

# ASIX2 -REPTE

Irene Castillo - Janeth Solano

Tutor -profesor: Jordi Binefa

20/05/2025

## ÍNDICE

<b>INTRODUCCIÓN.....</b>	<b>4</b>
<b>OBJETIVOS.....</b>	<b>5</b>
<b>PROYECTO.....</b>	<b>6</b>
<b>FASE A - Desarrollo de Planificación, Trello, GANTT.....</b>	<b>6</b>
Cuadro de dependencias.....	6
Trello.....	6
GANTT.....	7
<b>FASE B - VM y montaje de la maqueta.....</b>	<b>7</b>
Creación de usuarios.....	7
Creación de subdominios.....	8
Lista de materiales.....	8
Maqueta y conexión parte electrónica.....	10
<b>FASE C - Configuración y desarrollo del programa.....</b>	<b>11</b>
Instalación para el entorno de desarrollo de la Raspberry-Pi.....	11
Otras instalaciones necesarias.....	13
Añadir Red Wifi.....	13
Pantalla OLED.....	15
VPN - Configuración del servicio.....	22
SNAP.....	23
Información de acceso al maquinari.....	26
Creación y desarrollo de códigos del programa de luces.....	26
Código inicial.....	26
Código de testeo.....	28
Código de programa.....	28
Uso del Paho.....	28
Bot de telegram.....	32
Crear contenedor de Node-Red.....	33
Instalar modo dashboard para Node-Red.....	33
Añadir mensajería con Telegram.....	34
Configuración del Node-Red para el ventilador.....	36
Creación del bot.....	40
Configuración del bot.....	42
Añadido para dejar de enviar datos.....	45

Resultados de programa.....	47
Imágenes del estado de ejecución de programa.....	47
Snap.....	49
Python.....	49
<b>FASE D - Testeo y aplicación de correcciones del programa.....</b>	<b>51</b>
Monitorización y implementación de correcciones.....	51
<b>BIBLIOGRAFÍA.....</b>	<b>53</b>

## INTRODUCCIÓN

L'objectiu d'aquest projecte és la implementació dels coneixements adquirits durant aquests dos anys en la creació de programes i serveis per a una maqueta d'un tren.

Les funcions que inclou són: accés remot a través d'una VPN, script en Python utilitzant Paho, MQTT i Snap per controlar els botons i les llums i Node-Red que controla un sensor de CO<sub>2</sub> Virko.

El objetivo de este proyecto es la implementación de los conocimientos adquiridos durante estos dos años en la creación de programas y servicios para una maqueta de un tren.

Las funciones que incluye son: acceso remoto a través de una VPN, script en Python usando Paho, MQTT y Snap para controlar los botones y las luces y Node-Red que controla un sensor de CO<sub>2</sub> Virko.

This project's goal is to implement the knowledge acquired over the past two years in the development of programs and services for a model train.

The included functions are: remote access through a VPN, a Python script using Paho, MQTT and Snap to control buttons and lights and Node-Red to manage a Virko CO<sub>2</sub> sensor.

## **OBJETIVOS**

Integración de conocimientos adquiridos en ASIX1 y ASIX2:

### ➤ **ASIX1**

- SM3 - Fundamentos de programación: C. Olivé, J. Binefa
  - Snap
  - C
  - Python
- PJ1 - Sistemas Operativos Propietarios: J. Moya, J. Fernandez, E. Simón
  - Administración de dominios

### ➤ **ASIX2**

- PJ9 i PJ10 - DevOps i Cloud Computing: D. Collados
  - Dockers
  - Creación de máquinas virtuales en la nube
  - Despliegue de aplicaciones sobre una máquina virtual en la nube
- EH5 - Fundamentos de maquinari: D. Collados
  - Raspberry Pi
- SM6 - Administración de sistemas operativos: J. Fernández
  - Dominios
- SM8 - Servicios de red y internet: R. Catrisse
  - DHCP
  - DNS
- SM9 - Implementación de aplicaciones web: J. Binefa
  - Python
  - Paho
  - MQTT
- SM13 - Seguretat i alta disponibilitat
  - VPN
- PRJ, SM, EH i RPT: Carles Olivé
  - Trello
  - Gantt
  - Tabla de dependencias

## PROYECTO

El proyecto se ha dividido en 4 fases:

- Fase A - Desarrollo de Planificación, Trello, GANTT.
- Fase B - Desarrollo y montaje de la maqueta
- Fase C - Configuración y desarrollo del programa
- Fase D - Testeo y aplicación de correcciones del programa

### FASE A - Desarrollo de Planificación, Trello, GANTT

#### Cuadro de dependencias

Fecha actualización: 19-05-2025

Identificador	Actividad	Dependencias	Tiempo (horas)	Recursos	Estado
<b>Proyecto luminaria</b>					
<b>Fase a - Planificación</b>					
A	Reunión con tutor para inicio de proyecto	-	2	JB, JS	✓
B	Desarrollo del diagrama de Gantt, Trello y tabla de equivalencias del proyecto	A	2	JS, IC	✓
C	Desarrollo y definición del plano de la maqueta	B	4	JB, JS	✓
<b>Fase b - Maquinario</b>					
D	Reunión con tutor	C	3	JB, IC, JS	✓
E	Montaje de la parte electrónica en la maqueta de luminaria	B	3	JS	✓
F	Montaje de la parte del sensor CO2	B	3	JS	En curso
G	Implementación Node Red	B	1	JS	Pendiente
H	Cableado y conexión de maquinario	E-G	1	JS	En curso
<b>Fase c - Software</b>					
I	Reunión con tutor	H	3	JB, IC, JS, GP	✓
J	Analisis y definición del programa	C	3	IC, JS	✓
K	Configuración de las herramientas necesarias para el proyecto	I	3	IC, JS	En curso
L	Configuración de VPN	K	3	GP	✓
M	Desarrollo del programa de luminarias y sensores	K	60	IC, JS	En curso
N	Test del programa de luminaria y sensores	M	30	IC, JS	En curso
<b>Fase d - Pruebas finales</b>					
Ñ	Reunión final con tutor	N	3	JB, IC, JS	Pendiente
O	Implementación de correcciones al programa de luminaria y sensores	Ñ	30	IC, JS	En curso
P	Desarrollo del informe del proyecto	A-O	12	IC, JS	En curso

Trello: <https://trello.com/b/qYhpktb0/proyecto-final>

Recurso	Horas
Janeth (JS)	163
J. Binefa (JB)	15
Gerard (GP)	6
Irene (IC)	149

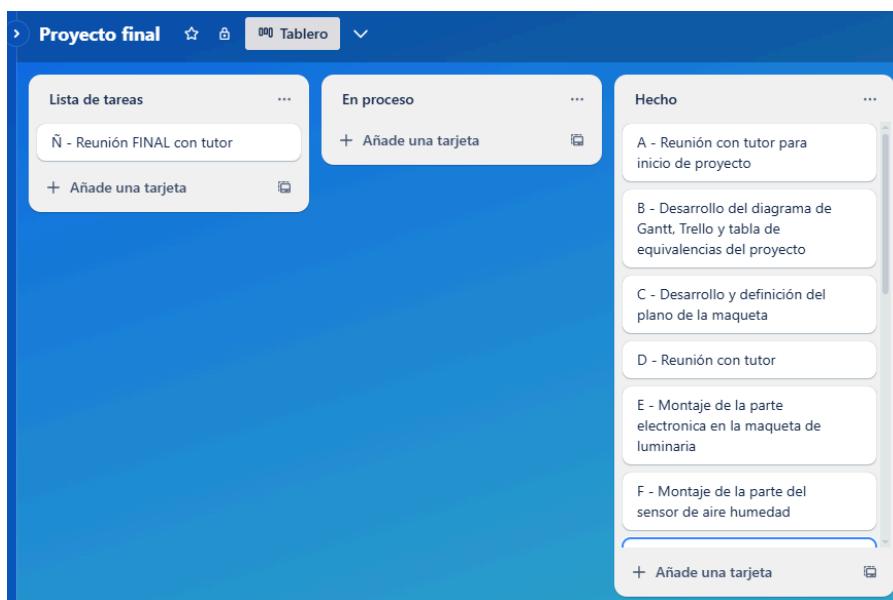
  

d'abril 2025									
Dl	Dm	Dc	Dj	Dv	Ds	Dg			
1	2	3	4	5	6	7	S27		
8	9	10	11	12	13				
14	15	16	17	18	19	20	S28		
21	22	23	24	25	26	27	S29	18	
28	29	30					S30	18	

de maig 2025									
Dl	Dm	Dc	Dj	Dv	Ds	Dg			
1	2	3	4	5	6	7	S31	30	
8	9	10	11	12	13				
15	16	17	18	19	20	21	S32	30	
22	23	24	25	26	27	28	S33	30	
29	30							24	

#### Trello



**Proyecto final**

**Lista de tareas**

- Ñ - Reunión FINAL con tutor
- + Añade una tarjeta

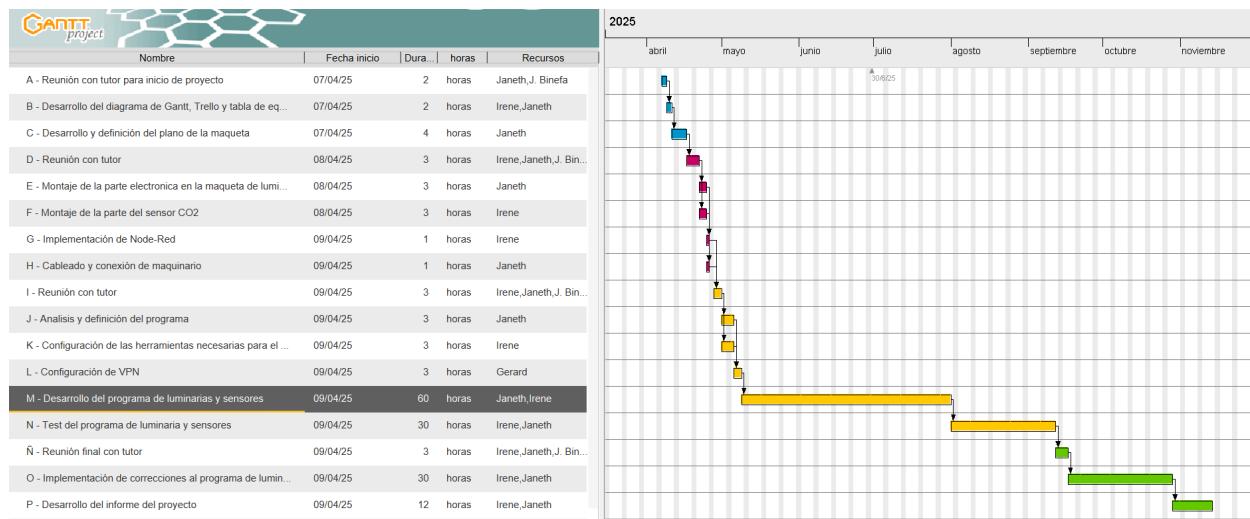
**En proceso**

- + Añade una tarjeta

**Hecho**

- A - Reunión con tutor para inicio de proyecto
- B - Desarrollo del diagrama de Gantt, Trello y tabla de equivalencias del proyecto
- C - Desarrollo y definición del plano de la maqueta
- D - Reunión con tutor
- E - Montaje de la parte electrónica en la maqueta de luminaria
- F - Montaje de la parte del sensor de aire humedad
- + Añade una tarjeta

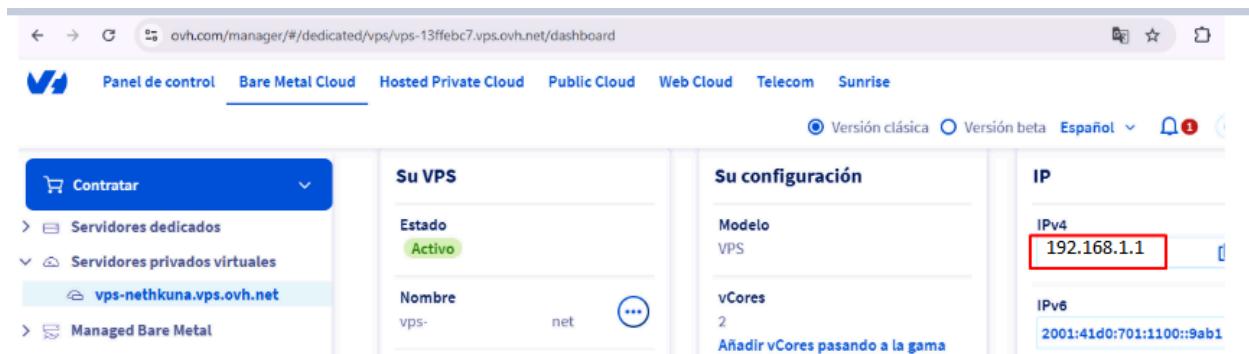
## GANTT



## FASE B - VM y montaje de la maqueta

### Creación de usuarios

Se crea la máquina virtual de OVH.



The screenshot shows the OVH cloud management interface. The top navigation bar includes links for Panel de control, Bare Metal Cloud, Hosted Private Cloud, Public Cloud, Web Cloud, Telecom, and Sunrise. A language switcher shows "Versión clásica" and "Español".

The main area displays a VPS configuration panel:

- Contratar** dropdown menu: Servidores dedicados, Servidores privados virtuales (selected), vps-nethkuna.vps.ovh.net, Managed Bare Metal.
- Su VPS** section: Estado (Activo), Nombre (vps- net).
- Su configuración** section: Modelo (VPS), vCores (2), Añadir vCores pasando a la gama.
- IP** section: IPv4 (192.168.1.1), IPv6 (2001:41d0:701:1100::9ab1).

Creación de usuarios para los miembros del grupo:

**Usuario:** irene

```
sudo useradd irene -u 2003 -g janeth -d /home/irene -m -s /bin/bash -G sudo -p
$(mkpasswd contraseña)
```

```
C:\Users\irene>ssh irene@57.129.43.164
irene@57.129.43.164's password:
Linux vps-13ffebc7 6.1.0-33-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.133-1 (2025-04-10) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

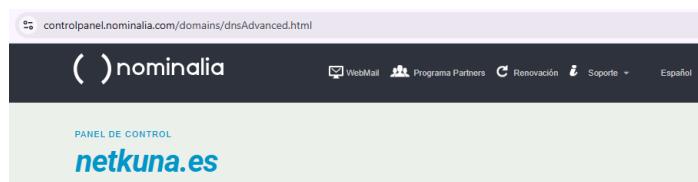
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
irene@vps-13ffebc7:~$
```

**Usuario:** gerard

```
sudo useradd gerard -u 2004 -g janeth -d /home/gerard -m -s /bin/bash -G sudo -p
$(mkpasswd contraseña)
```

### Creación de subdominios

En Nominalia. Hemos creado un subdominio para cada aplicación o servicio que desarrollemos.



<b>Subdominios</b>	
nodered.netkuna.es	Para medir CO2 con un sensor Virko y crear un bot de Telegram
snap.netkuna.es	Crear bloques para el testeo de partes de la maqueta

A continuación se detallan los componentes necesarios para el montaje de la maqueta del proyecto.

### Lista de materiales

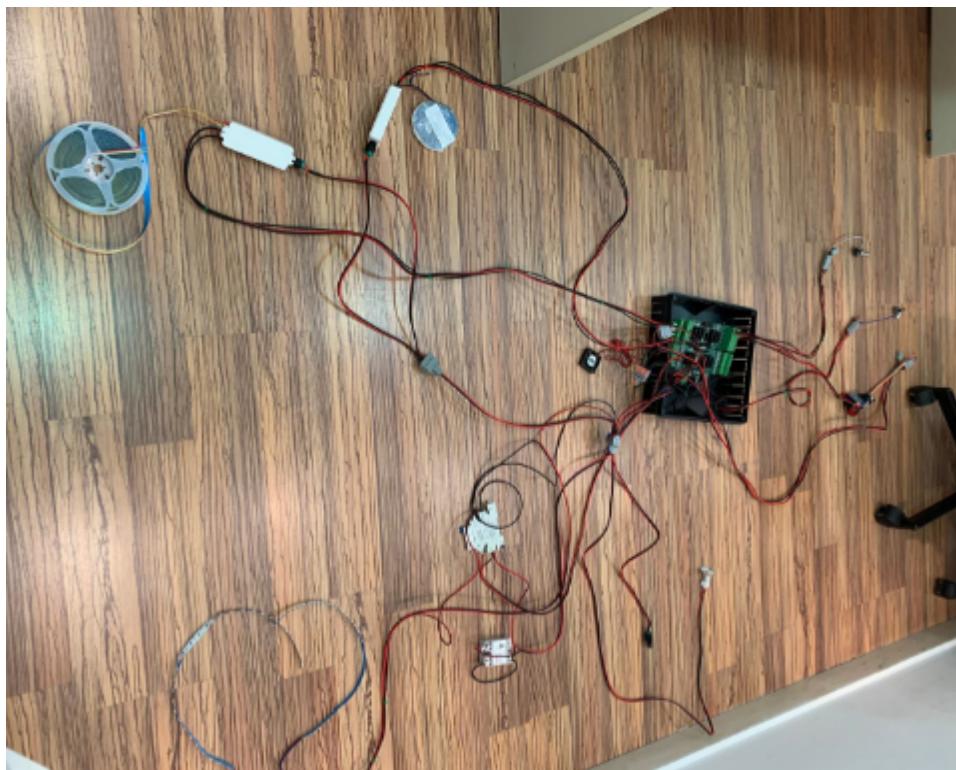
Cantidad	Materiales:
1	Raspberry Pi
1	Máquina virtual
1	MicroSD
---	Tira de leds -luz blanca y RGB
1	Fuente de energía

Cantidad	Materiales:
1	Ladrón de energía eléctrica
2	Luces spotlight
1	Sensor de CO2 Virko
2	Dimmer de leds
1	Botón de stop
1	Ventilador de aire
2	Placas vertebrae
1	Pantalla OLED
1	Champiñon
1	Selector
3	Reguladores de luz
5	Conectores de cable
---	Punteras para los cables
10	Plástico autoblocante
---	Bridas
4	Conectores led de 4 pines
---	Bridas
1	Head
1	Cinta autoadhesiva
1	Embellecedor de canales
30m	Cables de sonido de 1.5mm

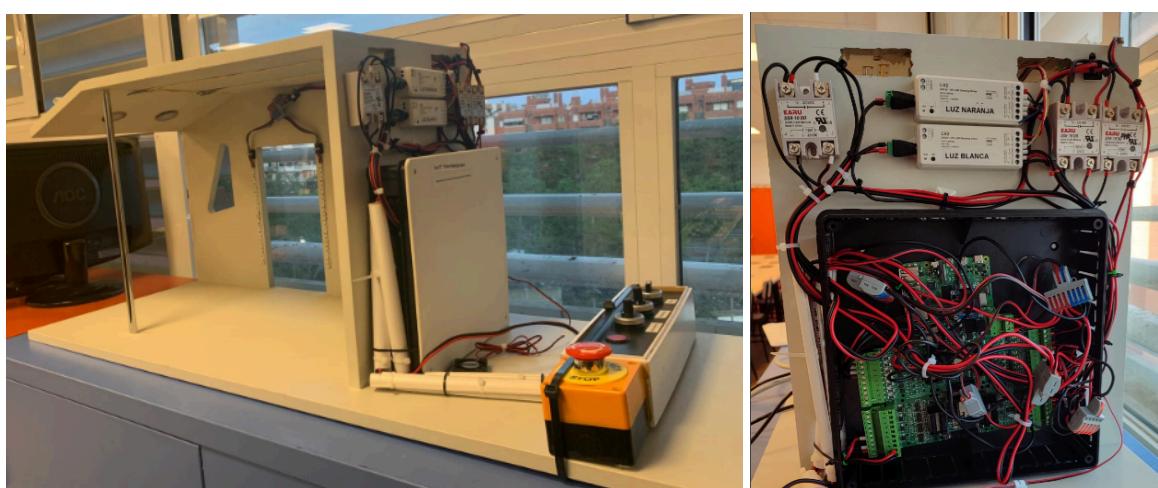
### Maqueta y conexión parte electrónica

Integración de la placa vertebræ, Raspberry Pi, pantalla OLED, luces led, spots, dimmers, reguladores de luz, etc...

### Parte electrónica



### Montaje final





## **FASE C - Configuración y desarrollo del programa**

El detalle de los programas necesarios son los siguientes:

### **Instalación para el entorno de desarrollo de la Raspberry-Pi**

```
sudo apt install python3-dev python3-pip python3-numpy libfreetype6-dev libjpeg-dev  
build-essential  
sudo apt install libsdl2-dev libsdl2-image-dev libsdl2-mixer-dev libsdl2-ttf-dev libportmidi-dev  
sudo apt install python3-smbus i2c-tools python3-pil python3-setuptools python3-rpi.gpio  
python3-venv  
sudo apt install libxml2-dev libxmlsec1-dev  
sudo apt install python3-av zlib1g-dev  
sudo apt install liblcms2-dev libopenjp2-7 libtiff6  
sudo apt install git
```

```
source venv/bin/activate  
python -m pip install - --upgrade pip  
python install - --upgrade pip setuptools wheel  
python -m pip install psutil  
python -m pip install -e .  
python -m pip install - --upgrade luma.oled
```

### **Docker**

```
sudo apt install docker.io -y  
sudo apt install docker-compose -y
```

### **VPN**

```
sudo apt install wireguard - instalación realizada en la máquina virtual
```

Raspberry Pi se ha creado un servicio vpn\_wg.service - el servicio se iniciará cuando se conecte a una red durante el proceso de inicio del sistema.

## **PAHO**

```
source venv/bin/activate  
pip install paho-mqtt  
pip install smbus
```

**Activar entorno virtual** para ejecutar los códigos en paho:

Salir del servicio con el comando: deactivate

```
repte@raspberrypi:~/codis/python $ source venv/bin/activate  
(venv) repte@raspberrypi:~/codis/python $ python paho01.py
```

Instalaciones necesarias:

```
pip install paho-mqtt  
pip install smbus
```

```
C:\Users\Janeth\Documents\Proyecto final\paho>python --version  
Python 3.13.3

C:\Users\Janeth\Documents\Proyecto final\paho>python  
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> quit()

C:\Users\Janeth\Documents\Proyecto final\paho>python paho_subs00.py

C:\Users\Janeth\Documents\Proyecto final\paho>python paho_subs00.py  
Traceback (most recent call last):  
  File "C:\Users\Janeth\Documents\Proyecto final\paho\paho_subs00.py", line 3, in <module>  
    import paho.mqtt.subscribe as subscribe  
ModuleNotFoundError: No module named 'paho'
```

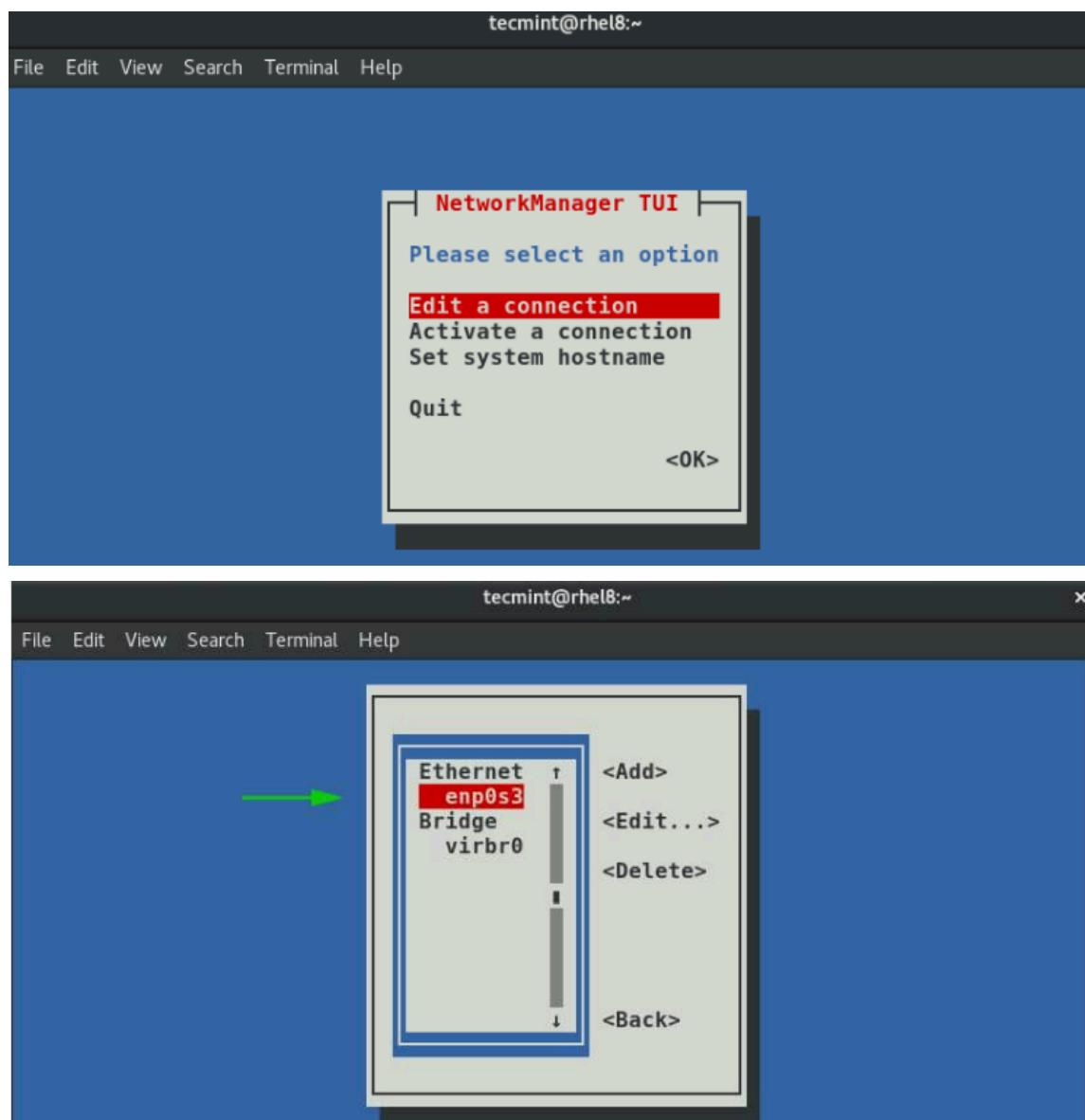
## Otras instalaciones necesarias

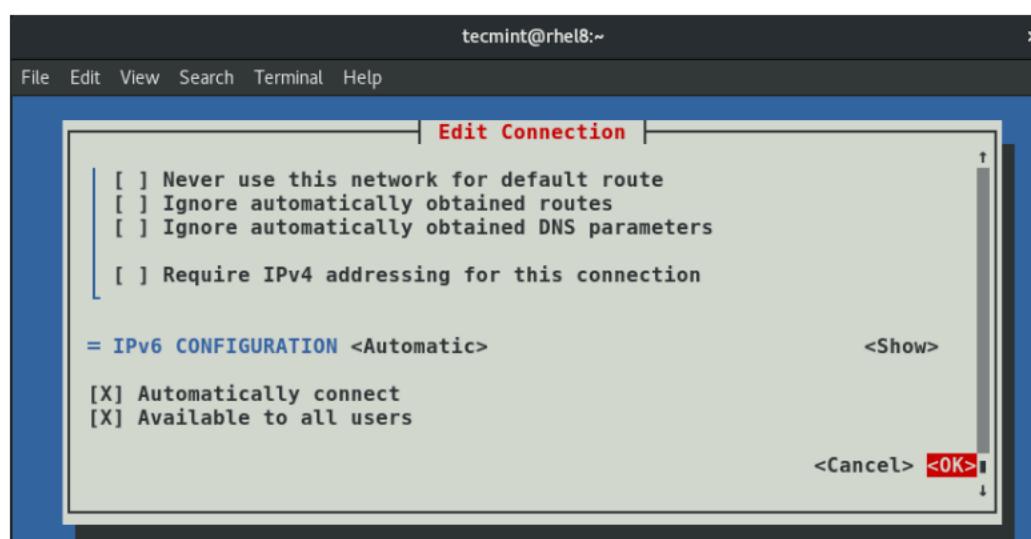
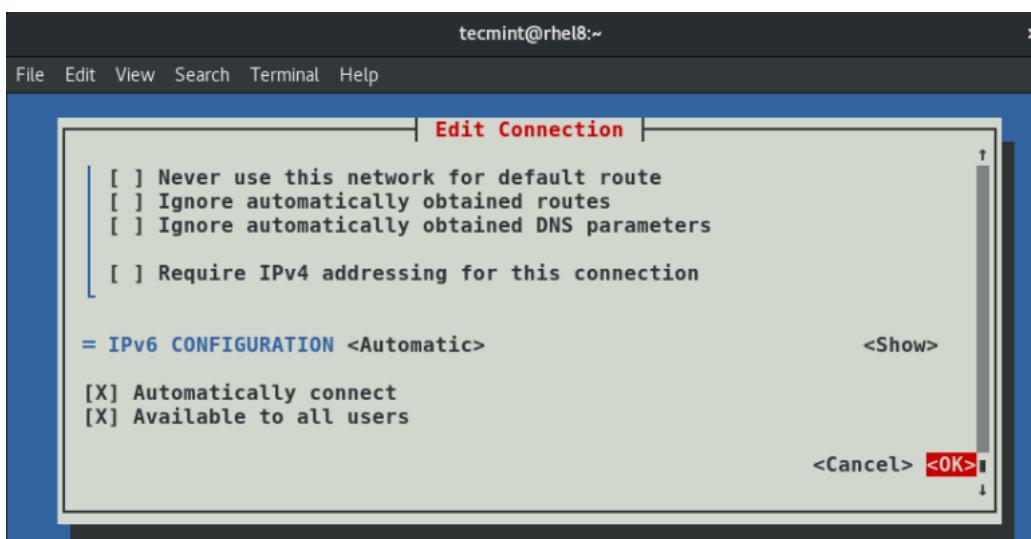
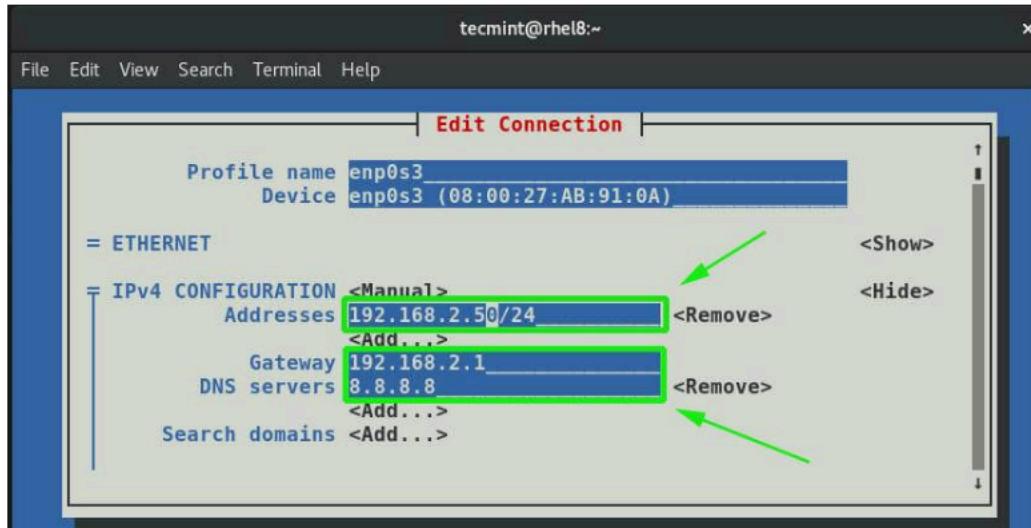
Configuración de la pantalla OLED para usar el bus I2C con los vertebræ, añadir red WIFI, pantalla OLED y la VPN.

### Añadir Red Wifi

En la Raspberry Pi ejecutar:

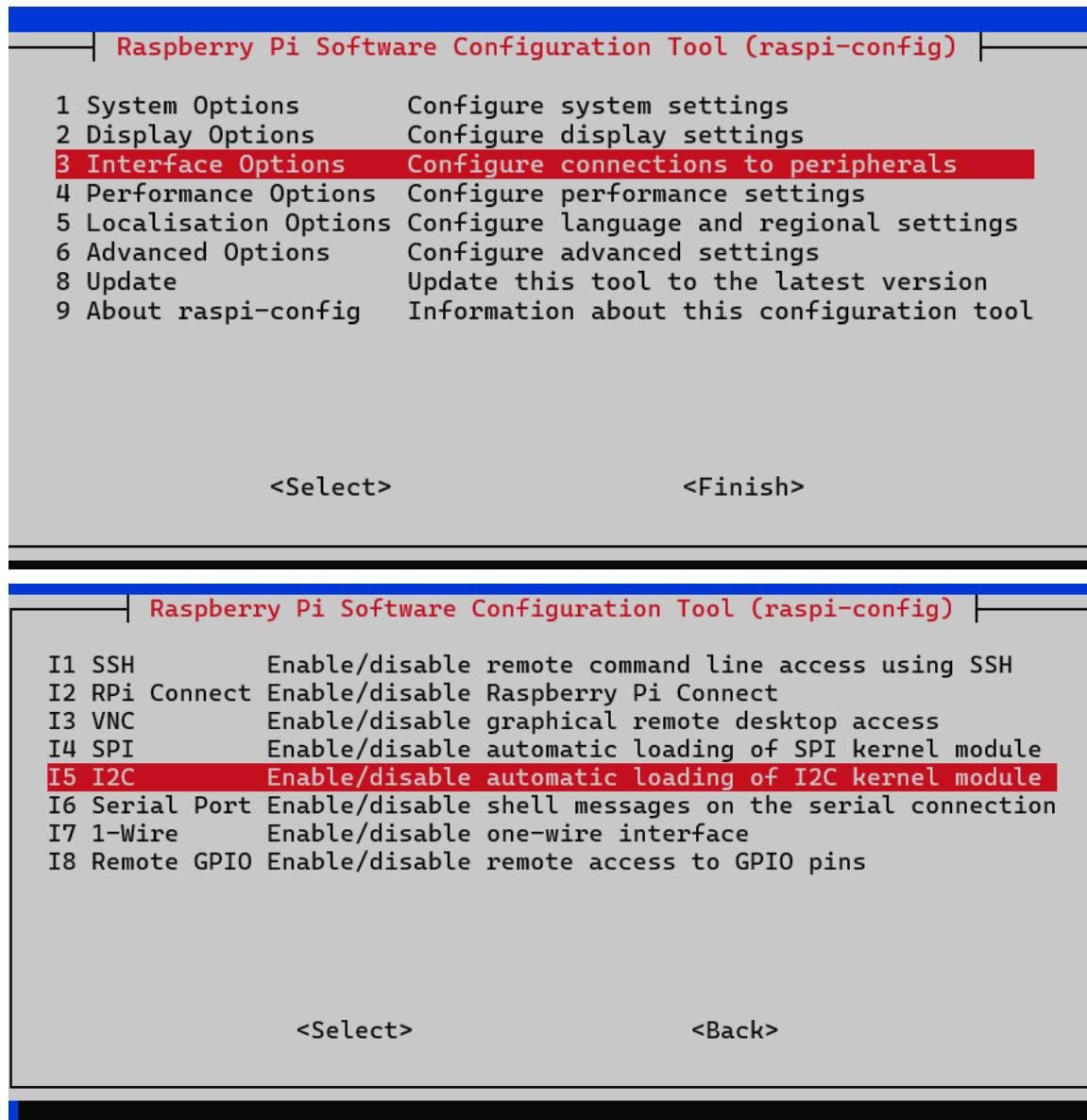
```
sudo nmtui
```





### Pantalla OLED

sudo raspi-config



Would you like the ARM I2C interface to be enabled?

<Yes>

<No>

The ARM I2C interface is enabled

<Ok>

Para hacer servir el bus CAN desde la Raspberry Pi:

```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options      Configure system settings
2 Display Options    Configure display settings
3 Interface Options  Configure connections to peripherals
4 Performance Options Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options   Configure advanced settings
8 Update             Update this tool to the latest version
9 About raspi-config  Information about this configuration tool

<Select>           <Finish>

Raspberry Pi Software Configuration Tool (raspi-config)

I1 SSH                Enable/disable remote command line access using SSH
I2 RPi Connect        Enable/disable Raspberry Pi Connect
I3 VNC                Enable/disable graphical remote desktop access
I4 SPI                Enable/disable automatic loading of SPI kernel module
I5 I2C                Enable/disable automatic loading of I2C kernel module
I6 Serial Port        Enable/disable shell messages on the serial connection
I7 1-Wire             Enable/disable one-wire interface
I8 Remote GPIO         Enable/disable remote access to GPIO pins

<Select>           <Back>
```

Would you like the SPI interface to be enabled?

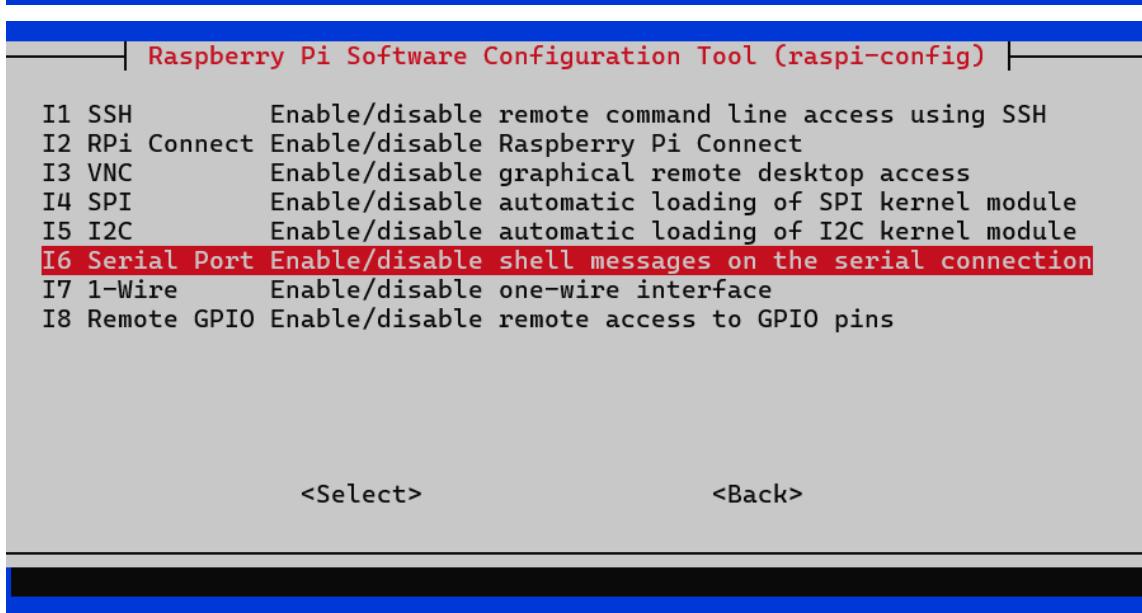
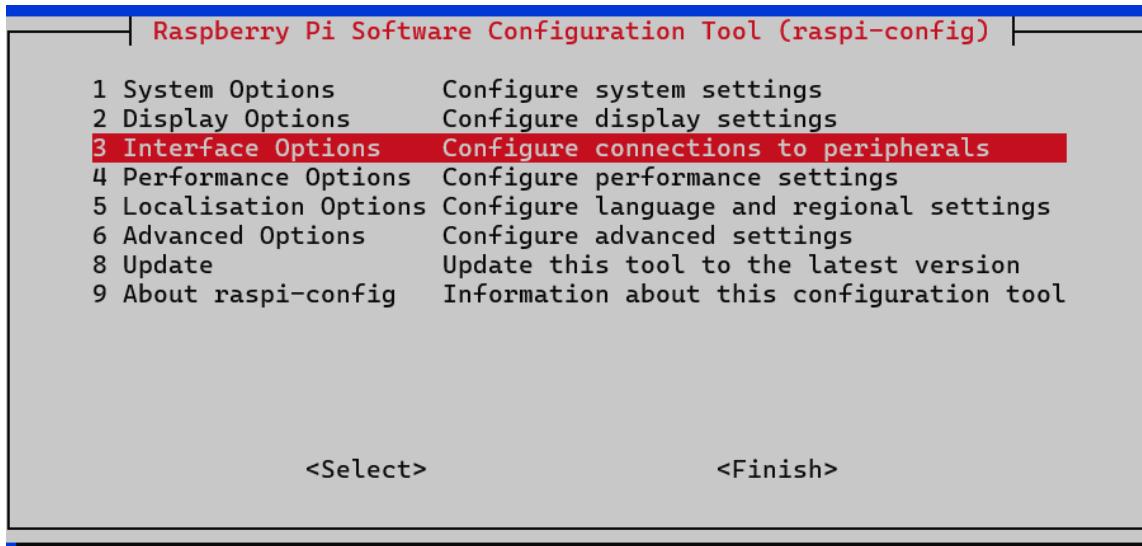
<Yes>

<No>

The SPI interface is enabled

<Ok>

Para hacer servir el bus ModBus RTU desde la Raspberry Pi:



Would you like a login shell to be accessible over serial?

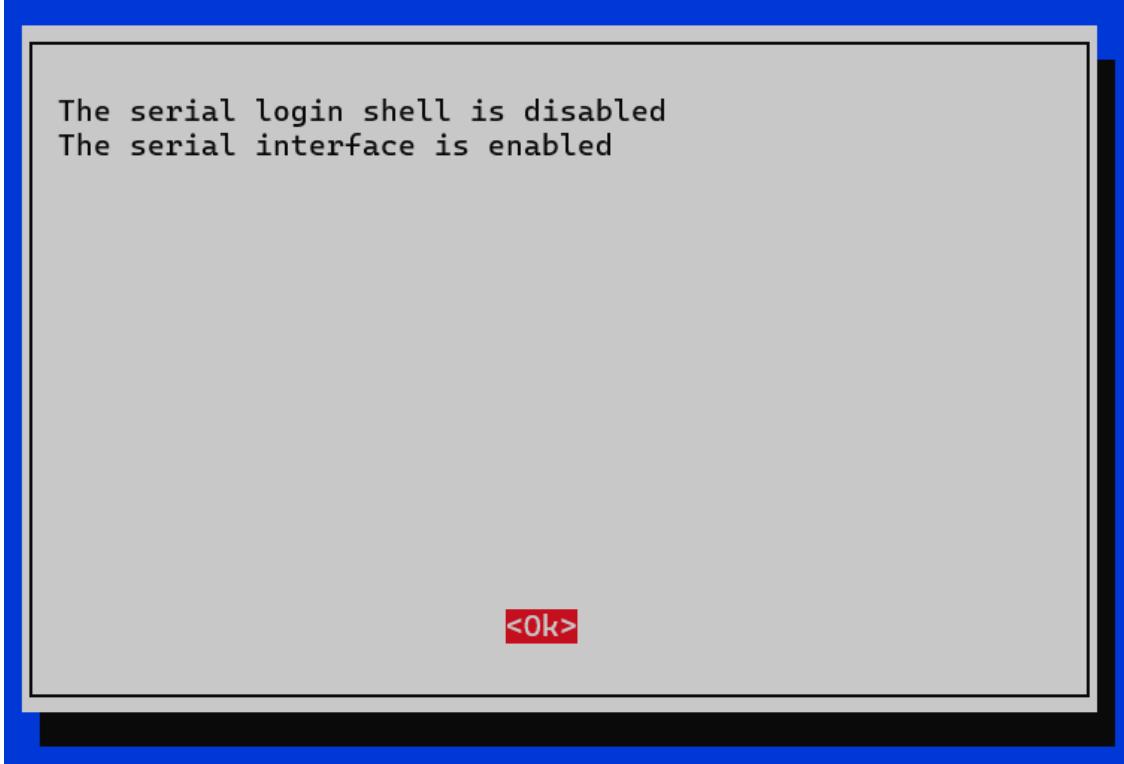
<Yes>

<No>

Would you like the serial port hardware to be enabled?

<Yes>

<No>



```
The serial login shell is disabled
The serial interface is enabled
```

<0k>

Después de hacer la configuración, reiniciar.



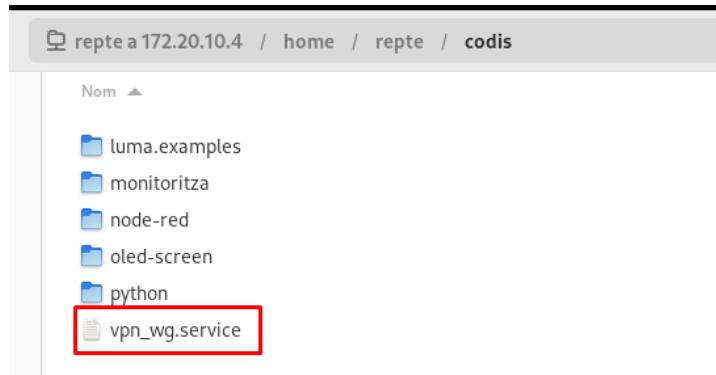
```
Would you like to reboot now?
```

<Yes>

<No>

## VPN - Configuración del servicio

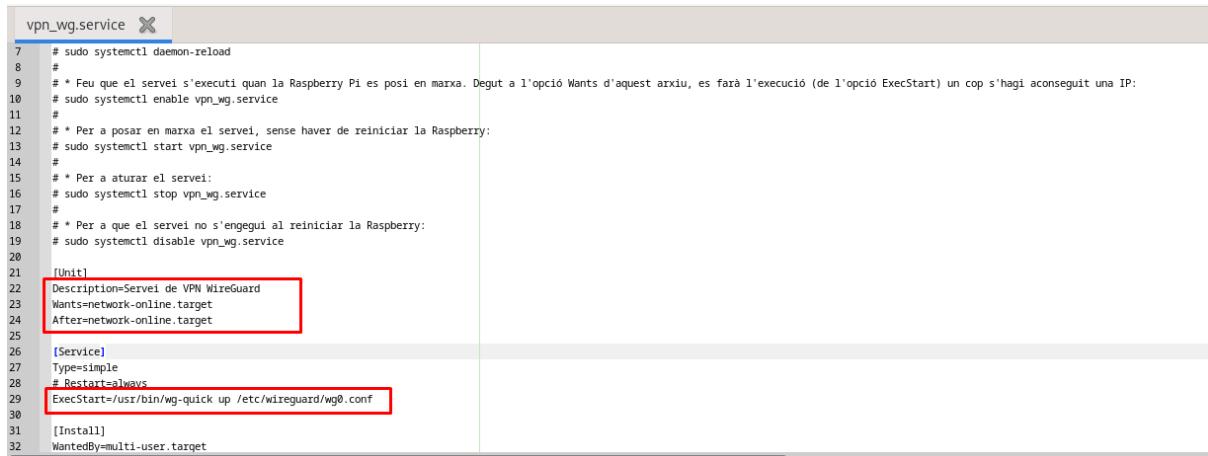
En la carpeta codis se crea el fichero: vpn\_wg\_service



Para configurar se sigue el ejemplo del contenido del siguiente enlace:

[https://recull.binefa.cat/files/IoT-Vertebrae/python/alternativa\\_oled/oled\\_head.service](https://recull.binefa.cat/files/IoT-Vertebrae/python/alternativa_oled/oled_head.service)

```
> ... > python > alternativa_oled > oled_head.service
1 # Contents of /etc/systemd/system/
2 # Copieu aquest arxiu a /etc/systemd/system/
3 # sudo cp oled_head.service /etc/systemd/system/
4 # La ruta absoluta serà: /etc/systemd/system/oled_head.service
5 #
6 # * Recarregueu els serveis:
7 # sudo systemctl daemon-reload
8 #
9 # * Feu que el servei s'executi quan la Raspberry Pi es posi en marxa. Degut a l'opció Wants d'aquest arxiu, es farà l'execució (de l'opció ExecStart) un cop s'hagi aconseguit una IP:
10 # sudo systemctl enable oled_head.service
11 #
12 # * Per a posar en marxa el servei, sense haver de reiniciar la Raspberry:
13 # sudo systemctl start oled_head.service
14 #
15 # * Per a aturar el servei:
16 # sudo systemctl stop oled_head.service
17 #
18 # * Per a que el servei no s'engegui al reiniciar la Raspberry:
19 # sudo systemctl disable oled_head.service
20
21 [Unit]
22 Description=Oled IoT-Vertebrae Head Service
23 Wants=network-online.target
24 After=network-online.target
25
26 [Service]
27 Type=simple
28 # Restart=always
29 ExecStart=/home/pi/codis/oled-screen/venv/bin/python /home/pi/codis/luma.examples/examples/sys_info_extended.py
30
31 [Install]
32 WantedBy=multi-user.target
```



```
vpn_wg.service
1 # sudo systemctl daemon-reload
2 #
3 # Feu que el servei s'executi quan la Raspberry Pi es posi en marxa. Degut a l'opció Wants d'aquest arxiu, es farà l'execució (de l'opció ExecStart) un cop s'hagi aconseguit una IP:
4 # sudo systemctl enable vpn_wg.service
5 #
6 # * Per a posar en marxa el servei, sense haver de reiniciar la Raspberry:
7 # sudo systemctl start vpn_wg.service
8 #
9 # * Per a aturar el servei:
10 # sudo systemctl stop vpn_wg.service
11 #
12 # * Per a que el servei no s'engegui al reiniciar la Raspberry:
13 # sudo systemctl disable vpn_wg.service
14
15 [Unit]
16 Description=Servei de VPN WireGuard
17 Wants=network-online.target
18 After=network-online.target
19
20 [Service]
21 Type=simple
22 # Restart=always
23 ExecStart=/usr/bin/wg-quick up /etc/wireguard/wg0.conf
24
25 [Install]
26 WantedBy=multi-user.target
```

Recargar los servicios:

```
repte@raspberrypi:~/codis $ sudo systemctl daemon-reload
repte@raspberrypi:~/codis $ sudo systemctl enable vpn_wg.service
Created symlink /etc/systemd/system/multi-user.target.wants/vpn_wg.service → /etc/systemd/system/vpn_wg.service.
```

```
repte@raspberrypi:~/codis $ sudo systemctl start vpn_wg.service
```

```
repte@raspberrypi:~/codis $ sudo reboot
```

Se ha comprobado que se puede acceder por VPN una vez que la Raspberry Pi se conecta a la red.

Comprobación de acceso:

```
janeth@vps-13ffebc7:~$ ssh repe@10.8.0.4
repte@10.8.0.4's password:
Linux raspberrypi 6.12.25+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.25-1+rpt1 (202
5-04-30) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed May 14 18:07:28 2025 from 192.168.2.140
repte@raspberrypi:~ $ date
Wed 14 May 18:11:21 BST 2025
```

## SNAP

Snap inicial: <https://snap.binefa.cat/snap.html> a partir del que se va a configurar el snap.netkuna.es.

## Poner snap en el dominio

Para poner snap en el dominio, tenemos de descargar el código fuente desde el github y luego moverlo a la carpeta var/www/html de nuestra máquina virtual.

```
irene@vps-13ffebc7:/var/www/html$ git clone https://github.com/jmoenig/Snap.git
Cloning into 'Snap'...
remote: Enumerating objects: 45101, done.
remote: Counting objects: 100% (114/114), done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 45101 (delta 77), reused 61 (delta 34), pack-reused 44987 (from 1)
Receiving objects: 100% (45101/45101), 1.40 GiB | 41.66 MiB/s, done.
Resolving deltas: 100% (33077/33077), done.
irene@vps-13ffebc7:/var/www/html$ ls
Snap index.html
```

Crear un docker con las instrucciones a continuación: [https://www.things.cat/index.php/Apache\\_i\\_PHP](https://www.things.cat/index.php/Apache_i_PHP)

The screenshot shows a terminal window with several tabs open. The active tab displays a Dockerfile with the following content:

```
FROM php:7.0-apache
COPY . /var/www/html
```

A red arrow points from the text "Exemple d'arxiu Dockerfile:" in the main content area to the word "Dockerfile" in the terminal output. The terminal output shows the creation of the Dockerfile and its subsequent build:

```
irene@vps-13ffebc7:~/var/www/html/Snap$ nano Dockerfile
irene@vps-13ffebc7:~/var/www/html/Snap$ docker build -t isnap .
Permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://127.0.0.1:2375/v1.24/build?buildargs=%7B%7D&cachefrom=&cgroupparent=&cputier=0&cputotal=0&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&netwrokmode=default&rm=1&shmsize=0&t=isnap&target=&ulimits=null&version=1": dial unix /var/run/docker.sock: connect: permission denied
irene@vps-13ffebc7:~/var/www/html/Snap$ sudo docker build -t isnap .
[sudo] password for irene:
Sending build context to Docker daemon 1.7GB
Step 1/2 : FROM php:7.0-apache
7.0-apache: Pulling from library/php
177e7ef0d6f9: Pull complete
9b8f92eda24: Pull complete
350207dcf1b7: Pull complete
a8a33d96b4e7: Pull complete
c0421d5b63d6: Pull complete
f76e300fbe72: Pull complete
af9ff1b9ce5b: Pull complete
d9f072d61771: Pull complete
37007e292198: Pull complete
8ba923990f24: Pull complete
98af8902979a: Pull complete
f1548c2cd376: Pull complete
e1062fd0605a: Pull complete
Digest: sha256:1d34b2e491a02ba7a8d26478132015e197a5ffea37f0a93b42621d11cfe04
2cc
Status: Downloaded newer image for php:7.0-apache
--> aa67a9c9814f
Step 2/2 : COPY . /var/www/html
--> d81f946ab3ec
Successfully built d81f946ab3ec
Successfully tagged isnap:latest
```

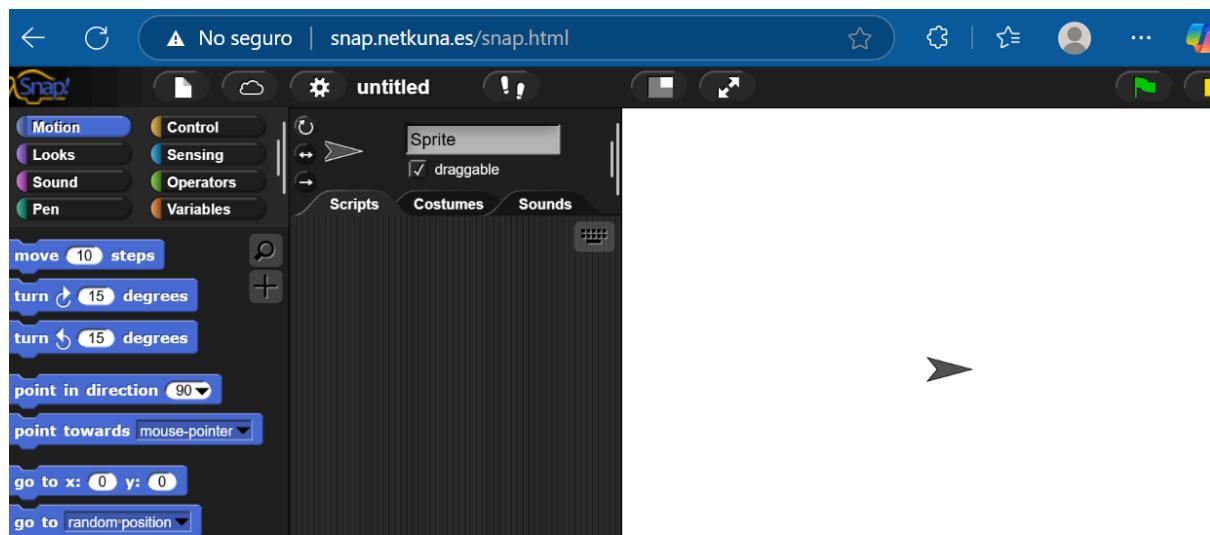
Activamos el docker con este comando:

```
docker run -dit \
--name snap \
-v /var/www/html/Snap:/var/www/html \
--network net \
-e VIRTUAL_HOST="snap.netkuna.es" \
-e LETSENCRYPT_HOST="snap.netkuna.es" \
-e VIRTUAL_PORT=80 \
-e TZ="Europe/Andorra" \
isnap
```

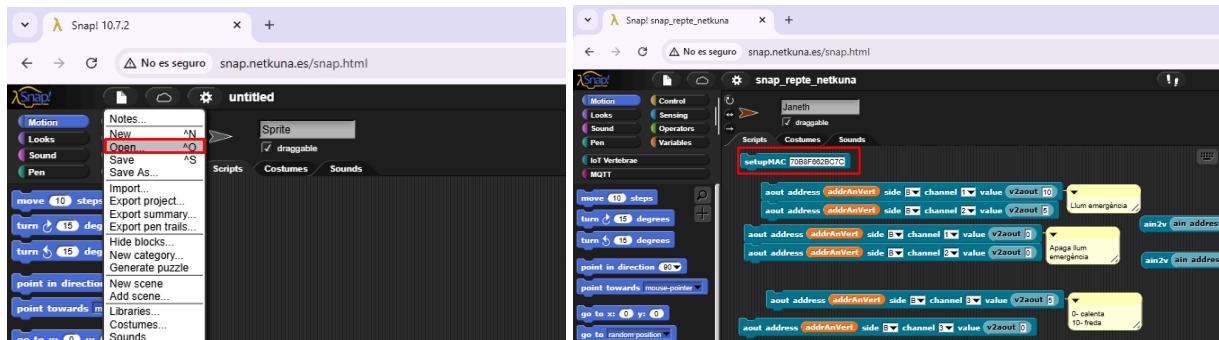
Además tenemos que activar el proxy inverso

```
irene@vps-13ffebc7:/var/www/html/Snap$ sudo docker ps -a
CONTAINER ID        IMAGE               CREATED             STATUS              NAMES
29ce1d7997af      isnap               20 seconds ago   Up 19 seconds   "docker-php-e
ntrypoi..."        apache:7.4        2 days ago       Up 23 hours     "docker-php-e
:::8080->80/tcp    apache               ntrypoi...
434649153c79      ipreus:1.0       2 weeks ago      Exited (0) 9 days ago
ntrypoi..."        determined_volhard "docker-php-e
33a45eba7f62      ipreus:1.0       2 weeks ago      Exited (0) 9 days ago
ntrypoi..."        cpreus               "docker-php-e
761372fa38e8      sm9-bd-php:latest  6 months ago     Exited (0) 2 weeks ago
ntrypoi..."        sm9_clot_netkuna_es "docker-php-e
303e42fb9b5b      mysql:8.1.0      6 months ago     Exited (0) 2 weeks ago
point.s..."         sm9-mysql_db_docker "docker-entry
522c00eb05e6      jrcs/letsencrypt-nginx-proxy-companion:latest "/bin/bash /a
pp/entr..."        6 months ago     Exited (0) 3 days ago
letsencrypt-helper
e3e271baaa35      jwilder/nginx-proxy:latest      "/app/docker-
entrypo..."        6 months ago     Exited (2) 3 days ago
reverse-proxy
irene@vps-13ffebc7:/var/www/html/Snap$ sudo docker start reverse-proxy
reverse-proxy
```

Luego al poner en el buscador [snap.netkuna.es](http://snap.netkuna.es) ya aparece el Snap.



Se importa el código inicial del [snap.binefa.com](http://snap.binefa.com) a [snap.netkuna.es](http://snap.netkuna.es)



### Información de acceso al maquinari

Acceso desde linux:

```
paho01.py
sftp://repte@192.168.1.1/home/repte/codis/python
```

sftp://repte@172.XX.XX.X/home/repte/codis/python  
sftp://repte@192.168.X.XXX/home/repte/codis/python

Accedo desde windows cmd:

ssh repte@172.XX.XX.X/home/repte/codis/python  
ssh repte@192.168.X.XXX/home/repte/codis/python

Acceso por vpn solo desde la máquina virtual de OVH (5X.XXX.XX.XXX):

ssh repte@10.X.X.X

### Creación y desarrollo de códigos del programa de luces

Se han usado los [códigos Python de ejemplo de uso del IoT-Vertebrae PLC Edu](#). A continuación se detalla información inicial con la que se va a desarrollar el programa.

#### Código inicial

```
>>> from i2c_iotv import *
>>> doutbit("0000","B",3,1) # activar luces rojas
>>> doutbit("0000","B",3,0) # apagar luces rojas
>>> doutbit("0000","B",2,1) # encender luz botón
>>> doutbit("0000","B",3,1) # encender luces rojas con e luz l botón al mismo tiempo
>>> doutbit("0000","B",3,0) # apagar luces rojas
>>> doutbit("0000","B",2,0) # apagar luz botón
>>> doutbit("0000","B",4,1) # encender luces verdes
>>> doutbit("0000","B",4,0) # apagar luces verdes
```

```
>>> doutbit("0000","B",1,1) # encender ventilador
>>> doutbit("0000","B",1,0) # apagar ventilador
>>> doutbit("0000","B",0,1) # encender spot
>>> doutbit("0000","B",0,0) # apagar spot
```

Creación de los archivos iniciales.

Tipo de salida/entrada	Nombre de archivo	Descripción
Salida-Digital	dout00.py	Puerta parpadeo luz rojo - verde rojo - verde
Salida-Digital	dout01.py	Spot – luz encendida siempre
Salida-Digital	dout02.py	Ventilador – encender y apagar
Entrada-Digital	din00.py	Botón stop
Entrada-Digital	din01.py	Selector
Entrada-Digital	din02.py	PRUEBAS - Botón stop de emergencia
Salida-Analógica	aout00.py	Luz techo
Salida-Analógica	aout01.py	Luz blanca de techo
Entrada-Analogica	ain00.py	Potenciómetros
Entrada-Analogica	ain01.py	Potenciómetro de brillo
Entrada-Analogica	ain02.py	Potenciómetro de temperatura

### Código de testeo

Descripción de los archivos de testeo.

Tipo de salida/entrada	Nombre de archivo	Descripción
--	codigo_general.py	Implementación de todos los códigos
--	test.py, test01.py	Pruebas de luces
--	paho01.py	Implementación de partes de códigos de luces

### Código de programa

Descripción de los archivos de programa.

Tipo de salida/entrada	Nombre de archivo	Descripción
--	main_paho.py	Código de ejecución del primary.py para evitar el uso de while
--	primary.py	Todo el código principal
--	paho02.py	Fragmento de MQTT que incluye el ventilador

### Uso del Paho

Pasos a seguir para activar el entorno virtual para ejecutar los códigos del paho:

source venv/bin/activate

```
repte@raspberrypi:~/codis/python $ source venv/bin/activate
(venv) repe@raspberrypi:~/codis/python $ python paho01.py
```

Salir del servicio con el comando: deactivate

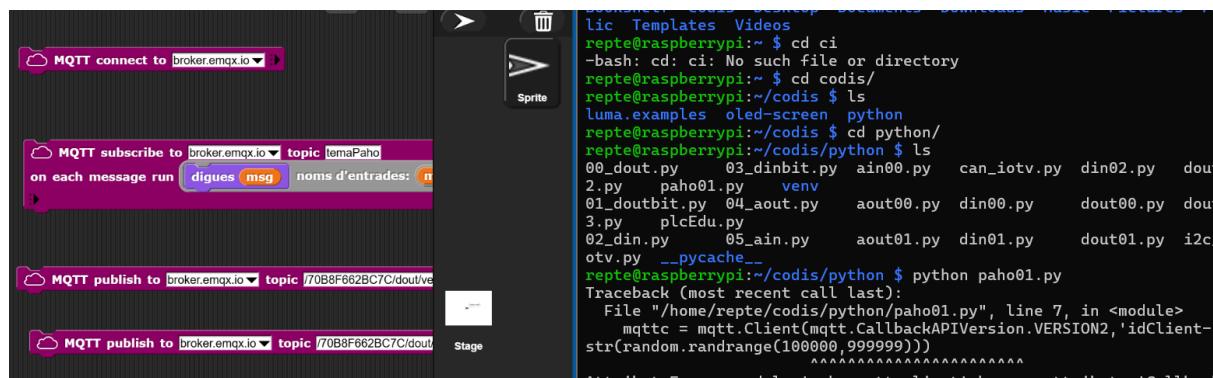
```
(venv) repe@raspberrypi:~/codis/python $ deactivate
repte@raspberrypi:~/codis/python $
```

Cuando se ejecuta el paho\_pub00.py, se reciben los resultados en paho\_subs00.py

```
C:\Users\irene\Desktop\Proyecto final\paho>python paho_pub00.py
C:\Users\irene\Desktop\Proyecto final\paho>python paho_pub00.py
C:\Users\irene\Desktop\Proyecto final\paho>
```

```
C:\Users\irene\Desktop\Proyecto final\paho>python paho_subs00.py
topic = temaPaho
payload = Escola del Clot
topic = temaPaho
payload = Escola del Clot
```

Desde el Snap si ejecutamos el publish y en la raspberry tenemos funcionando el paho01.py, decimos que se enciendan las luces verdes con 1 y que se apaguen con 0.



Cuando hagamos un cambio en el código para que funcione se tiene que cerrar el programa y volver a abrirlo para que se actualice.

### paho\_subs00.py

```
import random,time

import paho.mqtt.subscribe as subscribe
import paho.mqtt.client as mqtt
import paho.mqtt.properties as properties

mqttc = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2,'idClient-'+str(random.randrange(100000,999999)))
# mqttc.username_pw_set("usuari","contrasena");
mqttc.connect("broker.emqx.io", 1883)

def on_message(clientId, userdata, message):
    msg = str(message.payload.decode("utf-8"))
    print('topic = ', message.topic)
    print('payload = ', msg)

mqttc.on_message = on_message
mqttc.subscribe('temaPaho')
```

```
mqttc.loop_start()  
# n = 0  
while True:  
    print("",end="")  
    time.sleep(1)
```

### paho\_pub00.py

```
import random,time  
  
import paho.mqtt.subscribe as subscribe  
import paho.mqtt.client as mqtt  
import paho.mqtt.properties as properties  
  
mqttc = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2,'idClient-'+str(random.randrange(100000,999999)))  
# mqttc.username_pw_set("usuari","contrasenya");  
mqttc.connect("broker.emqx.io", 1883)  
  
mqttc.publish('temaPaho','Escola del clot')
```

### paho01.py

```
# mqttc.username_pw_set("usuari","contrasenya");  
mqttc.connect("broker.emqx.io", 1883)  
  
def on_message(clientId, userdata, message):  
    msg = str(message.payload.decode("utf-8"))  
    print('topic = ', message.topic)  
    print('payload = ', msg)  
    if message.topic == "/70B8F662BC7C/dout/verd":  
        if msg == '1':  
            doutbit('0000','B',4,1)  
        else:  
            doutbit('0000','B',4,0)  
  
mqttc.on_message = on_message  
mqttc.subscribe('/70B8F662BC7C/#')
```

```
mqttc.loop_start()  
# n = 0  
while True:  
    print("",end="")  
    time.sleep(1)
```

### paho01.2.py

```
import random,time  
from i2c_iotv import *  
from aout00 import lnaranja  
from din00 import boto  
import paho.mqtt.subscribe as subscribe  
import paho.mqtt.client as mqtt  
import paho.mqtt.properties as properties  
  
mqttc =  
mqtt.Client(mqtt.CallbackAPIVersion.VERSION2,'idClient-'+str(random.randrange(100000,999999)))  
# mqttc.username_pw_set("usuari","contrasenya");  
mqttc.connect("broker.emqx.io", 1883)  
  
def on_message(clientId, userdata, message):  
    msg = str(message.payload.decode("utf-8"))  
    print('topic = ', message.topic)  
    print('payload = ', msg)  
    # Luz verde  
    if message.topic == "/70B8F662BC7C/dout/verd":  
        if msg == '1':  
            doutbit('0000','B',4,1)  
        else:  
            doutbit('0000','B',4,0)  
    # Luz roja  
    if message.topic == "/70B8F662BC7C/dout/vermell":  
        if msg == '1':  
            doutbit('0000','B',3,1)  
        else:  
            doutbit('0000','B',3,0)
```

```

# Luz naranja
if message.topic == "/70B8F662BC7C/dout/emergencia":
    if msg == '1':
        lnaranja()
    else:
        aout("0000","B",1,v2aout(0))
        aout("0000","B",2,v2aout(0))

# Luz stop
if message.topic == "/70B8F662BC7C/dout/stop":
    if msg == '1':
        boto()
    else:
        doutbit("0000","B",2,0)

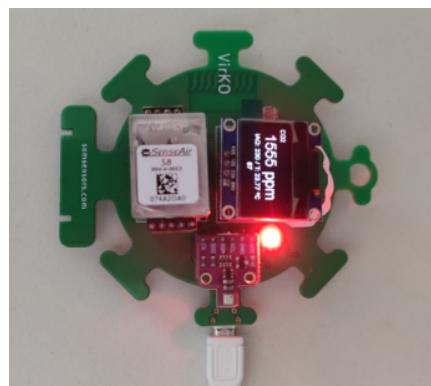
mqttc.on_message = on_message
# El hashtag sirve para decir que se suscriba a todos los temas
mqttc.subscribe('/70B8F662BC7C/#')

mqttc.loop_start()
# n = 0
while True:
    print("",end="")
    time.sleep(1)

```

### Bot de telegram

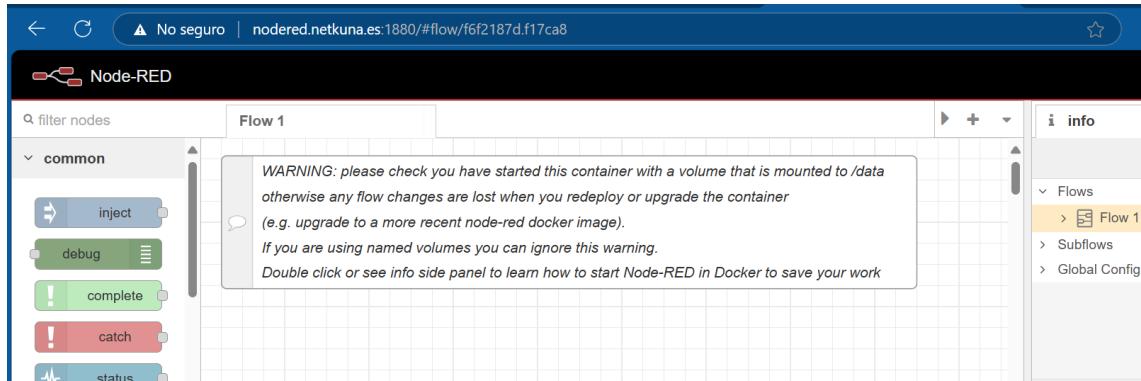
Imagen de equipamiento utilizado:



### Crear contenedor de Node-Red

Creamos un docker para poder acceder al Node-Red:

```
docker run -itd -p 1880:1880 -v dades:/data --device /dev/i2c-1 --user node-red:994 --name dockerNodeRed nodered/node-red
```



Tutorial de como usar Node-Red -> [Node-RED Hello, world! Example | Opto 22 Developer](#)

### Instalar modo dashboard para Node-Red

Acceder al bash del docker de Node-Red:

```
docker exec -it dockerNodeRed bash
```

```
irene@vps-13ffebc7:~$ sudo docker exec -it dockerNodeRed bash
de15f3d7f8f7:~$ npm i node-red-dashboard

added 25 packages, and audited 337 packages in 3s

63 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 10.8.2 → 11.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.3.0
npm notice To update run: npm install -g npm@11.3.0
npm notice
```

Una vez dentro del contenedor ejecutar la orden para descargar el módulo:

```
npm i node-red-dashboard
```

Después hemos de acceder a la carpeta correspondiente y ejecutar:

```
cd node_modules
cd node-red-dashboard
npm install
```

```
npm notice
de15f3d7f8f7:~$ cd node_modules/
de15f3d7f8f7:~/node_modules$ cd node-red-dashboard/
de15f3d7f8f7:~/node_modules/node-red-dashboard$ npm install
```

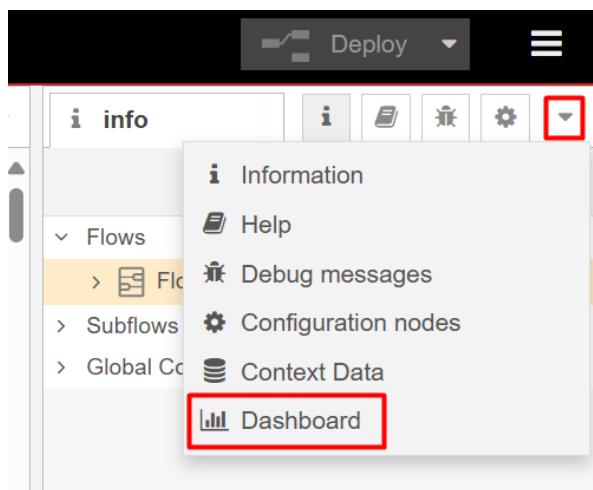
Reiniciamos Node-Red:

```
docker stop dockerNodeRed
docker start dockerNodeRed
```

```
irene@vps-13ffebe7:~$ sudo docker stop dockerNodeRed
dockerNodeRed
irene@vps-13ffebe7:~$ sudo docker start dockerNodeRed
dockerNodeRed
```

Ahora podemos acceder a Node-Red a través del puerto 1880:

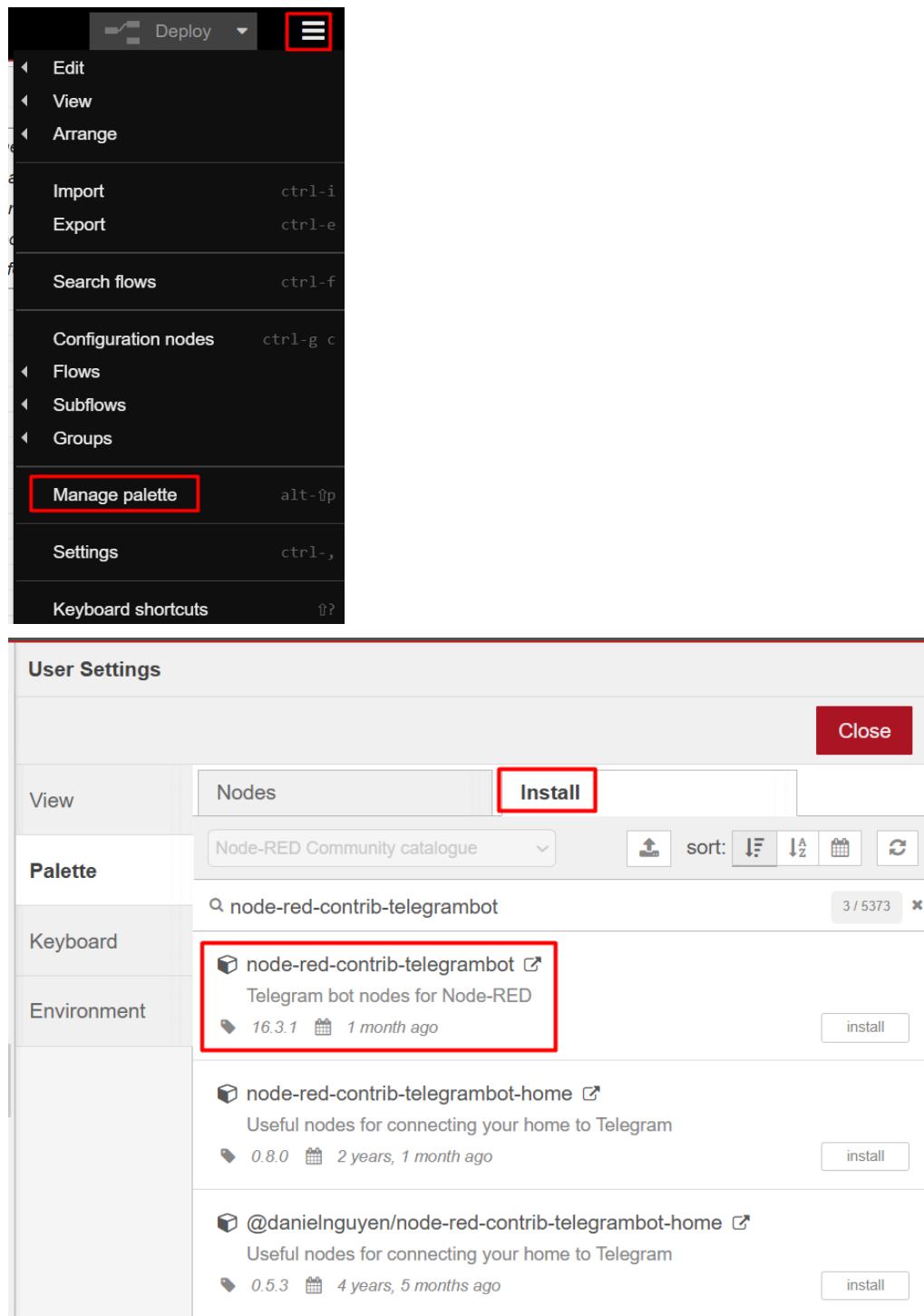
[http://numero\\_ip:1880](http://numero_ip:1880)

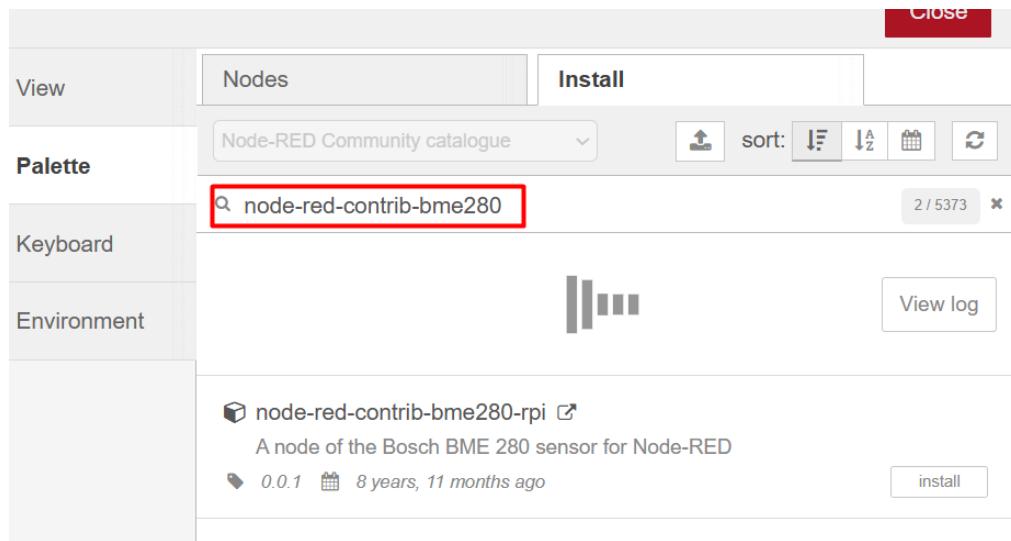


### Añadir mensajería con Telegram

En la interfaz de Node-Red, vamos al menú, después a Manage palette y en la pestaña “Install” buscamos y descargamos el módulo “node-red-contrib-telegrambot” y “node-red-contrib-bme280” para controlar el sensor.

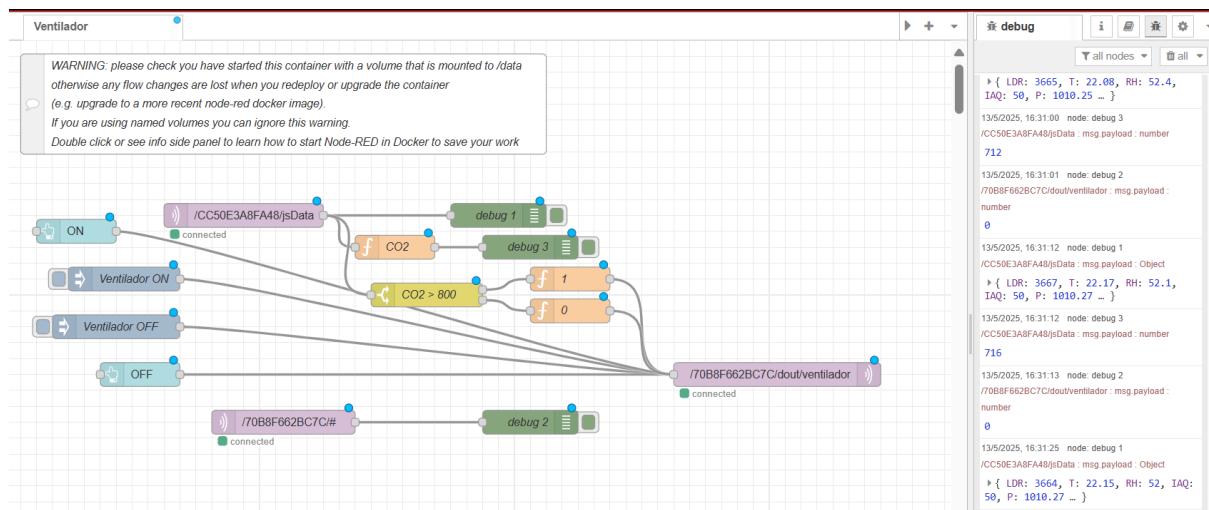
[Making \(and deploying\) an Interactive Telegram Bot in Node.js](#) -> tutorial completo de como crear el bot





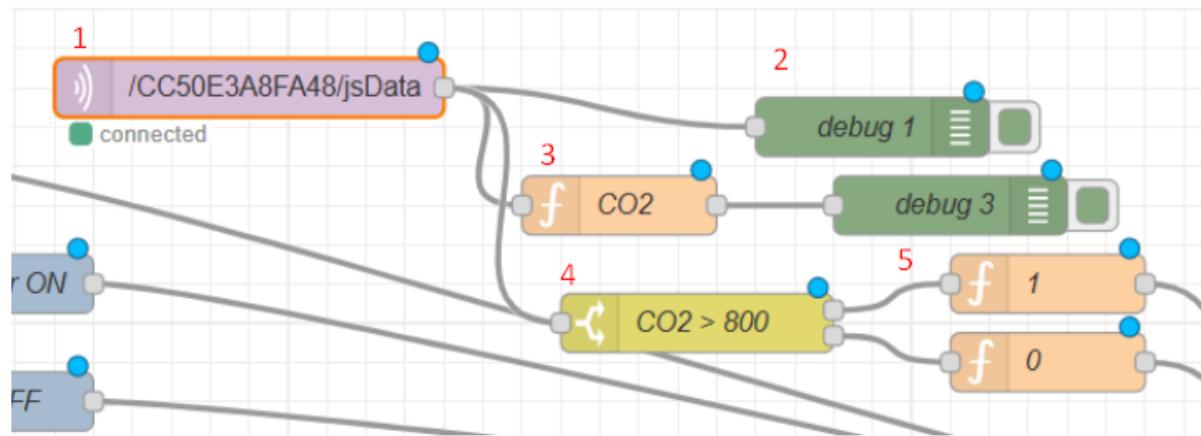
## Configuración del Node-Red para el ventilador

Control del ventilador con Node Red - Estructura:

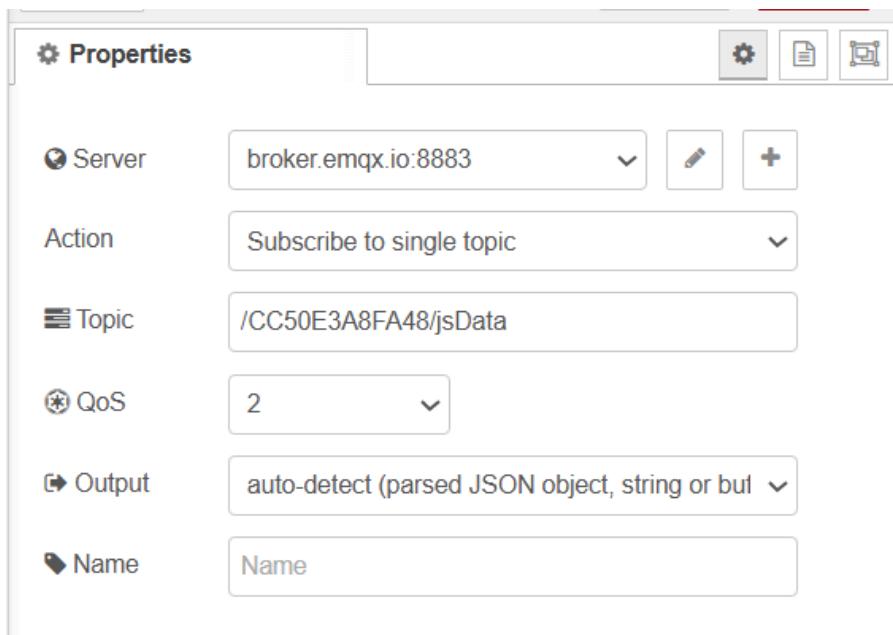


Virko:

Desde aquí comprobamos los datos de la Virko:



1. Conexión, nos suscribimos al tema /jsData



2. Debug 1 muestra toda la información de la Virko

```
13/5/2025, 16:34:38 node: debug 1
/CC50E3A8FA48/jsData : msg.payload : Object
  ▼object
    LDR: 3658
    T: 21.76
    RH: 52.4
    IAQ: 35
    P: 1010.3
    G: 297416.34
    Gs: 74
    RHs: 19
    Estat: 5
    CO2_ppm: 709
13/5/2025, 16:34:38 node: debug 3
```

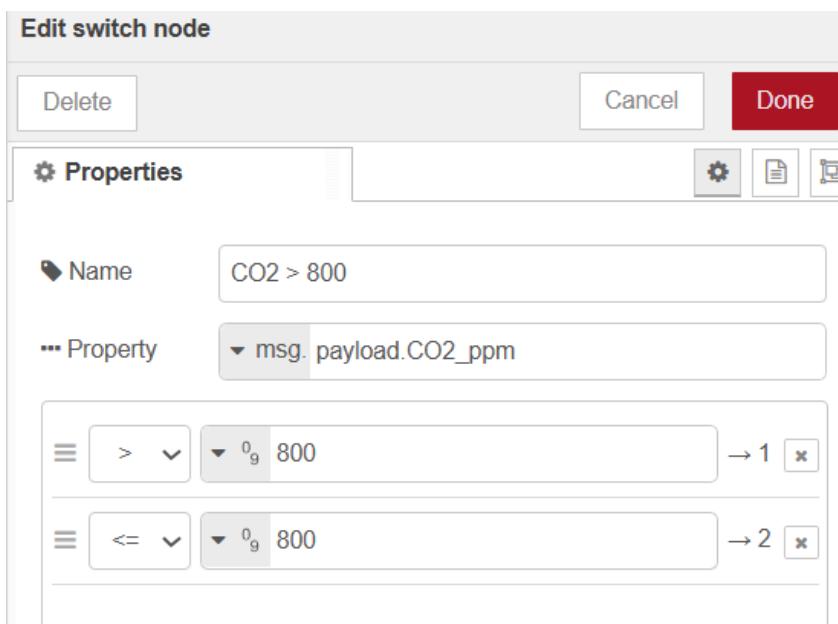
### 3. CO2, Recoge los datos de CO2



Debug 3 Muestra la cantidad de CO2

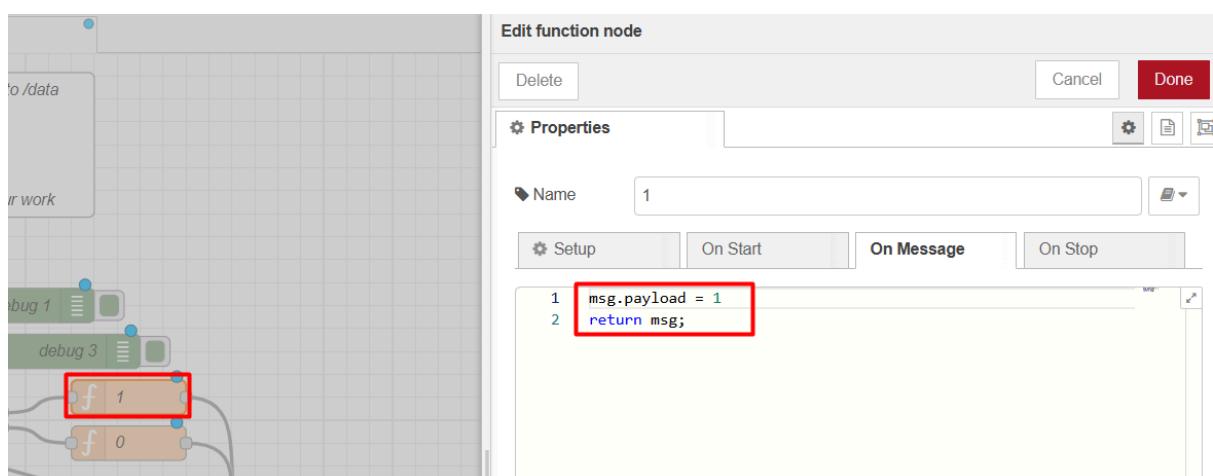
```
13/5/2025, 16:46:59 node: debug 3
/CC50E3A8FA48/jsData : msg.payload : number
701
```

4. Condicional, en el 1 se hace una acción cuando el CO2 es menor a 800, en el 2 se hace una acción si es mayor o igual a 800

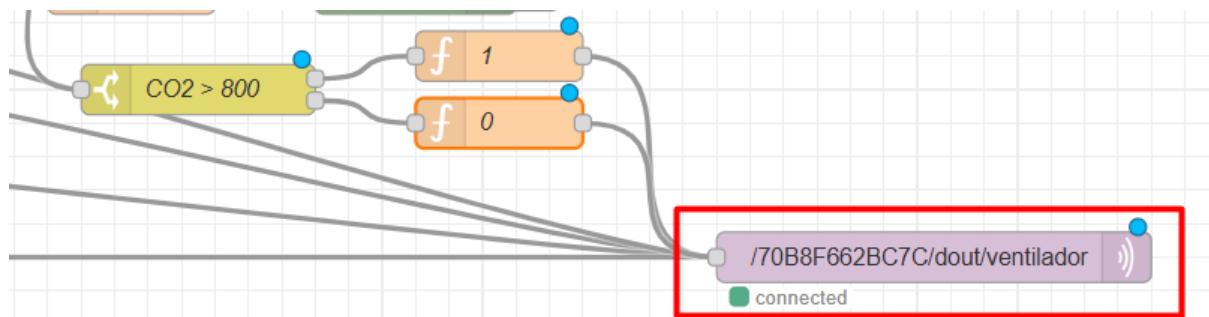


5. Acciones según las condiciones

Si hay menos de 800 se envia un 1, si hay mas o igual que 800 se envia un 0



Estos mensajes se envían al ventilador



En el código del ventilador vemos que si recibe un 1 se activa y si recibe un 0 se apaga

```

# Ventilador
if message.topic == "/70B8F662BC7C/dout/ventilador":
    if msg == '1':
        doutbit("0000", "B", 1, 1)
    else:
        doutbit("0000", "B", 1, 0)
  
```

### Creación del bot

En nuestro Telegram buscamos @botfather y en el chat ponemos estos tres mensajes:

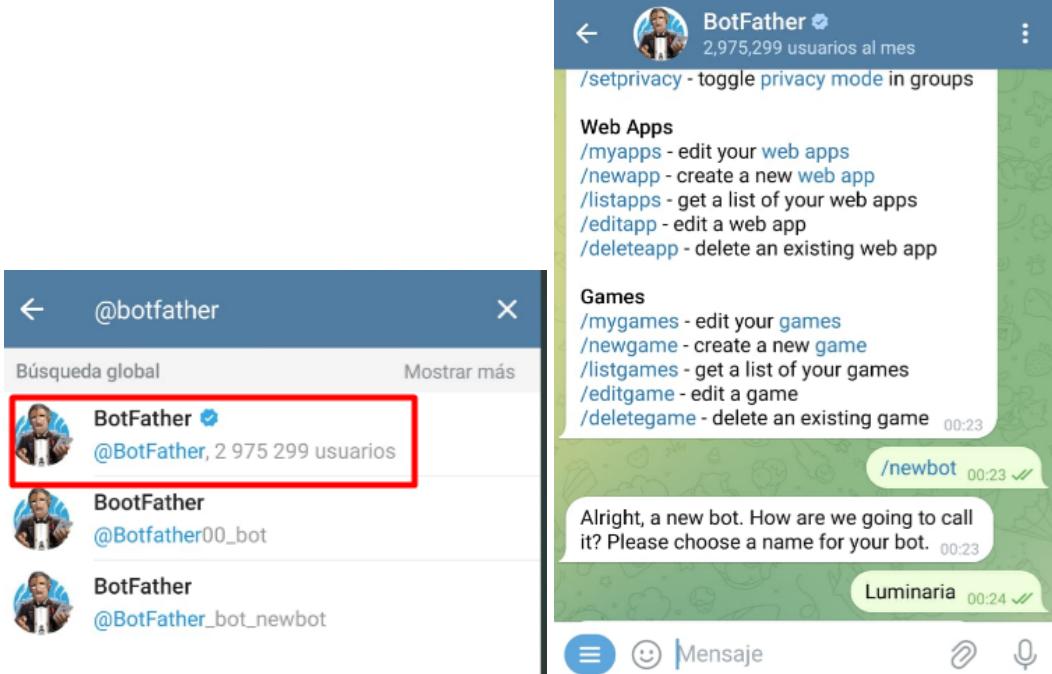
/newbot

Un nombre para el bot que es el que se mostrará cuando se le envíen mensajes:

Luminaria

y un nombre de usuario para el bot (importante que termine en \_bot)

luminaaria\_bot



The screenshot shows a Telegram chat with the BotFather (@botfather). The user has chosen the username 'BotFather' for their new bot, which is highlighted with a red box. The BotFather responds by asking for a name for the bot, and the user replies 'Luminaria'. The BotFather confirms the creation of the bot and provides its token and usage instructions.

```

@botfather
Búsqueda global Mostrar más
BotFather ✅
@BotFather, 2 975 299 usuarios
BootFather
@Botfather00_bot
BotFather
@BotFather_bot_newbot

BotFather ✅
2,975,299 usuarios al mes
Sorry, this username is already taken.
Please try something different. 00:24
lumi_bot 00:24 ✓
Sorry, this username is already taken.
Please try something different. 00:24
lumin_bot 00:24 ✓
Sorry, this username is already taken.
Please try something different. 00:24
luminaaria_bot 00:25 ✓

Done! Congratulations on your new bot. You
will find it at t.me/luminaaria\_bot. You can
now add a description, about section and
profile picture for your bot, see /help for a
list of commands. By the way, when you've
finished creating your cool bot, ping our Bot
Support if you want a better username for it.
Just make sure the bot is fully operational
before you do this.

Use this token to access the HTTP API:
7470485169:AAG88xRn2RvfQYxjSk_PqUQBSi0UFIG7Mks
Keep your token secure and store it safely,
it can be used by anyone to control your
bot.

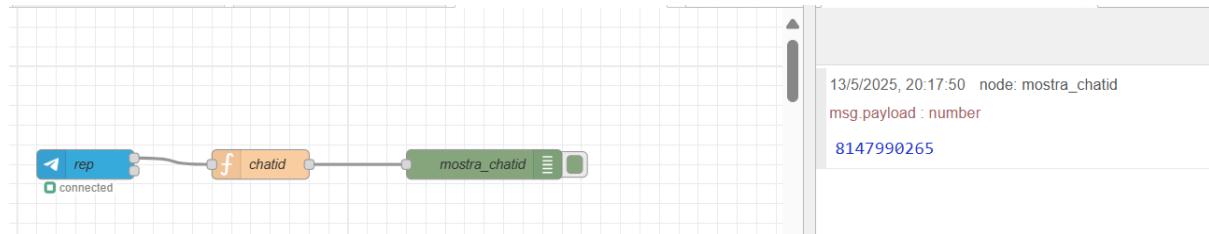
For a description of the Bot API, see this
page: https://core.telegram.org/bots/api

```

El usuario botfather enviará un mensaje con un token que debemos guardar para más adelante. Una vez guardado entramos al enlace que se encuentra en el mismo mensaje para acceder al chat de nuestro nuevo bot.

### **Configuración del bot**

Para saber el id del chat, usamos este flow que cuando un usuario envía un mensaje se ve el número asignado a esa conversación:



rep:

Bot	luminaaria_bot	<input type="button" value="▼"/>	<input type="button" value="✎"/>	<input type="button" value="+"/>
Name	rep			
Download Directory	Download directory			
Filter	<input type="checkbox"/> commands (from configured command nodes)			

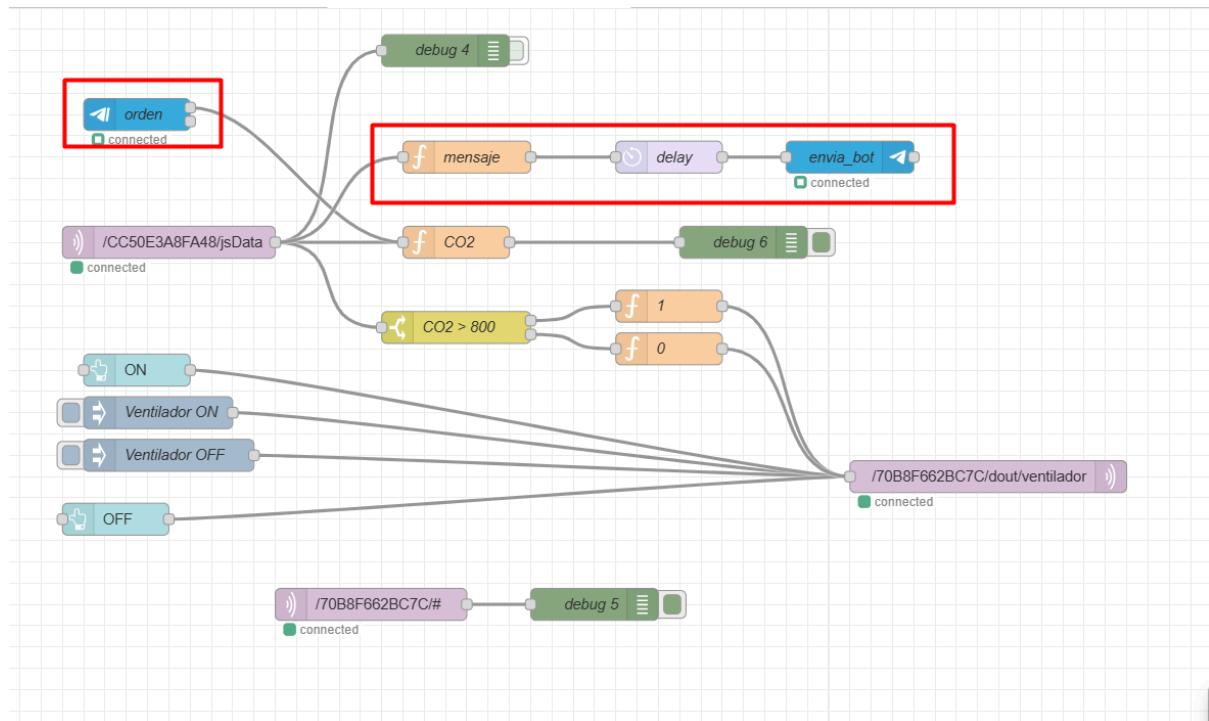
chatid:

Name	chatid	<input type="button" value="▼"/>	
<input type="radio"/> Setup	<input type="radio"/> On Start	<input checked="" type="radio"/> On Message	<input type="radio"/> On Stop
<pre>1 msg.payload = msg.payload.chatId; 2 return msg;</pre>			

mostra\_chatid:

Properties		<input type="button" value="⚙"/>	<input type="button" value="📄"/>	<input type="button" value="📝"/>
Output	<input type="button" value="▼"/> msg.payload			
To	<input checked="" type="checkbox"/> debug window <input type="checkbox"/> system console <input type="checkbox"/> node status (32 characters)			
Name	mostra_chatid			

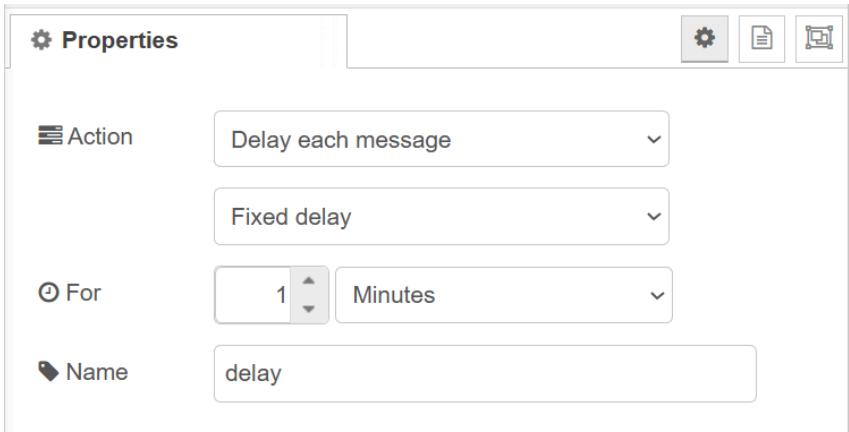
En el node-red del CO2 añadimos esta parte:



Orden: Esto permite que cuando se escriba /dades se empiecen a mandar los datos de CO2

<input checked="" type="checkbox"/> Bot	luminaaria_bot	<input type="button" value=""/>	<input type="button" value=""/>
<input checked="" type="checkbox"/> Name	orden		
<input checked="" type="checkbox"/> Command	/dades		
<input checked="" type="checkbox"/> Register at	<input type="checkbox"/> telegram server		

Delay: hace que cada mensaje se envíe cada 1 minuto



**Properties**

Action	Delay each message
Fixed delay	
For	1 Minutes
Name	delay

Mensaje: Establece el formato en el que se enviará la respuesta: CO2: xxx

◆ Name

Setup  On Start  **On Message**

```

1  var co2 = msg.payload.CO2_ppm;
2  msg.payload = {
3      "content": "CO2: " + co2,
4      "chatId":
5          8147990265,
6      "type": "message"
7  }
8  return msg;

```

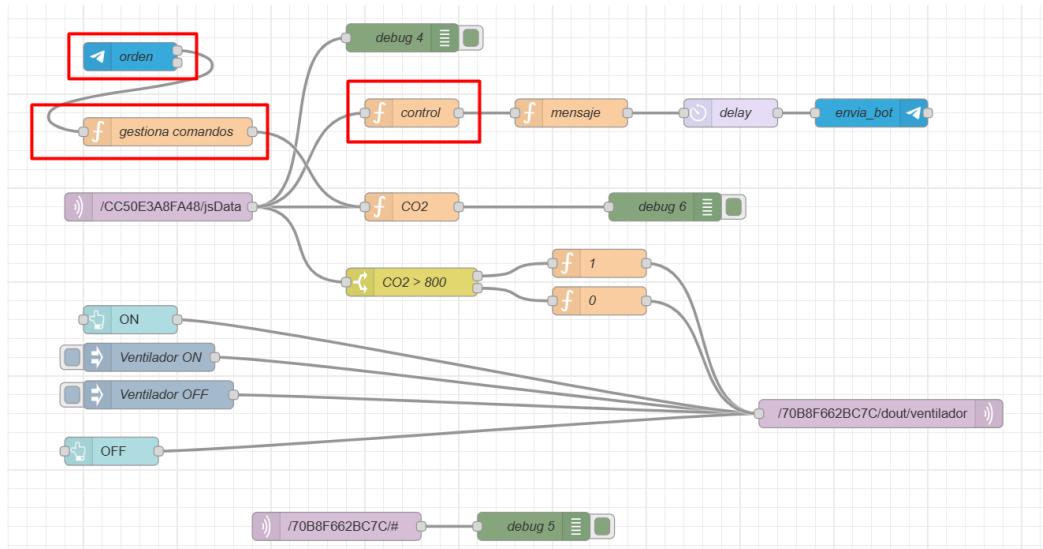
Envia\_bot: indica a que bot hay que mandar la respuesta

**Properties**   

Bot	<input type="text" value="luminaaria_bot"/> <span style="float: right;">▼  </span>
Name	<input type="text" value="envia_bot"/>
<input checked="" type="checkbox"/> Send errors <input type="checkbox"/> to second output	

### Añadido para dejar de enviar datos

En el nodo anterior cuando el usuario envía /dades empieza a recibir los datos de CO2 constantemente, para poder pararlo hemos añadido lo siguiente:



Orden:

Properties	
Bot	luminaaria_bot
Name	orden
Download Directory	Download directory
Filter	<input type="checkbox"/> commands (from configured command nodes)

Gestiona comandos, comprueba si el mensaje enviado al bot es /dades o /stop:

```

1 let texto = msg.payload.content;
2
3 if (texto === "/dades") {
4   global.set("enviar_dades", true);
5   node.warn("Activado envío de datos");
6 } else if (texto === "/stop") {
7   global.set("enviar_dades", false);
8   node.warn("Desactivado envío de datos");
9 }
10 // No pasa el mensaje al flujo principal
11 return null;
12
13
  
```

Control, si el mensaje es /dades, envía la información, si no lo es no devuelve nada:

```

1  if (global.get("enviar_dades") === true) {
2    return msg;
3  }
4  return null;
5

```

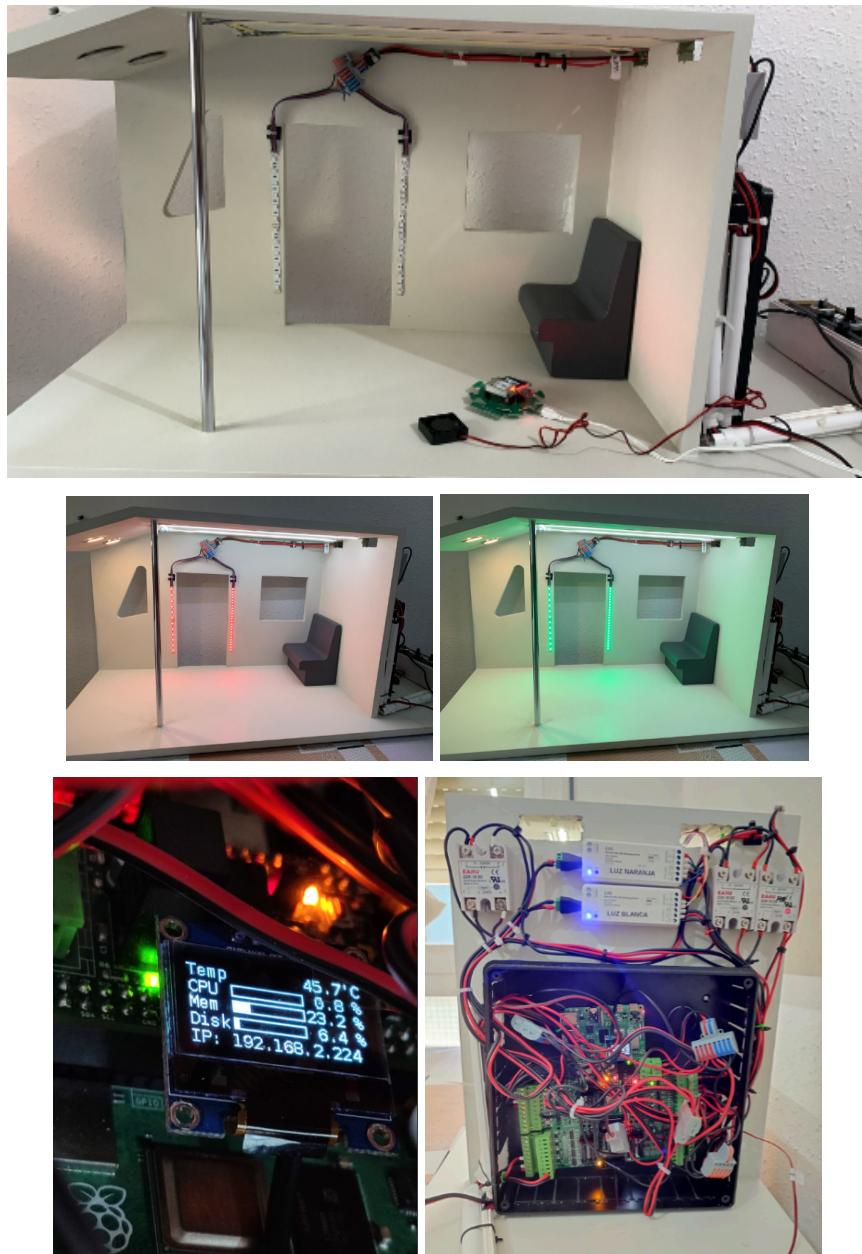
Resultado:



## Resultados de programa

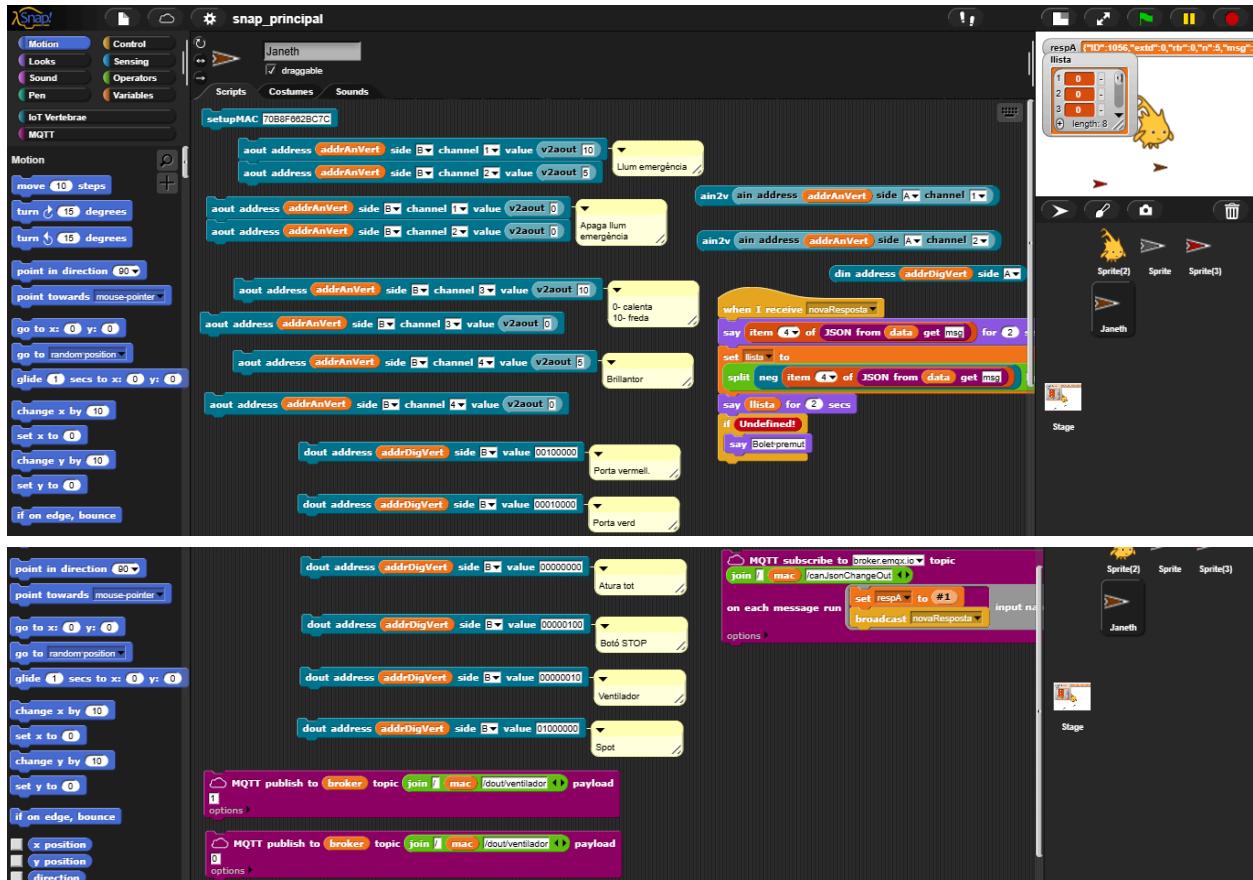
### Imágenes del estado de ejecución de programa

Ejecución del programa de luces.



## Snap

Control de luces de puerta, techo, luz del spotlight, potenciómetros, selector, champiñón, botón stop y ventilador.



## Python

```
repte@raspberrypi:~/codis/python $ python dout00.py
repte@raspberrypi:~/codis/python $
```



Ejecución de las luces de la puerta.

Archivo [test02.py](#) implementación de códigos para la ejecución en el [paho03.py](#).

Imagen código test02.py

```
repte@raspberrypi:~/codis/python $ cat test02.py
from i2c_iotv import *
import time
import signal
import threading

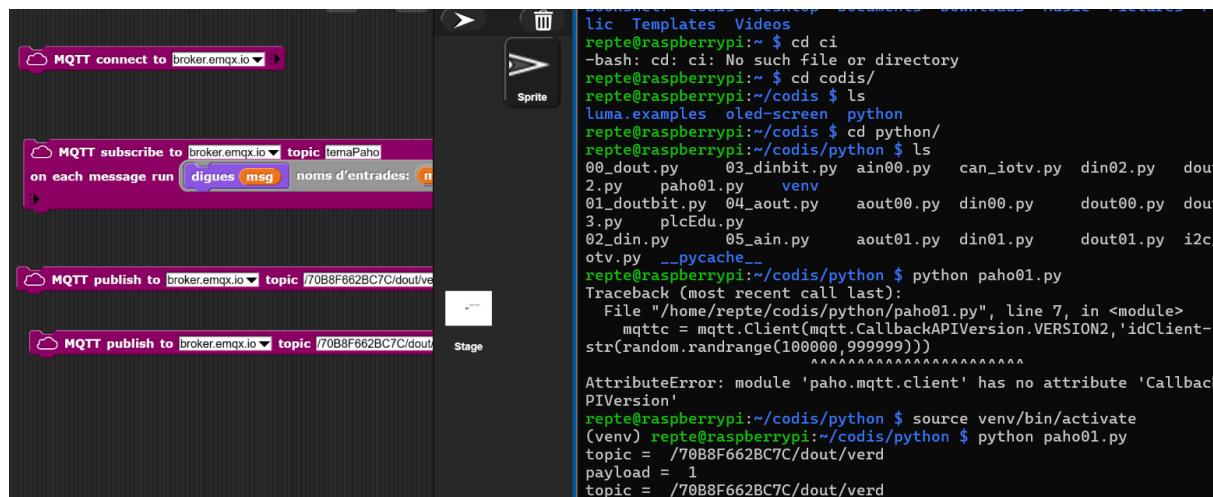
luz_naranja_lock = threading.Lock()

puertas_abriendo_lock = threading.Lock()
puertas_abriendo = False
emergencia = False

def v2aout(voltage0_10):
    retV = round((voltage0_10*4095)/10)
    if retV < 0:
        retV = 0
    if retV > 4095:
        retV = 4095
```

## Paho

Desde el Snap si ejecutamos el publish y en la raspberry tenemos funcionando el paho01.py, decimos que se enciendan las luces verdes con 1 y que se apaguen con 0.



## **FASE D - Testeo y aplicación de correcciones del programa**

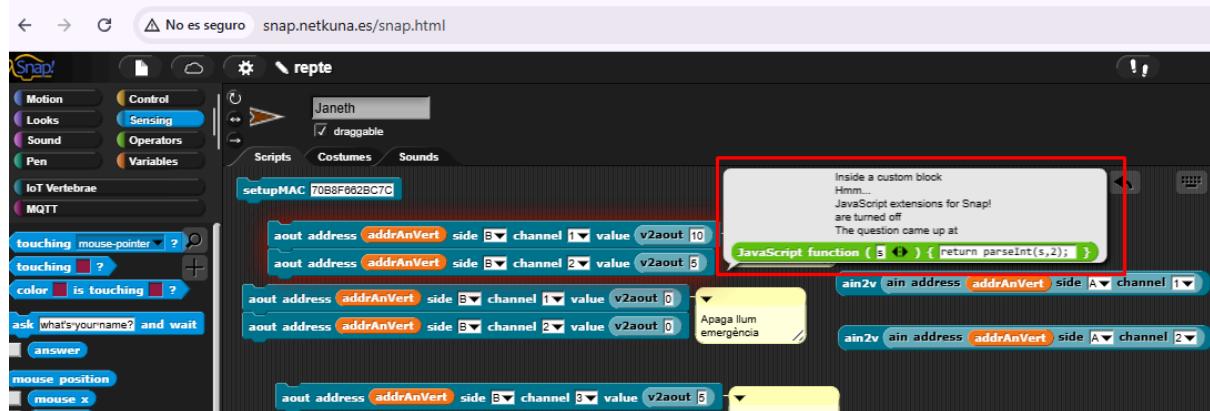
### **Monitorización y implementación de correcciones**

Incidencias en el proceso de desarrollo de los códigos:

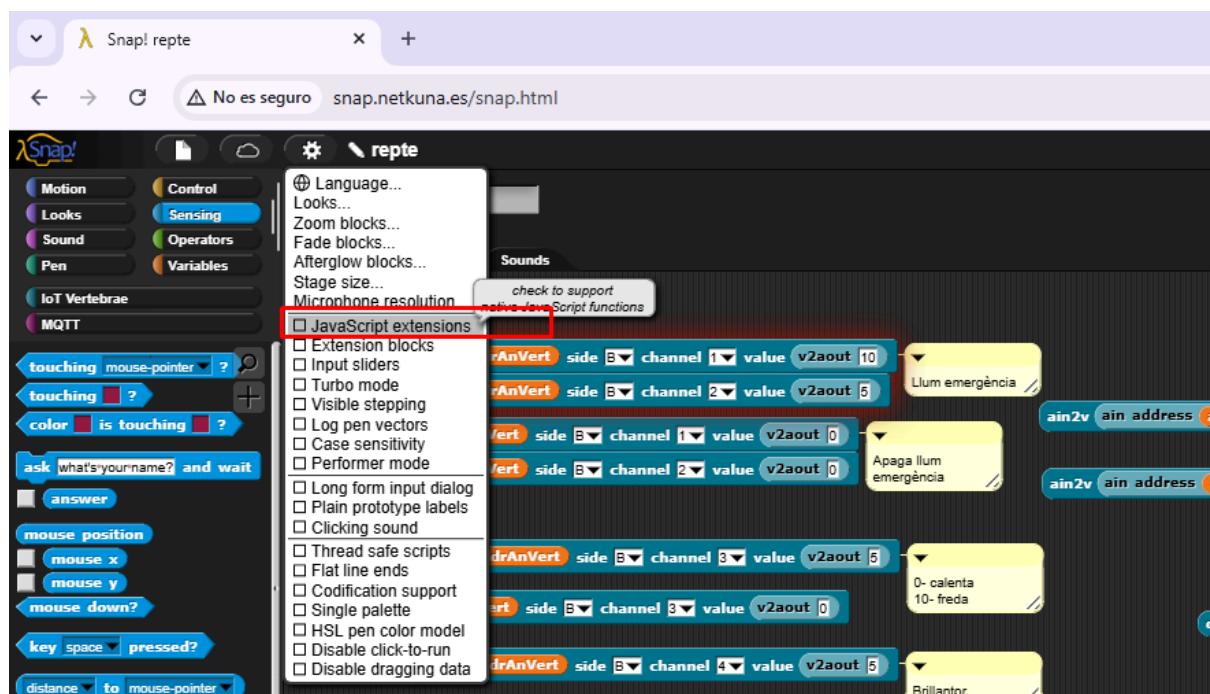
Tipo de salida/entrada	Nombre de archivo	Descripción de incidencia	Solución
Salida-Digital	dout00.py	while true - Al usar los while los códigos no terminaban bien lo que generaba errores en las siguientes ejecuciones .	Como solución se apagó la raspberry y como solución definitiva el código se cambió añadiendo un signal.pause() que lo que hace es que el programa solo se ejecute si se recibe una señal.
Entrada-Digital	din02.py	while true - Al usar los while los códigos no terminaban bien lo que generaba errores en las siguientes ejecuciones .	Como solución se apagó la raspberry y como solución definitiva el código se cambió añadiendo un signal.pause() que lo que hace es que el programa solo se ejecute si se recibe una señal.
--	paho01	from din00 import boto  En la función botón había un while que generaba un bucle infinito e impedía que se ejecutara el resto del código	Como solución hemos quitado el while de din00.py y generado una alternativa en din02.py

## Incidència Snap:

No nos podemos conectar y nos sale el siguiente cuadro de incidencia.



En este caso se debe hacer lo siguiente para conectar el Snap al código fuente de la caja electrónica de mandos de luces, selector, etc.



Incidència VPN:

En caso de que no se acceda al [snap.netkuna.com](http://snap.netkuna.com) como primer paso se debe verificar el estado del docker como se indica abajo.

```
janeth@vas-13ffebc7:/var/www/html/Snap$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS
 NAMES
1cf52b49fid   apache:7.4    "docker-php-entrypoi..."  16 hours ago  Up 16 hours  0.0.0.0:8080->80/tcp, :::8080->80/tcp
apache
29celid7997af  isnap         "docker-php-entrypoi..."  20 hours ago  Up 20 hours  80/tcp
  snap
434649153c79  ipreus:1.0    "docker-php-entrypoi..."  2 weeks ago   Exited (0) 10 days ago
determined_vohard
33a45eba7f60  ipreus:1.0    "docker-php-entrypoi..."  2 weeks ago   Exited (0) 10 days ago
  ipreus
761372fa38e8  sm9-bd-phplatest "docker-php-entrypoi..."  6 months ago  Exited (0) 2 weeks ago
sm9_clot_netkuna_es
303e42fb9b5b  mysql:8.1.0    "docker-entrypoint.s..."  6 months ago  Exited (0) 2 weeks ago
sm9-mysql_db_docker
52c2c00eb05e6 jrcs/letsencrypt-nginx-proxy-companion:latest  "/bin/bash /app/entr..."  6 months ago  Exited (0) 3 days ago
letsencrypt-helper
e3e271baaa35  jwilder/nginx-proxy:latest   "/app/docker-entrypo..."  6 months ago  Up 20 hours  0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :
```

Si el estado fuera Exited se debe activar el docker.

sudo docker start 761372fa38e8 (container ID)

Verificar otra vez: docker ps -a

## BIBLIOGRAFÍA

[things.cat](#)

[IoT-Vertebrae](#)

[IoT-Vertebrae - Explicacion](#)

[Ejemplo de uso del IoT vertebrae](#)

[Paho MQTT y Python](#)

[Instalación de Node-RED](#)

[Uso de Node-RED](#)

[Crear bot de Telegram](#)

[Sistema domòtic amb Internet de les Coses](#)

[Creación de contenedores docker](#)

[Cómo configurar la red IP con la herramienta nmtui](#)