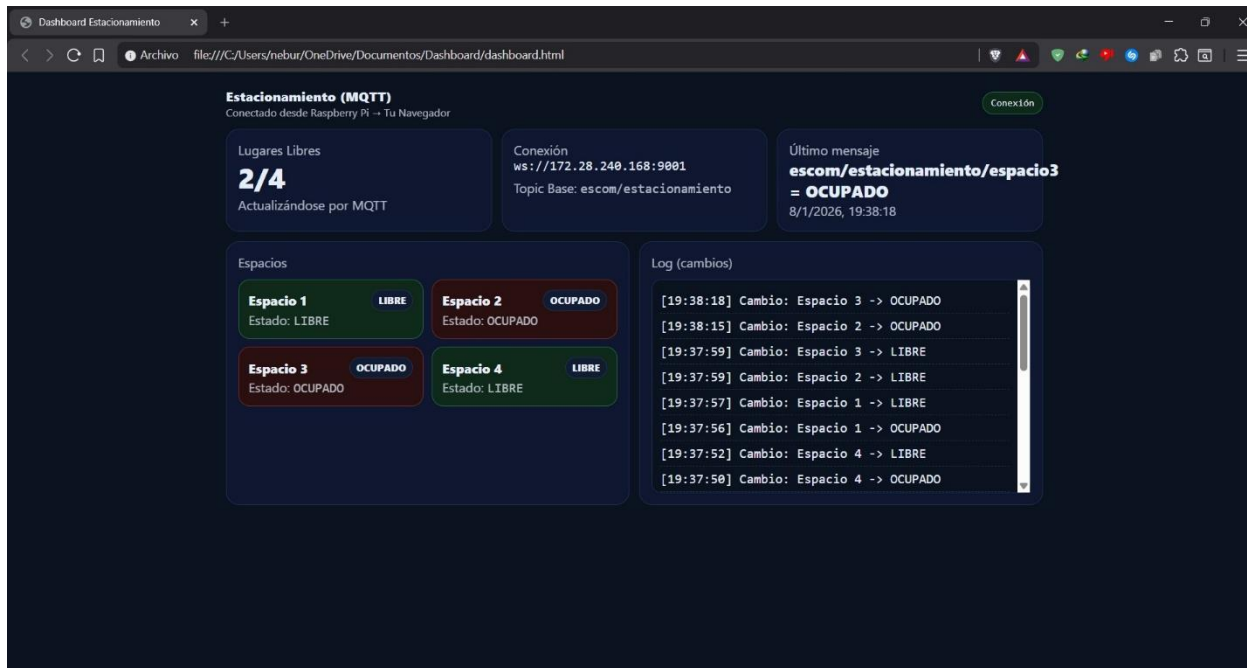
Espacios

Espacio 1	Espacio 2
LIBRE Estado: LIBRE	OCUPADO Estado: OCUPADO
Espacio 3	Espacio 4
OCUPADO Estado: OCUPADO	LIBRE Estado: LIBRE

Log (cambios)

- [19:38:18] Cambio: Espacio 3 -> OCUPADO
- [19:38:15] Cambio: Espacio 2 -> OCUPADO

07:38 p. m.  
08/01/2026



```
172.28.240.168 (raspberrypi) - RealVNC Viewer
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/espacio0 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

4G 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