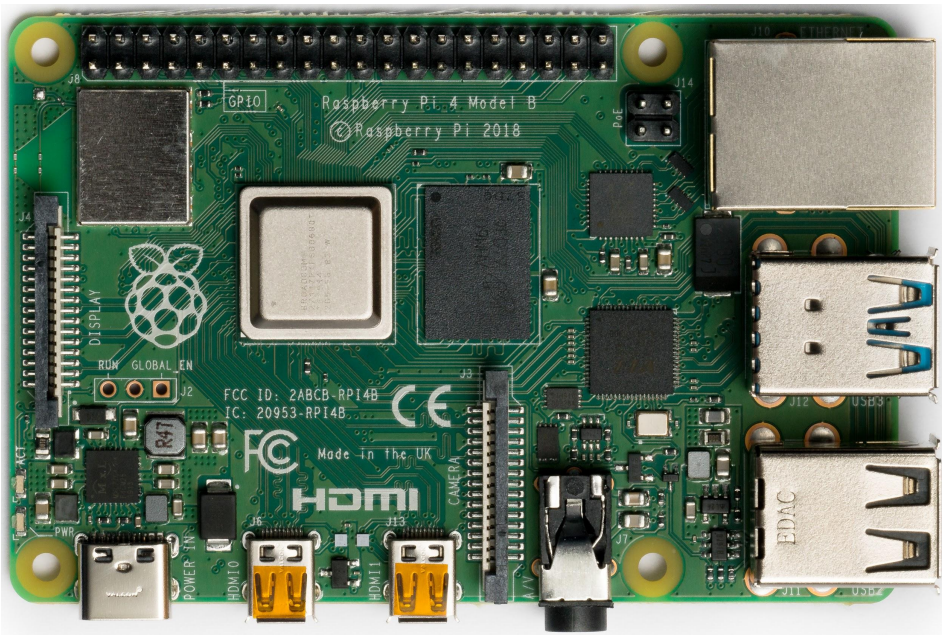
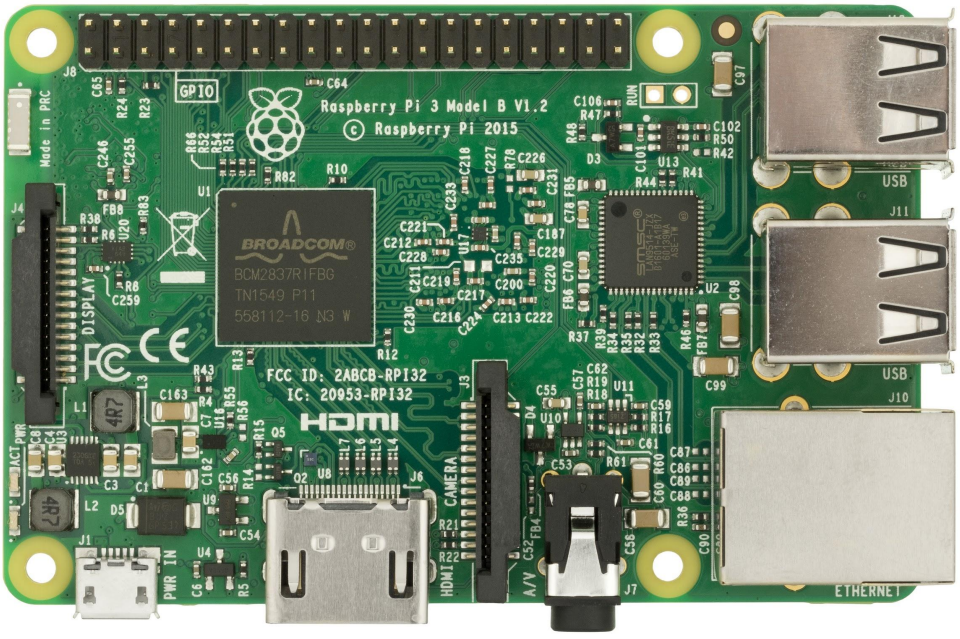
Escuela de Innovación Profesional
Berrikuntza Profesionalen Eskola
GOBIERNO DE ESPAÑA · MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
Financiado por la Unión Europea
NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia

Raspberry-PLC





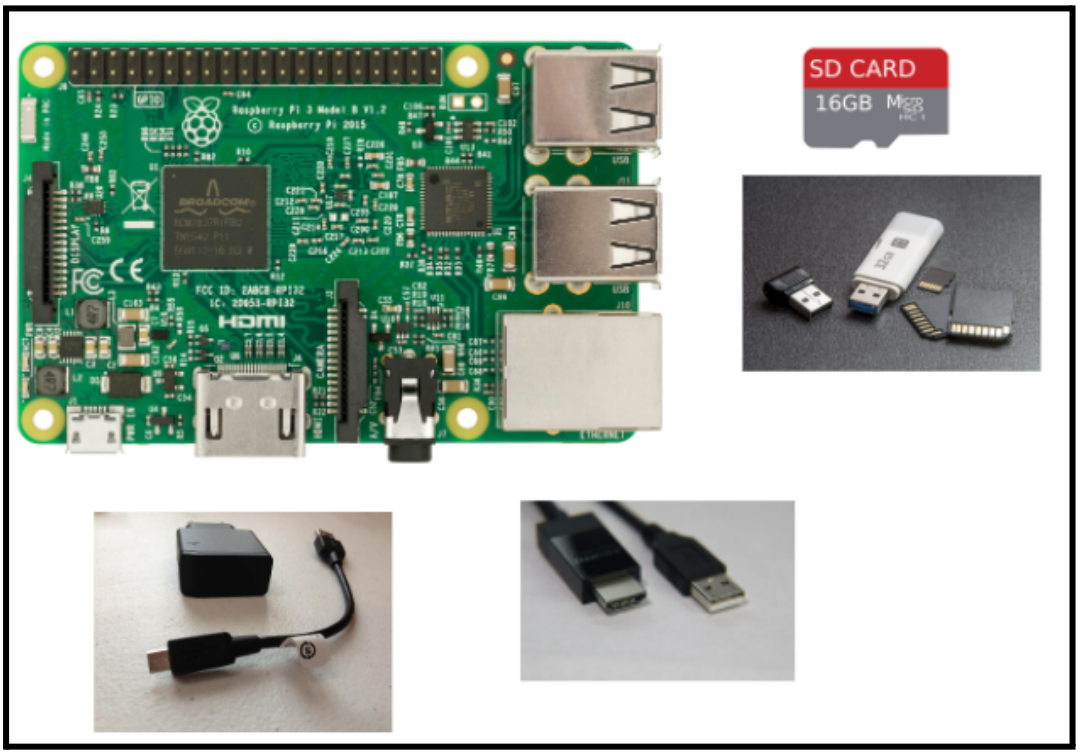
Introducción

Raspberry Pi (RPi) es un miniordenador muy utilizado por sus altas prestaciones y su reducido coste económico. Se basan en los chips ARM, microprocesadores con una arquitectura diferente a los empleados en los equipos habituales de sobremesa (Intel/AMD).

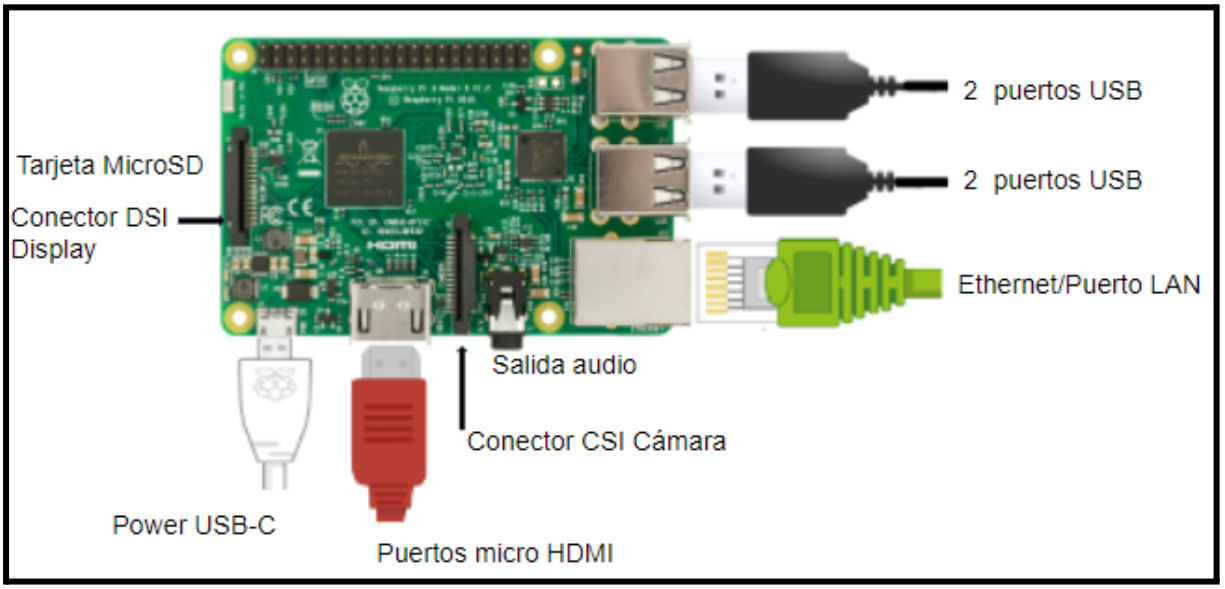
Sin embargo es compatible con sistemas operativos Linux (p.e. Raspbian) y Windows IoT, lo que permite ejecutar un gran número de aplicaciones disponibles en los sistemas tradicionales.

En el pequeño RPI encontramos, entre otros, un procesador de 4 núcleos, conexiones RAM, USB, Ethernet y HDMI y las últimas versiones están equipadas con un módulo WiFi y Bluetooth.

Componentes Raspberry Pi 3B



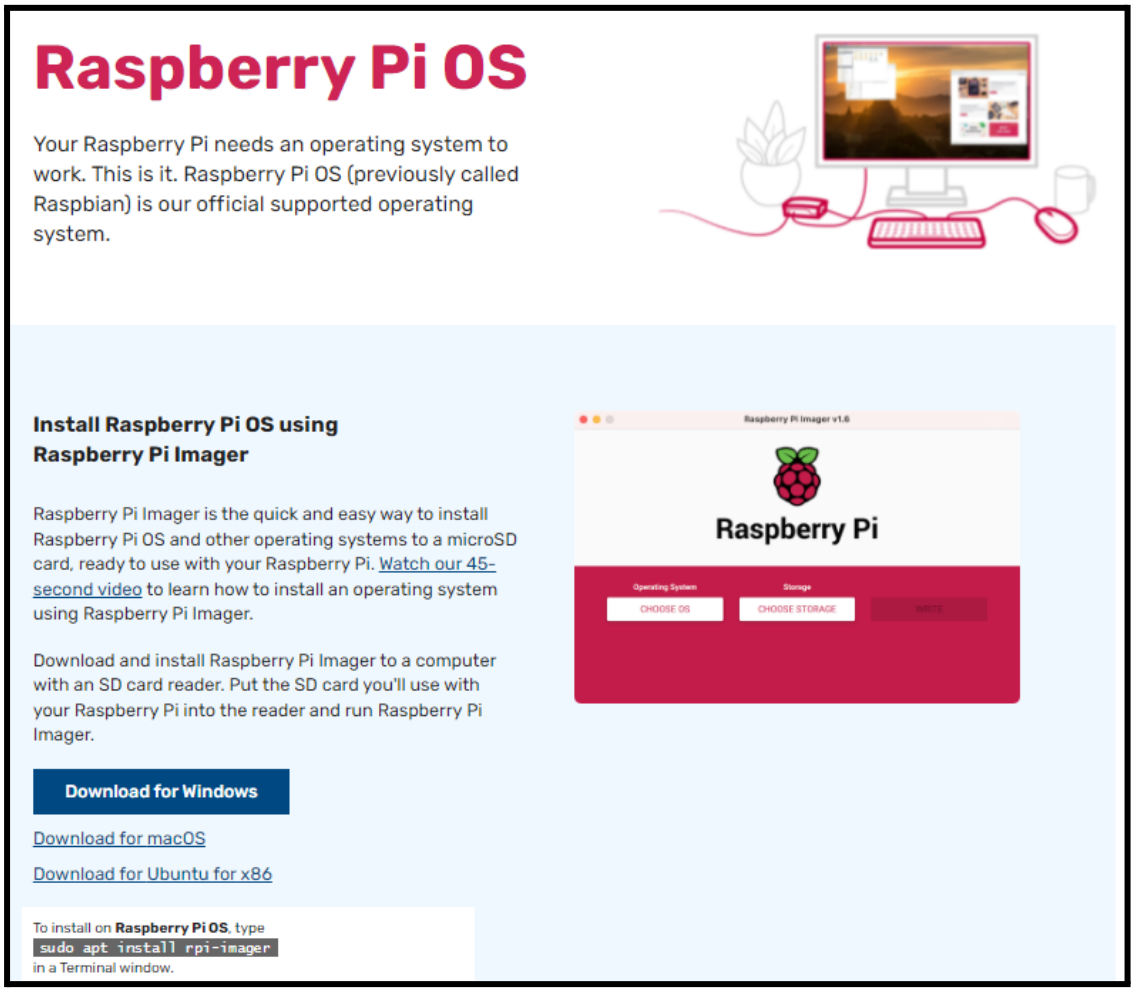
- Micro SD de al menos 4Gb y preferiblemente de 8Gb o 16 Gb.
- Adaptador o lector de SD card
- Fuente de alimentación externa tipo teléfono móvil con micro USB
- Cable HDMI



Instalación sistema operativo

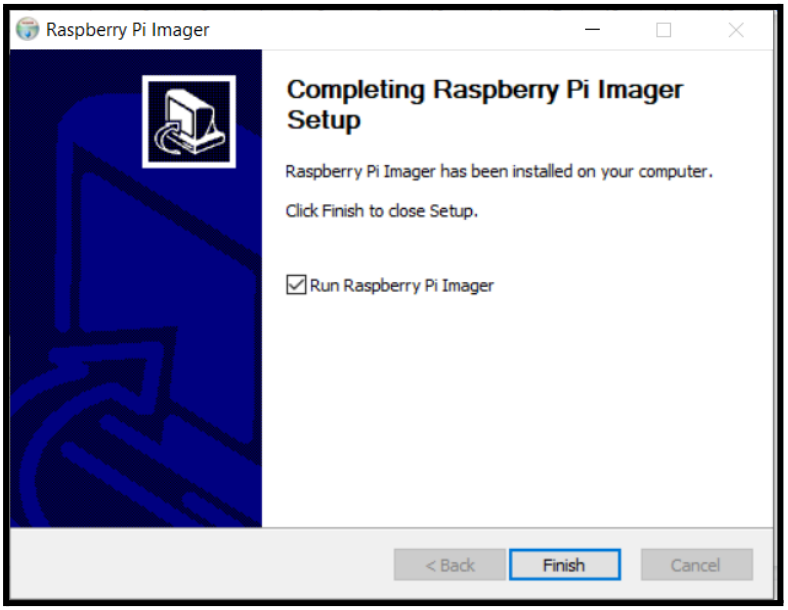
Iremos a la página oficial de Raspberry Pi y nos descargamos la imagen que necesite nuestro sistema.

- https://www.raspberrypi.org/downloads/raspbian/



Hemos realizado la descarga para Windows y tenemos el archivo imager_1.7.1.exe, pulsamos doble click para ejecutar.

Aparece la pantalla de bienvenida y pulsamos Install
Se instalan los archivos y pulsamos Finish.



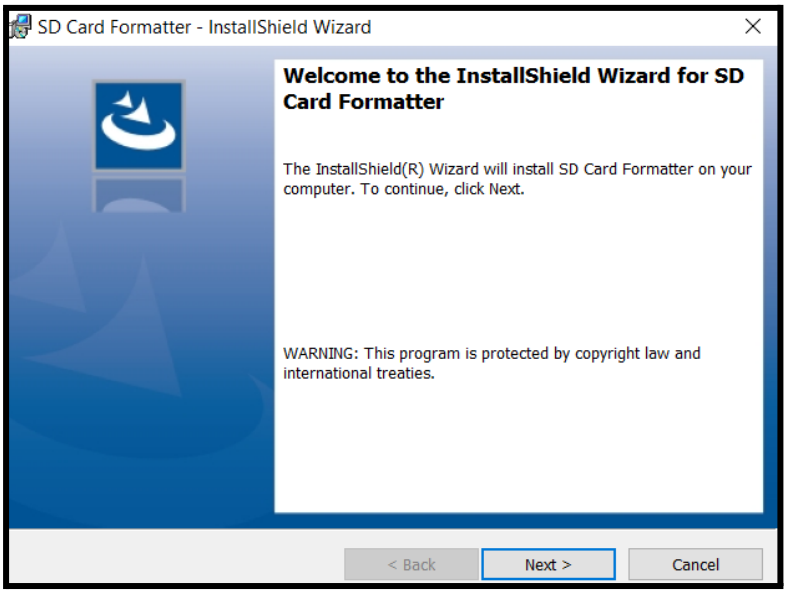
Aunque no sea estrictamente necesario, se recomienda formatear previamente la SD en Fat16 o Fat32. Para ello podemos usar el **SD formatter**, un programa que podéis descargar en:

https://www.sdcard.org/downloads/formatter_4/index.html

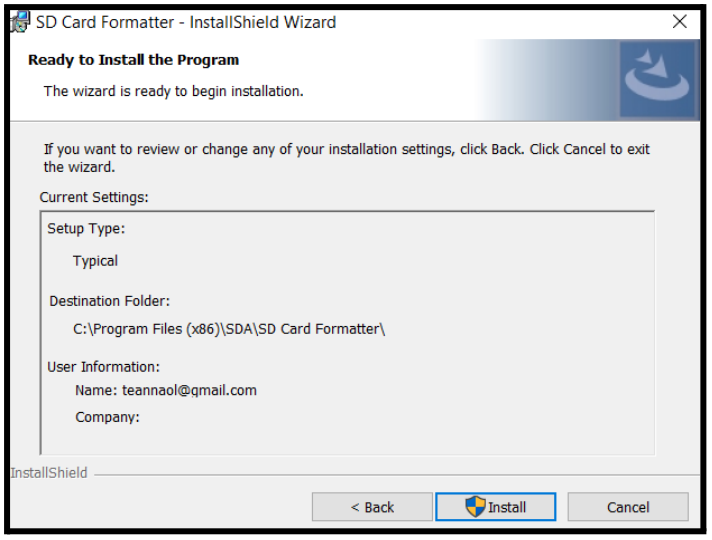
Seleccionamos el sistema y aceptamos los términos. Hemos elegido Windows y descargado un archivo zip, **SDCardFormatterv5_WinEN.zip** que descomprimimos.

Ahora ejecutaremos el archivo **SD Card Formatter 5.0.2 Setup EN.exe** que instalará el programa para formatear tarjetas SD.

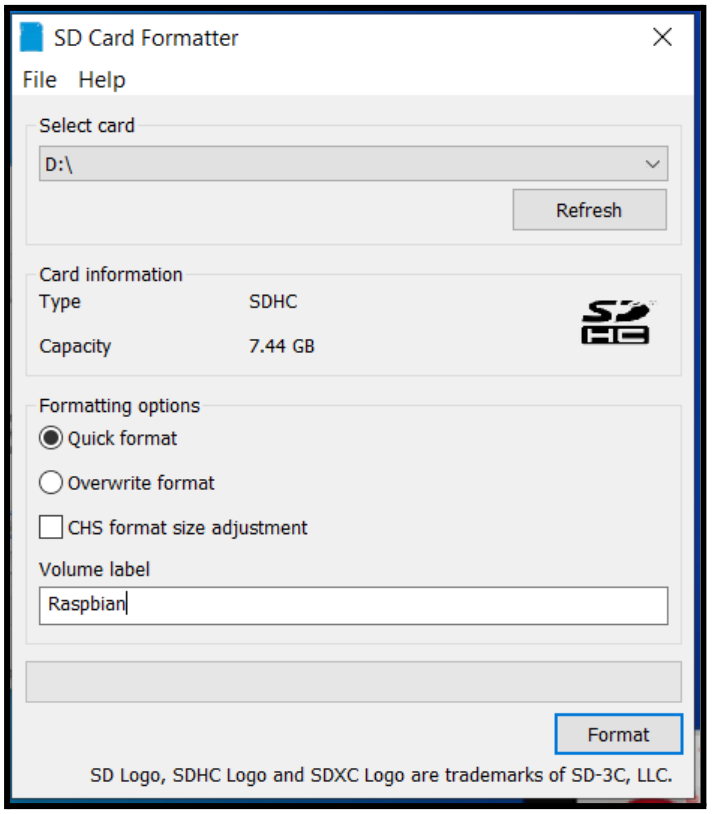
El asistente nos guiará en la instalación



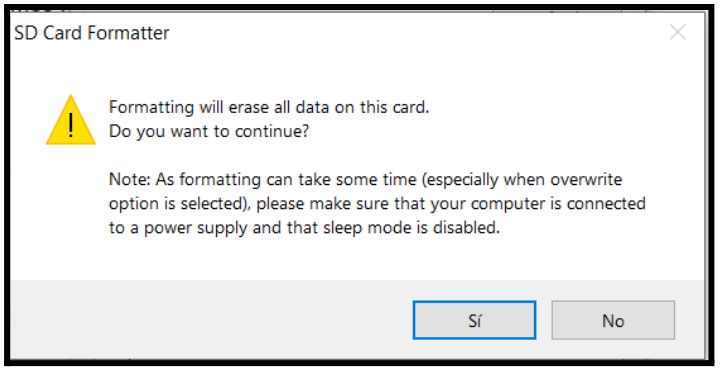
Aceptaremos la licencia y continuamos la instalación con las opciones por defecto.



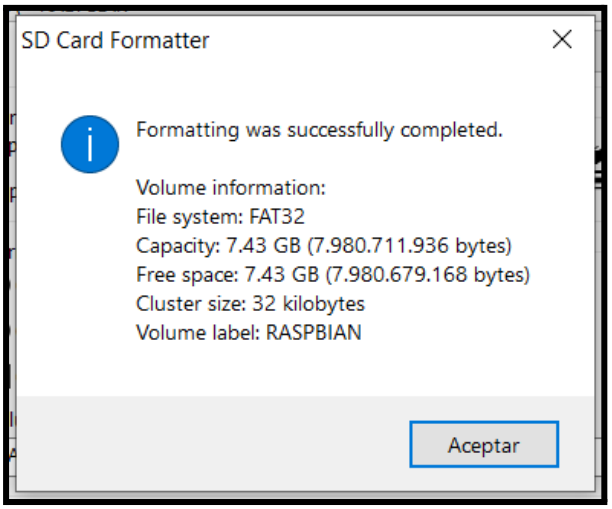
Finalizada la instalación ejecutaremos el programa y nos aparece la siguiente pantalla



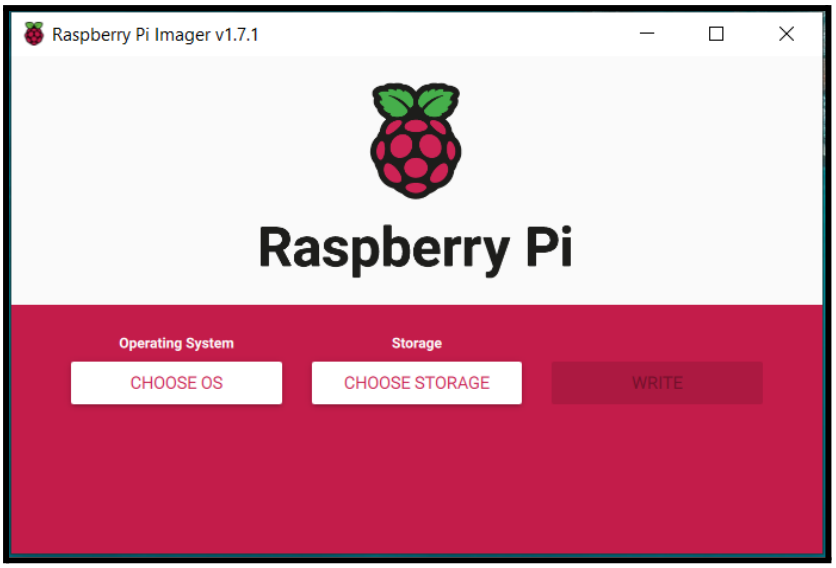
Introducir la tarjeta en el PC, nos aparecerá y podemos asignarle un nombre y elegiremos Quick format. Pulsamos Format y aparece pantalla de advertencia y pulsamos Sí.



Ya tenemos la tarjeta SD lista para copiar el sistema operativo de las RaspBerry Pi.

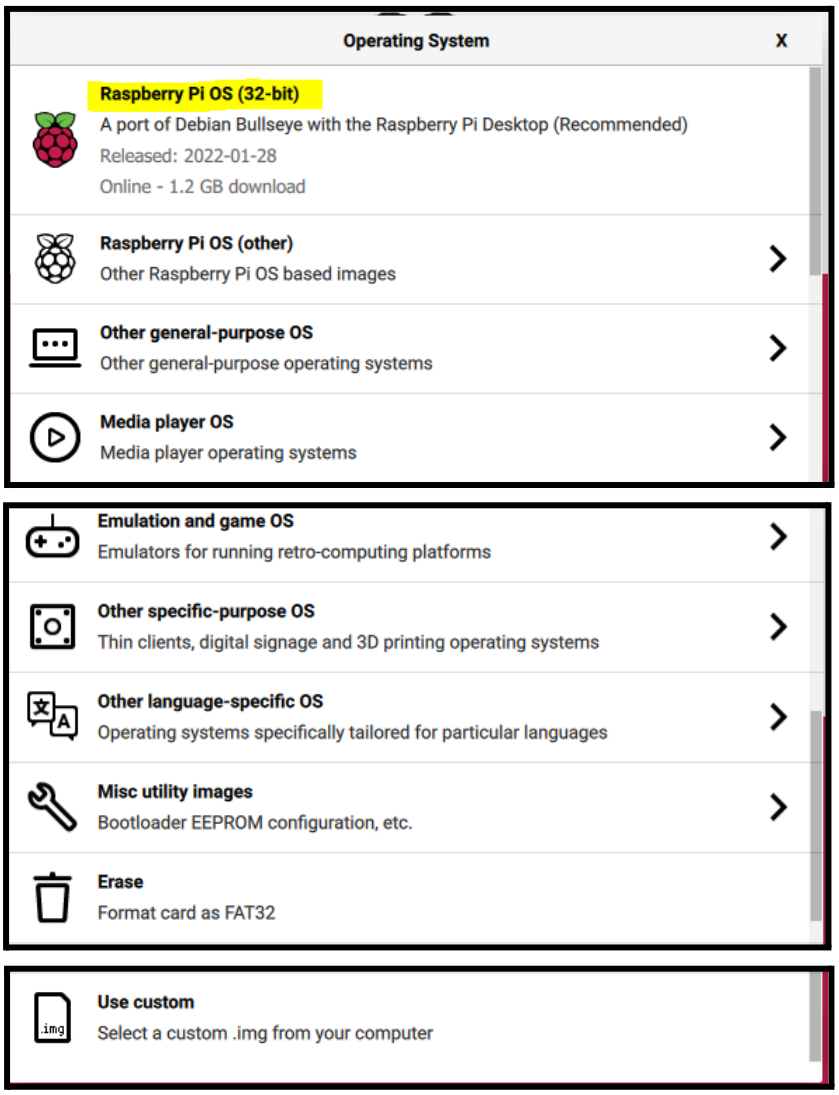


Ejecutamos el programa RaspBerry PI Imager que instalamos anteriormente.



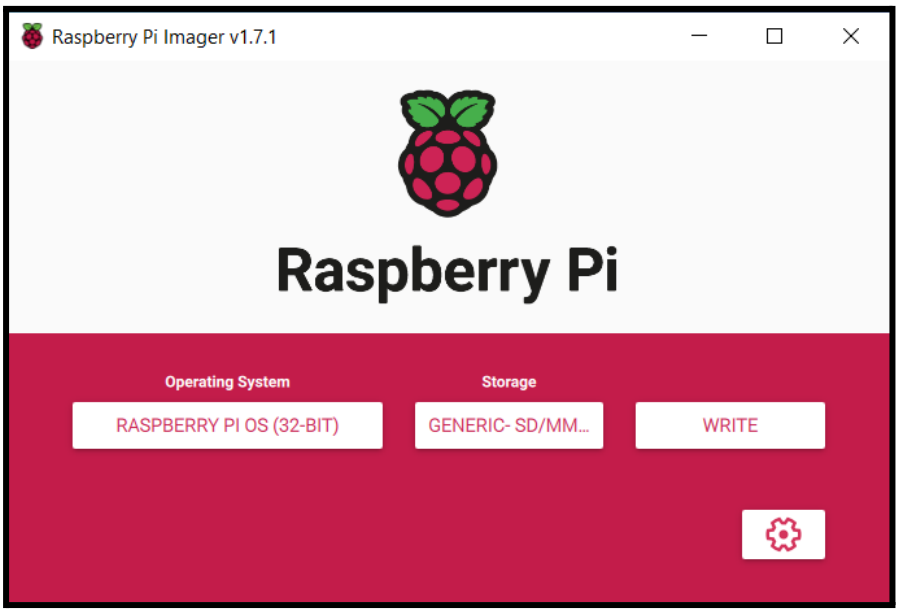
Seleccionamos, en la parte izquierda de la pantalla (Operating System), el sistema operativo con el que deseamos trabajar y, en la parte derecha, la tarjeta en la que vamos a grabar.

Existen diferentes sistemas operativos como emuladores de juegos, otras imágenes de raspberry. Seleccionamos Raspberry PI OS (32-bit) que es la opción recomendada.

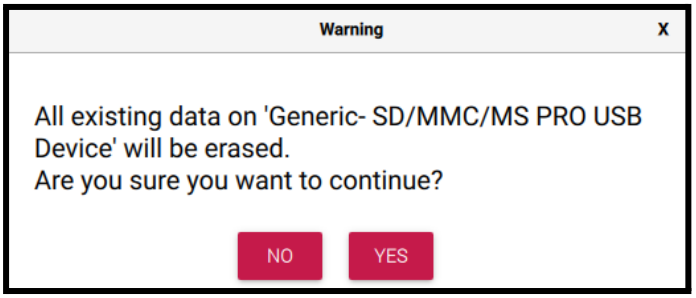


Ahora pulsamos sobre Storage y seleccionamos la tarjeta

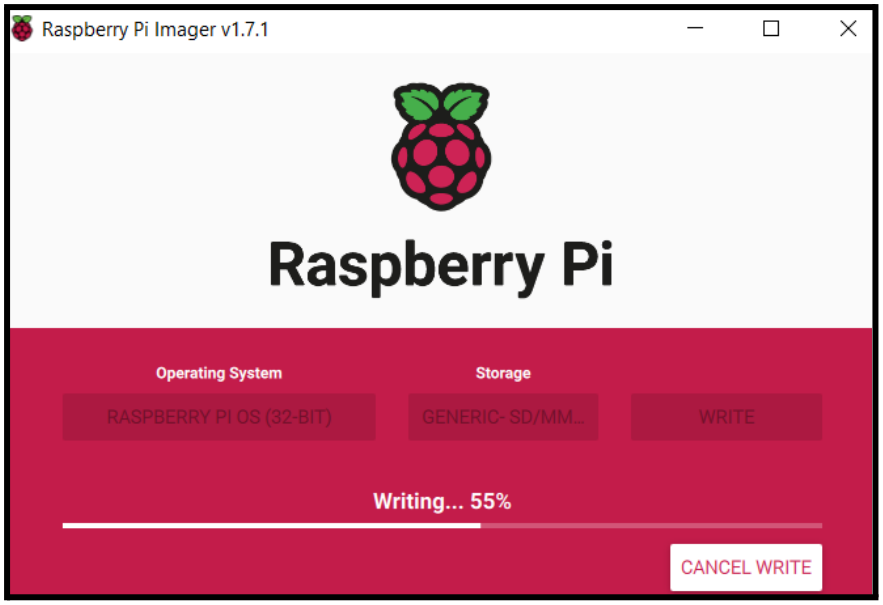
Una vez seleccionado el sistema y la tarjeta nos aparece habilitado el botón de WRITE que pulsaremos.



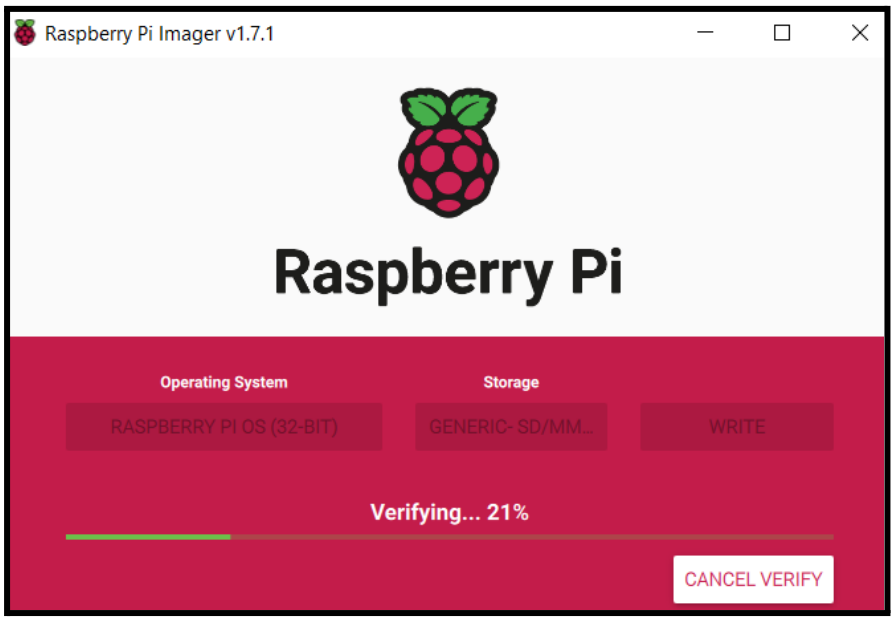
Nos aparece una pantalla de confirmación de escritura con la advertencia de que se van a eliminar todos los datos de la tarjeta.



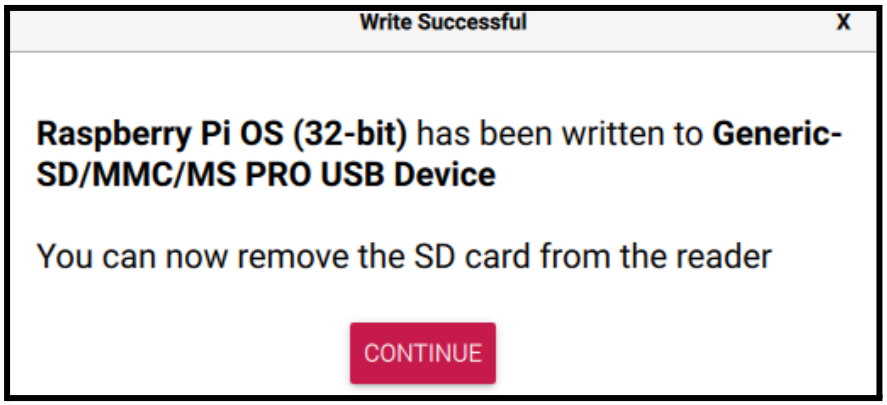
Al pulsar YES se copiaran todos los archivos del sistema operativo.



Seguido se realiza una verificación.



Al finalizar nos aparece pantalla de confirmación y ya tenemos el sistema Raspberry PI en la tarjeta SD.



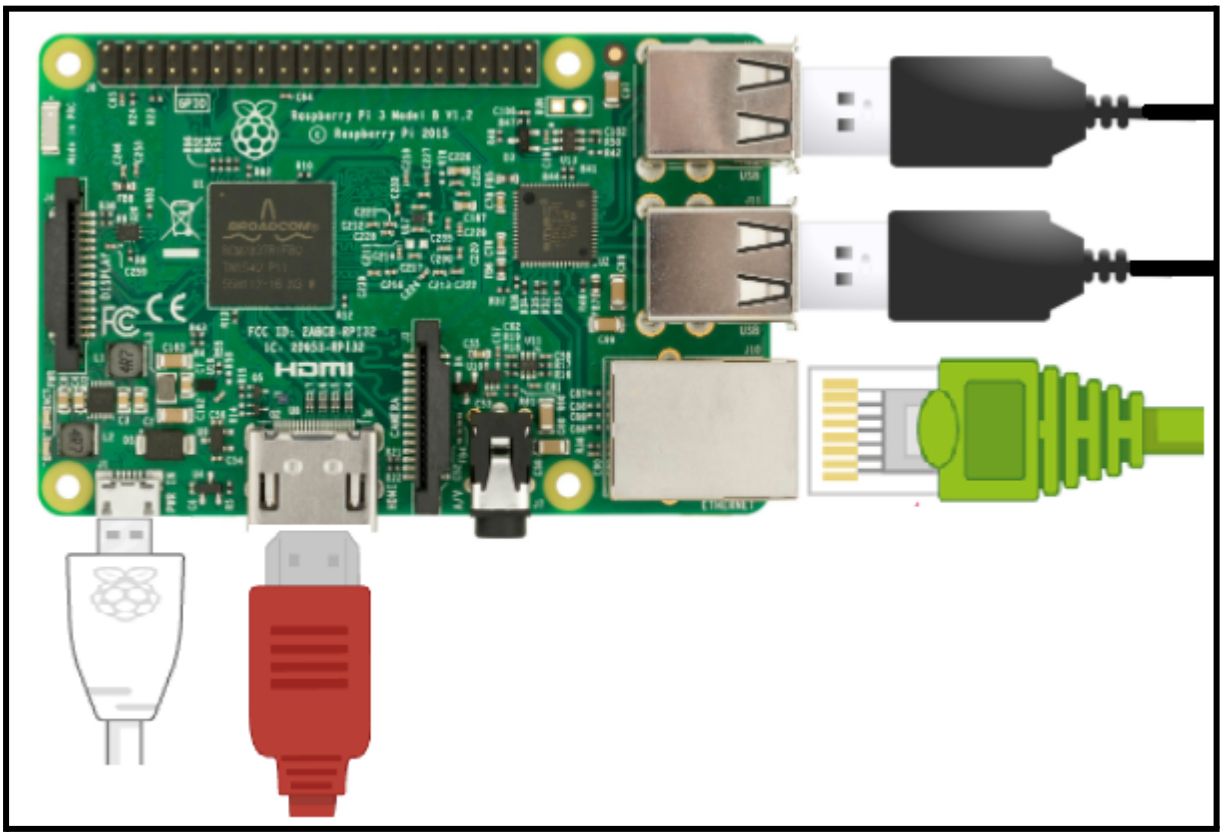
Arrancar Raspberry primera vez

Introducimos la tarjeta SD, en la que hemos grabado el sistema operativo, en la raspberry.

Conectamos a través de los puertos USB un teclado y un mouse.

Ponemos el cable HDMI y el cable ethernet para conectarnos a internet (o mediante la red WIFI.

Finalmente conectamos la fuente de alimentación.



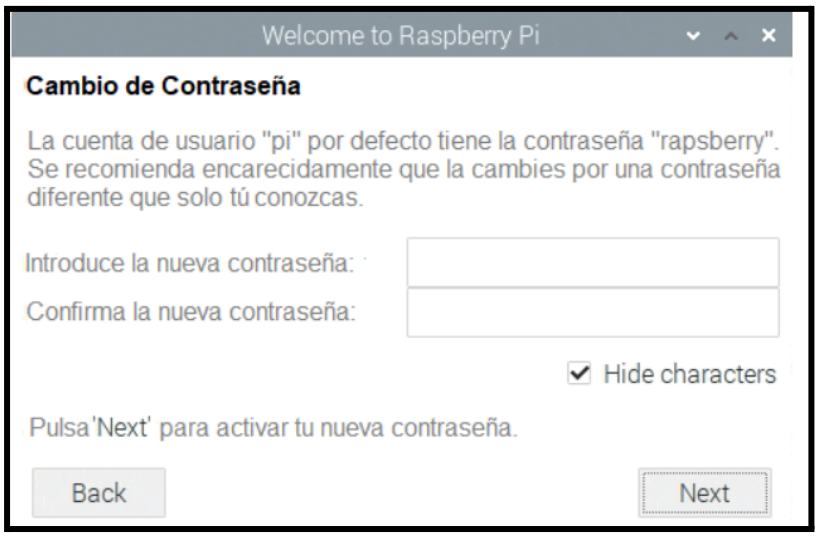
No existe interruptor o botón de encendido así que simplemente es conectar el cable de alimentación a la corriente eléctrica.

La primera vez que arrancamos la Raspberry nos llevará más tiempo y aparecerán numerosos mensajes por la pantalla.

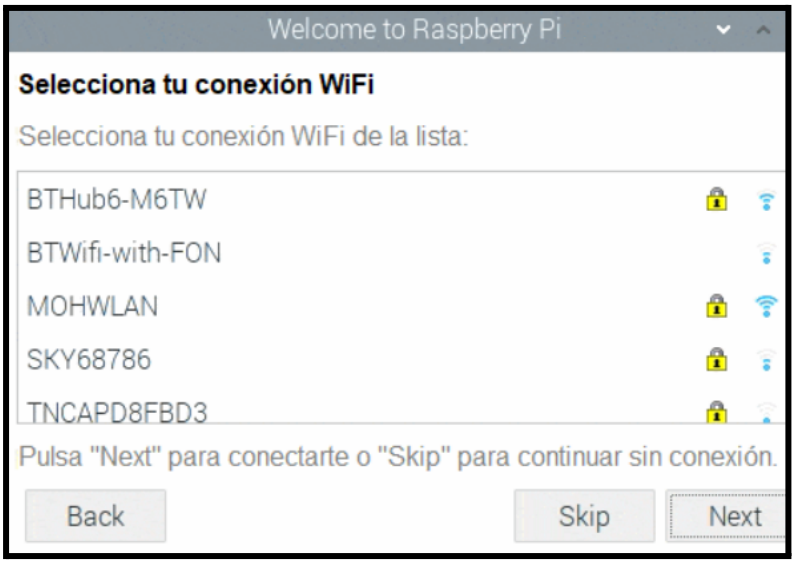
Aparece la ventana de bienvenida a Raspberry PI y si hay un servidor DHCP en nuestra red nos asignará una dirección IP que debemos anotar para posteriormente poder conectarnos por SSH o VNC.

Seleccionamos el país, idioma y zona horaria.

El usuario por defecto es pi y ponemos una nueva contraseña.



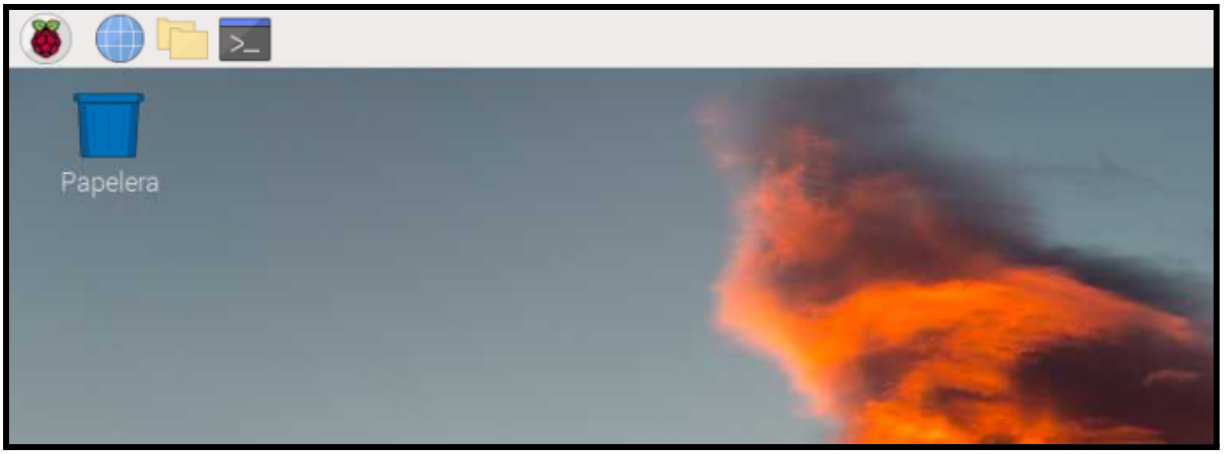
Aparece una pantalla para la conexión Wifi, si no deseamos realizar por el momento la conexión vía WIFI, pulsaremos Skip y podremos configurarla más adelante.



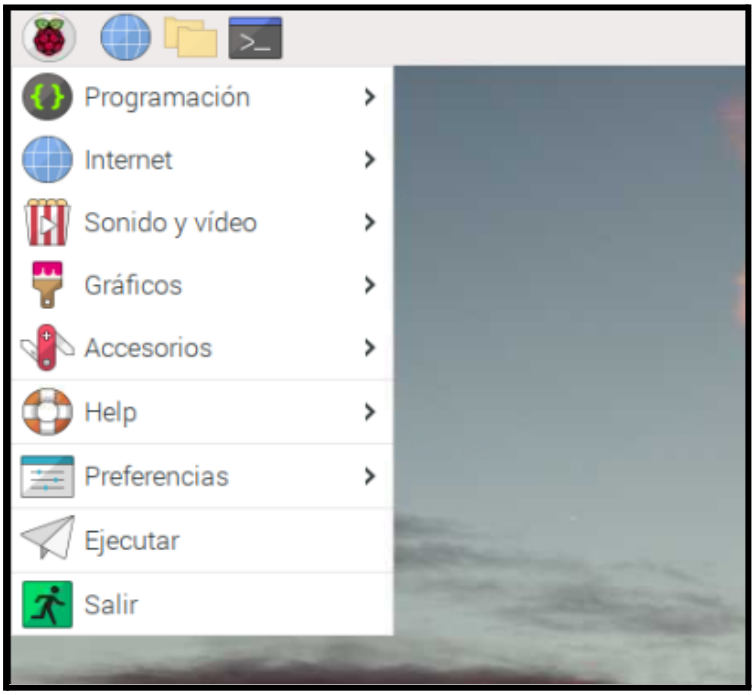
Se actualiza el software.

Al finalizar la configuración inicial debemos reiniciar.

Finalizada la configuración inicial aparece la siguiente pantalla de inicio.

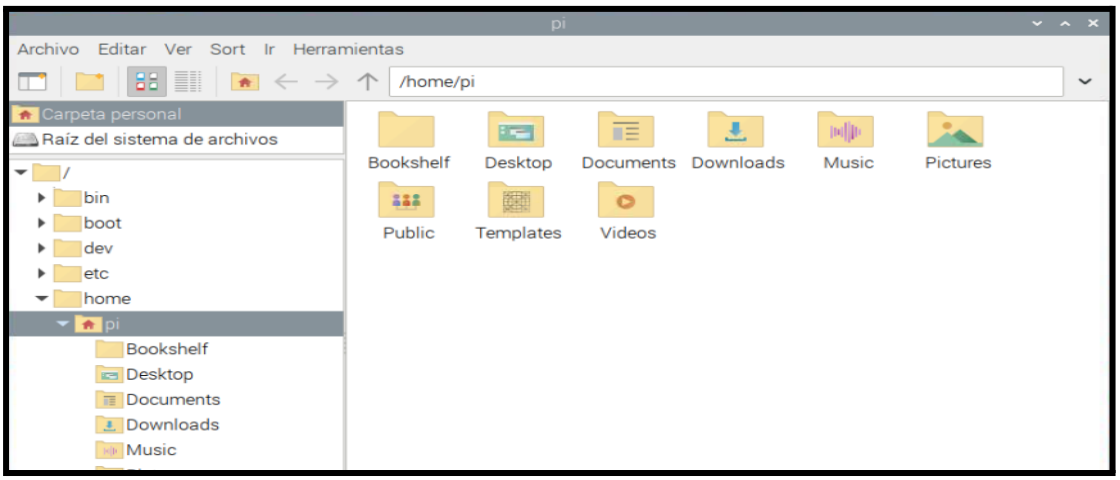


Marcando la frambuesa se nos despliega un menú con diferentes submenús, programación, acceso internet, sonido y vídeo, gráficos, accesorios, help y preferencias.

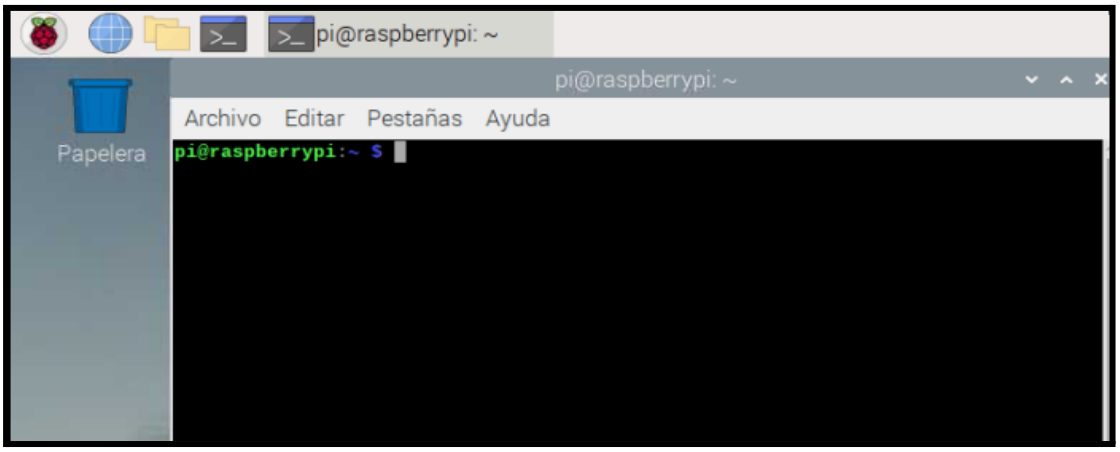


Chromium es el navegador web de Raspberry.

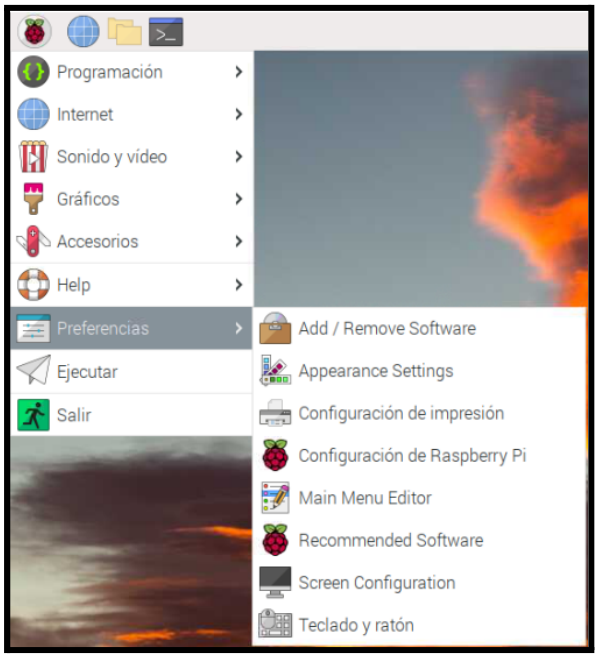
Disponemos de Explorador de archivos.



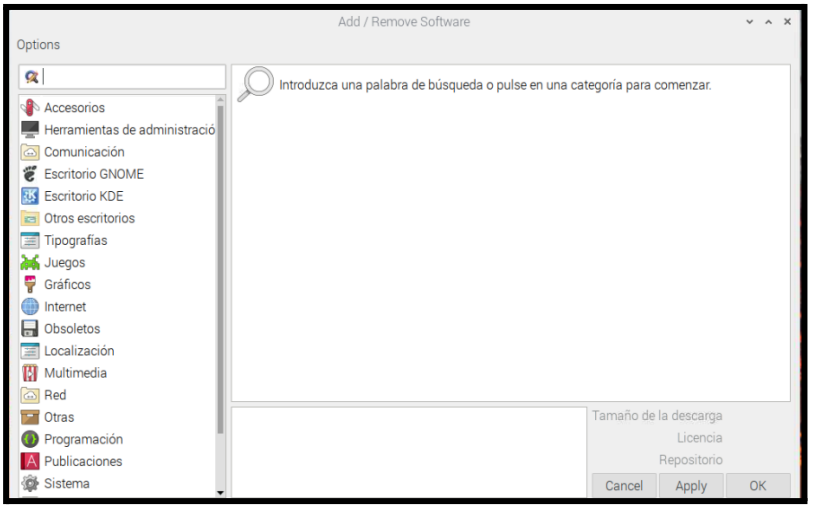
Acceso al terminal de comandos.



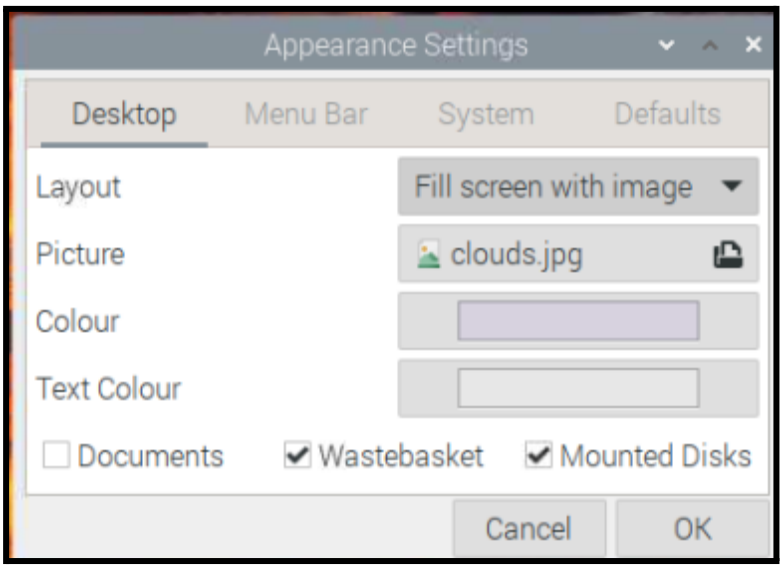
Vamos a profundizar en el menú de preferencias.



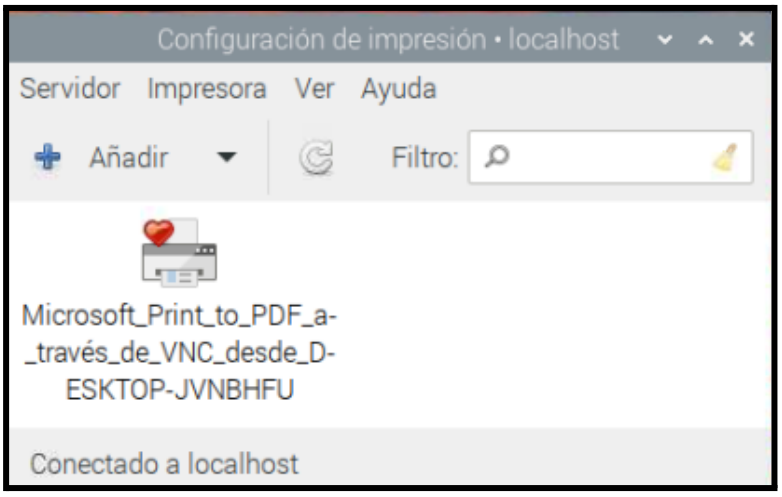
- Añadir o borrar software, aparece el software instalado y podemos añadir o borrar.



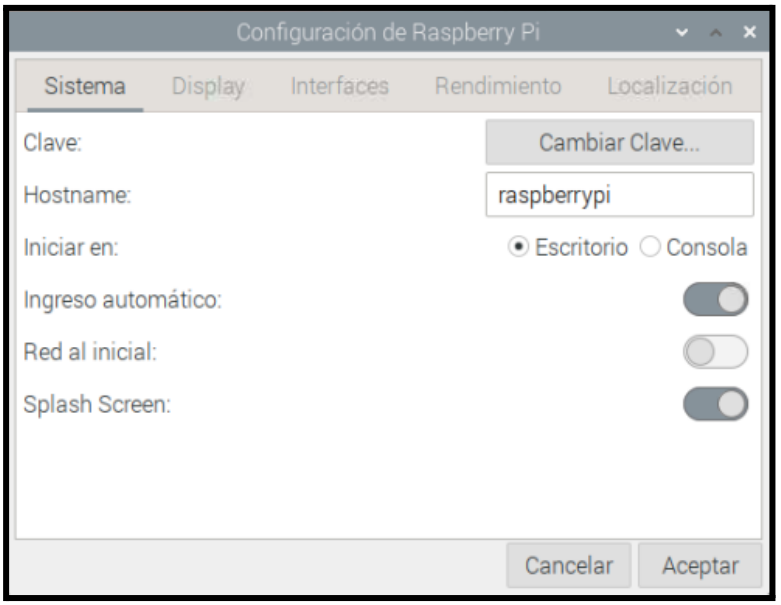
- Preferencias de apariencia



- Configuración de impresión



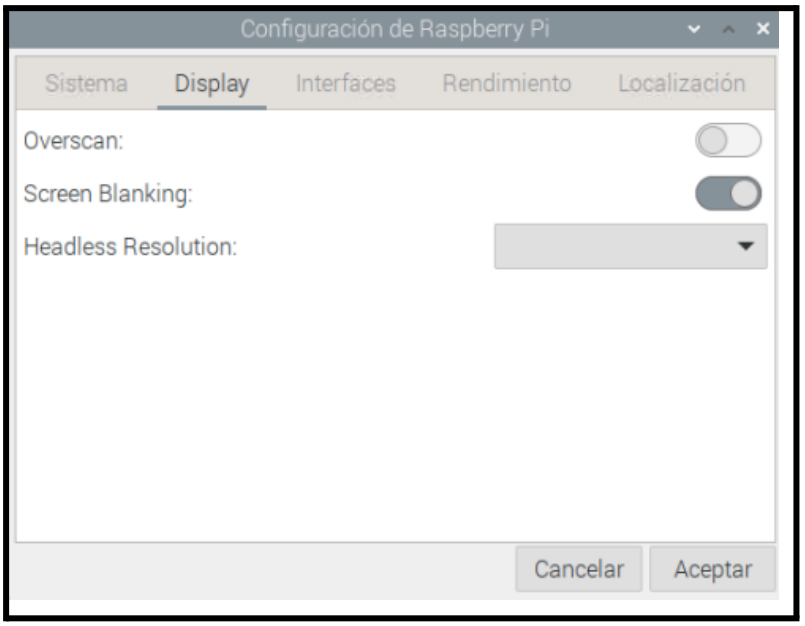
- Configuración de Raspberry Pi



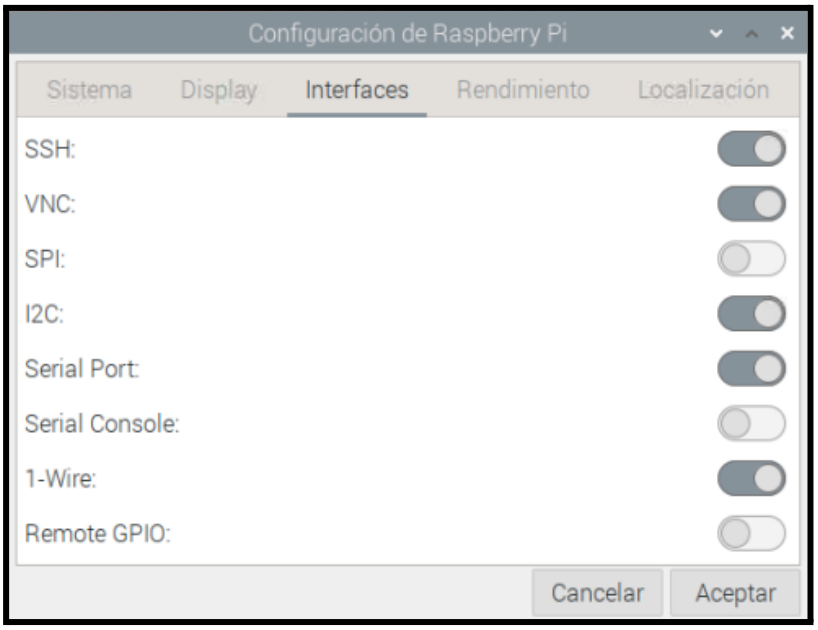
Sistema

- Cambiar la clave.
- Inicio Escritorio/Consola, el modo de iniciar la Raspberry.
- Ingreso automático,el sistema operativo Raspbian iniciará sesión automáticamente en el usuario predeterminado ‘pi’ sin solicitar una contraseña. Por razones de seguridad, se recomienda desmarcar esta opción, para que el sistema solicite una contraseña en cada arranque.
- Red al inicial, arranque sólo si estamos en red.
- Splash Screen, activamos si deseamos una pantalla de bienvenida.

Display

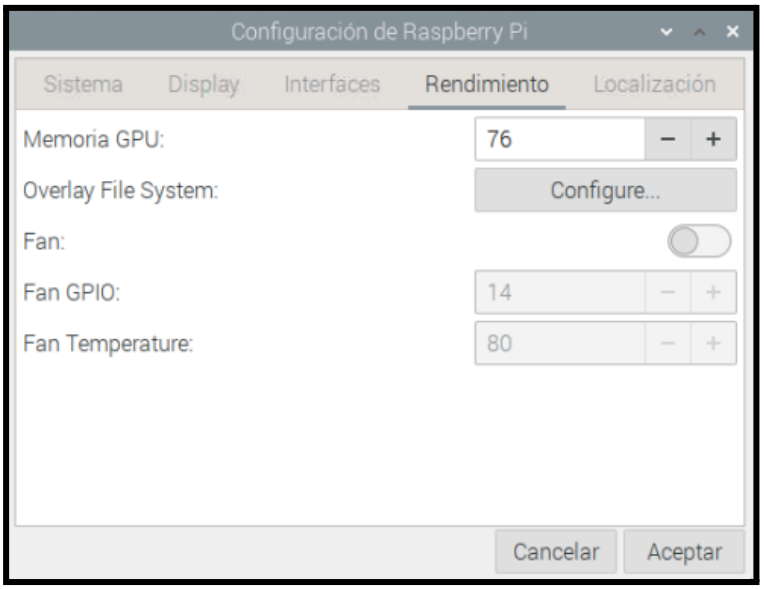


Interfaces



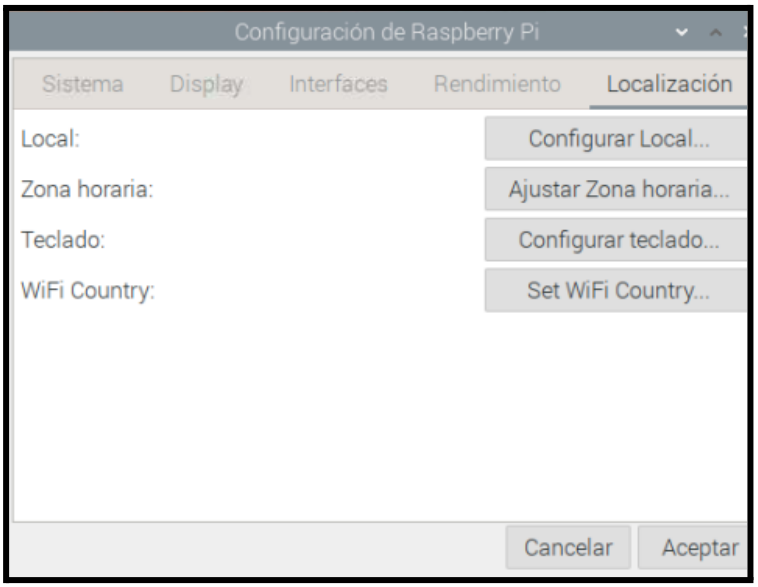
- **SSH**, permite el acceso remoto usando SSH.
- **VNC**, permite el acceso remoto usando VNC.
- **SPI** (o Interfaz Periférica Serial) permite la comunicación con más de 100 componentes periféricos a la vez. La Raspberry actúa como “maestro” frente a todos los dispositivos conectados como “esclavos” permite comunicarse con ellos a alta velocidad utilizando 4 cables. Para ello asigna y habilita los pines **SPI**.
- **I2C**, usa solo dos cables. Habilita los pines **I2C**.
- **Serial Port**, habilita los pines serie (**Rx, Tx**).
- Serial Console (permitiría la conexión en modo consola a través de del puerto serial, nos aseguramos de tenerlo desactivado)
- **1-Wire**. Permite la conexión serie con dispositivos mediante un solo hilo
- Remote GPIO, permite acceder a los pines GPIO de tu Raspberry Pi desde otra computadora

Rendimiento



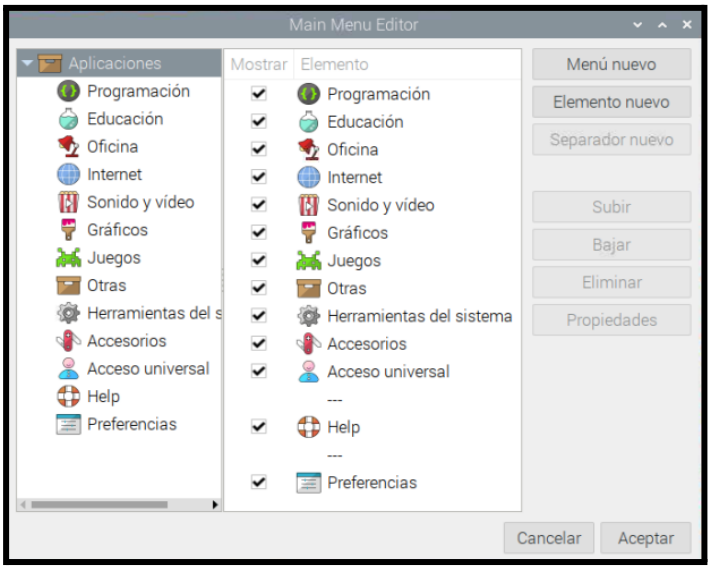
Se recomienda dejar los valores predeterminados.

Localización

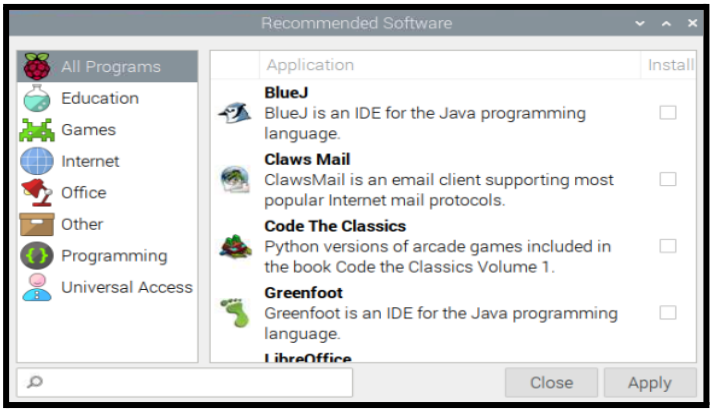


Se puede configurar la zona horaria, la configuración local y la distribución del teclado.

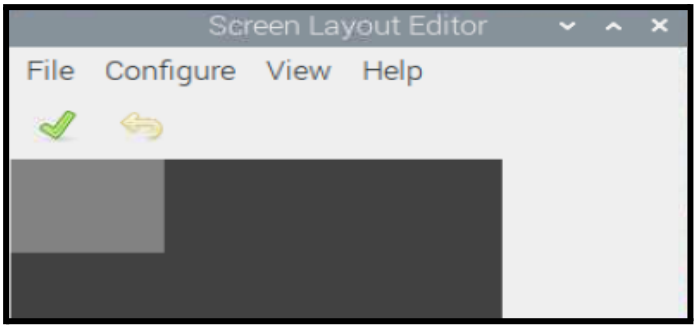
- Main menu editor



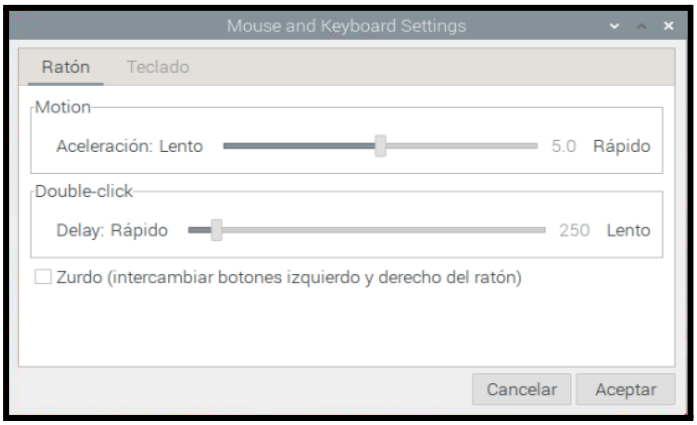
- Software recomendado para usar Raspberry



- Configuración de la pantalla

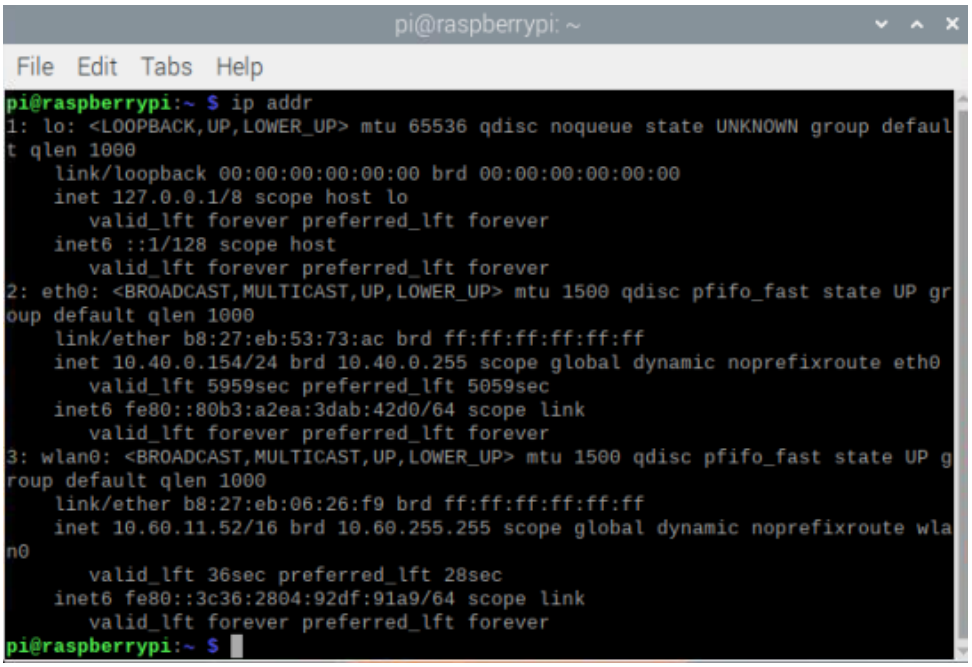


- Teclado y ratón



Asignar una IP fija a Raspberry Pi

Ejecutamos **ip addr** en la terminal, la IP que tenemos asignada aparecerá en el valor inet.



Para poder asignar una IP fija a nuestra Raspberry Pi tenemos que modificar el archivo **/etc/dhcpcd.conf** con el siguiente comando: **sudo nano /etc/dhcpcd.conf**

En este archivo debemos añadir las siguientes líneas al final del archivo, dependiendo de los datos que hayamos conseguido en el primer paso.
En la línea **interface** escribiremos el nombre de la interfaz a la que queremos asignarle la IP estática. Si es el cable de red será la interface **eth0** y si es la WiFi será la interface **wlan0**.
A continuación, añadiremos una línea llamada static ip_address=, con la IP que queremos asignar a la Raspberry Pi terminado en /24. En la siguiente línea escribiremos static routers=10.40.0.1 donde 10.40.0.1 será la IP de la puerta de enlace. Y en la última línea

pondremos static domain_name_servers= y los servidores DNS que tengamos, siempre separados por un espacio. En mi caso quedaría como sigue:

interface eth0
static ip_address=10.40.0.154/24
static routers=10.40.0.1
static domain_name_servers=10.50.0.11 8.8.8.8 8.8.4.4

```
# Static IP configuration:
interface eth0
static ip_address=10.40.0.154/24
#0static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=10.40.0.1
static domain_name_servers=aeg.eus 8.8.8.8 8.8.4.4
```

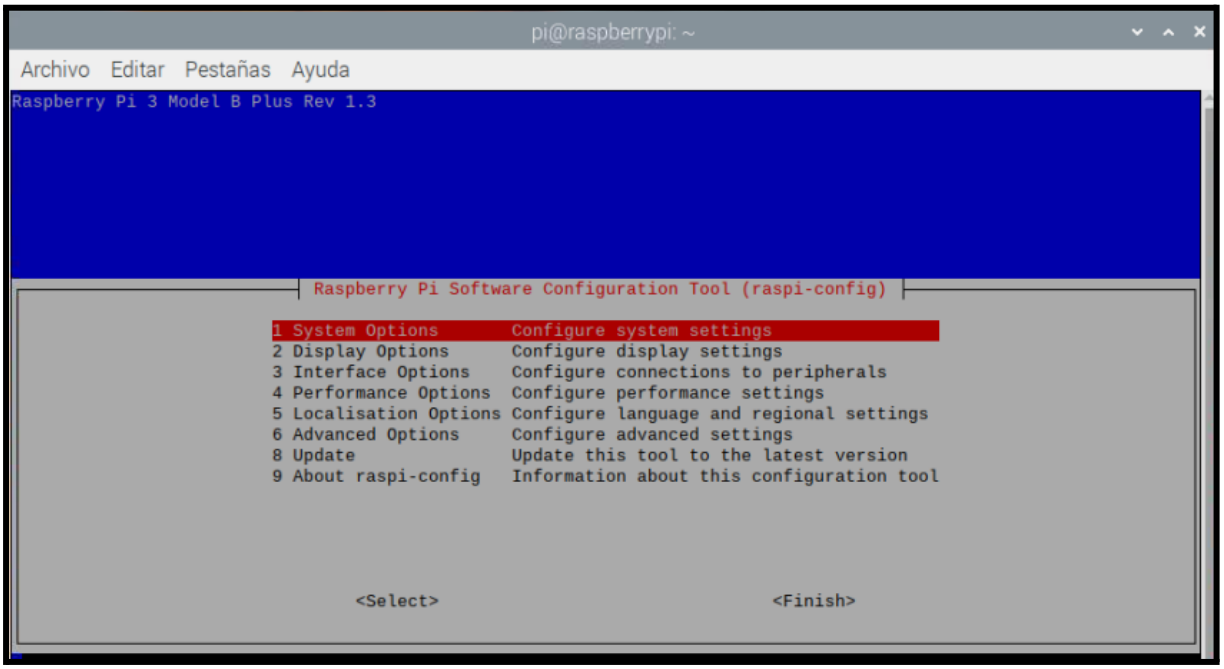
Configuración por comandos

Si queremos configurar Raspberry Pi desde la consola de comandos.

Desde una consola de comandos, ejecutamos el comando **sudo raspi-config**

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

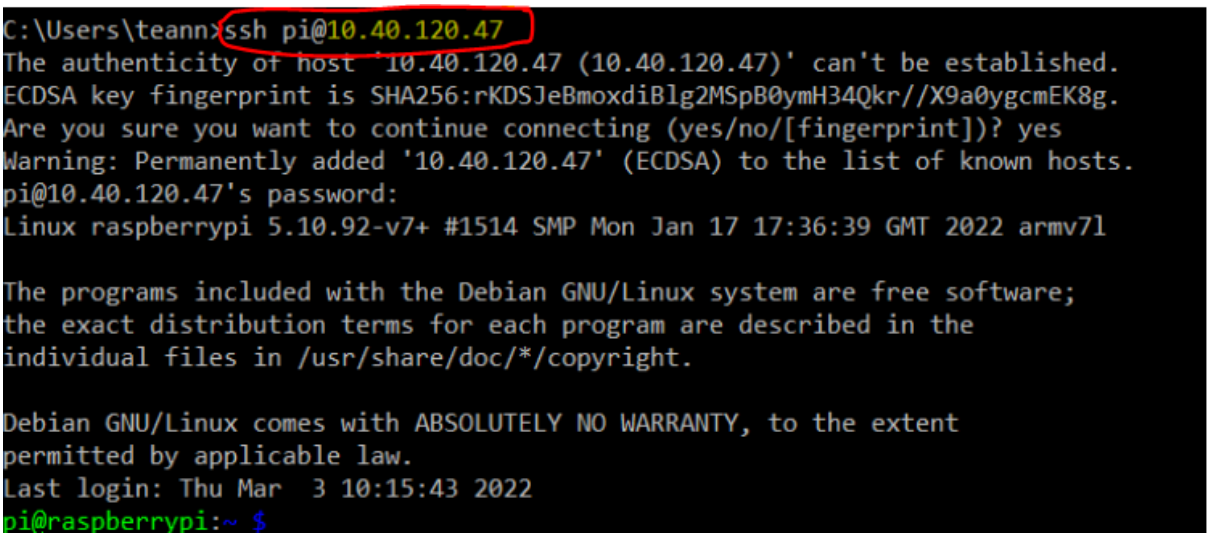
Aparece una ventana en la que tenemos las mismas opciones que hemos visto en el anteriormente en el entorno gráfico.



Conexión remota

Los dos métodos para como controlar una raspberry pi en forma remota son:

- **SSH**, permite administrar en forma remota a sistemas basados en Linux. El ordenador desde el que nos vamos a conectar a Raspberry debe tener instalado un cliente SSH.
  Para conectarnos debemos conocer el usuario y contraseña además de la dirección IP de Raspberry.
  En la terminal del ordenador ejecutamos ssh usuario@IP (de Raspberry) y se nos solicitará la contraseña.



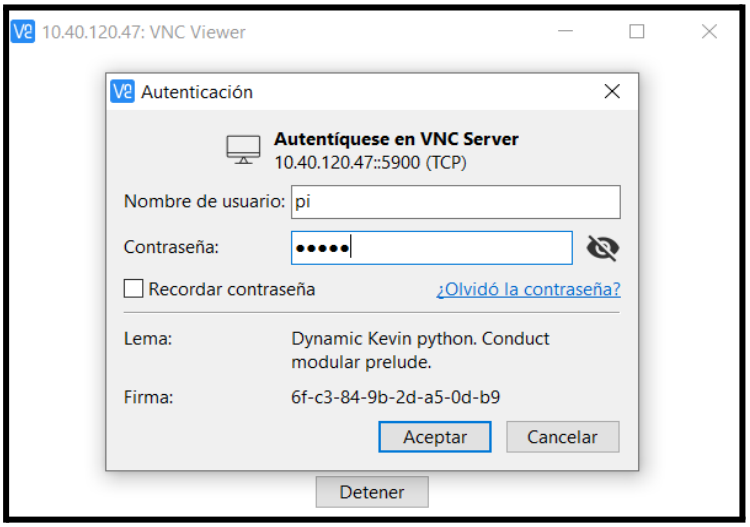
- **VNC** Server/Viewer, nos permite controlar en forma remota y con un escritorio virtual (como si tuviéramos un monitor virtual) a la Raspberry.
  Existen dos versiones.

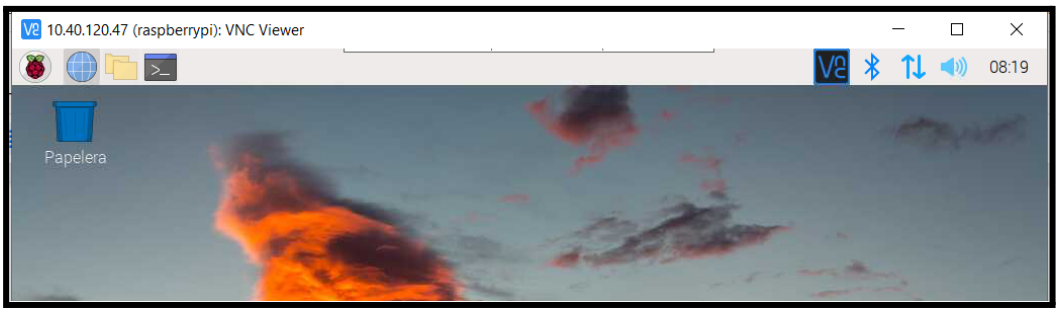
  - VNC Server. Es el programa que se instalará en la computadora (Raspberry) que quiere ser controlada remotamente.
    Cuando hemos habilitado la opción se ha instalado el programa.
  - VNC Viewer o cliente. Es el programa cliente el cual nos permitirá conectarnos a Raspberry.
    En el ordenador instalaremos VNC Viewer realizando la descarga desde el siguiente enlace https://www.realvnc.com/es/connect/download/viewer/
    Seleccionamos nuestro sistema operativo y descargamos.

Ejecutamos el archivo descargado para realizar la instalación de VNC Viewer.

Ejecutamos VNC Viewer y ponemos la dirección IP de Raspberry.

Se solicita el usuario y contraseña de Raspberry.

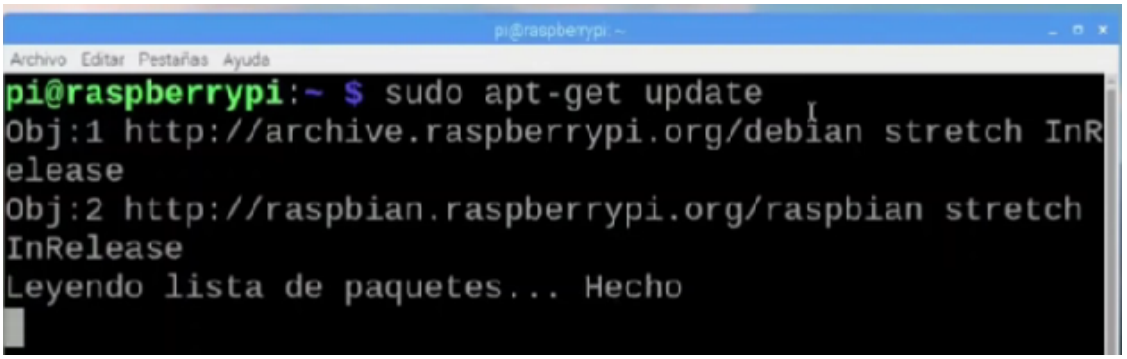



10.40.120.47 (raspberrypi): VNC Viewer
08:19
Papelera

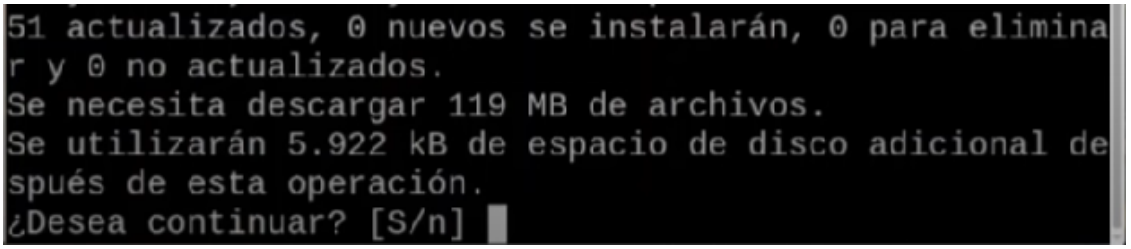
Comandos básicos SO

Podemos desde la interfaz gráfica pulsar Terminal o Ctrl+Alt+T.
Debemos actualizar el sistema y para ello ejecutamos dos comandos:

- **sudo apt-get update**
  Actualiza la lista de repositorios para la descarga de software



- **sudo apt-get upgrade**
  Actualiza el software de la lista que acabo de actualizar. Nos aparece información de lo que se va a descargar y el espacio necesario. Contestaremos S a la pregunta.

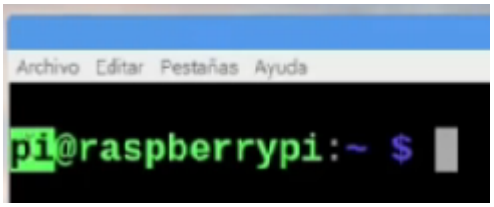


Si fuera necesario instalar un programa nuevo ejecutaremos:

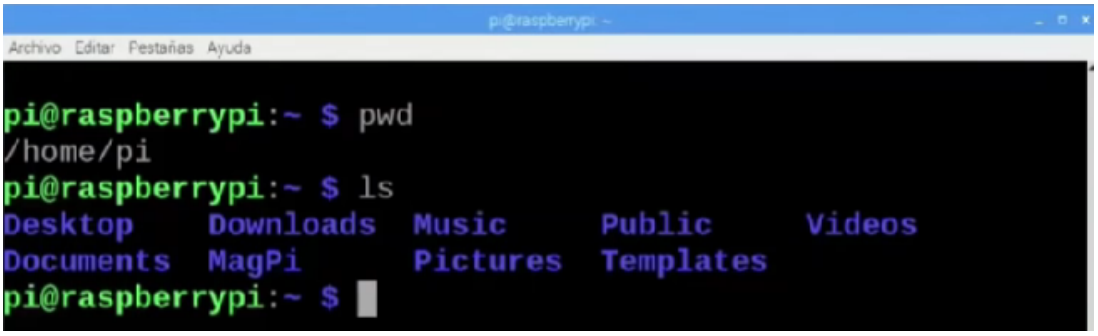
- **sudo apt-get install** "*nombre del programa*"

Si deseas limpiar el terminal ejecutar el comando **clear**.

Durante la instalación se ha creado un usuario por defecto **pi** y **raspberrypi** no indica que estamos conectados a la placa o ordenador que está identificado con esa etiqueta.



Usa el comando **pwd** para encontrar la ruta del directorio (carpeta) de trabajo actual en el que te encuentras.
El comando **ls** se usa para ver el contenido de un directorio.



cd se usa para navegar por los archivos y directorios de Linux. Te pedirá la ruta completa o el nombre del directorio, dependiendo del directorio de trabajo actual en el que te encuentres.

Supongamos que estás en /home/nombredeusuario/Documentos y deseas ir a Fotos, un subdirectorio de Documentos. Para hacerlo, simplemente escribe el siguiente comando:

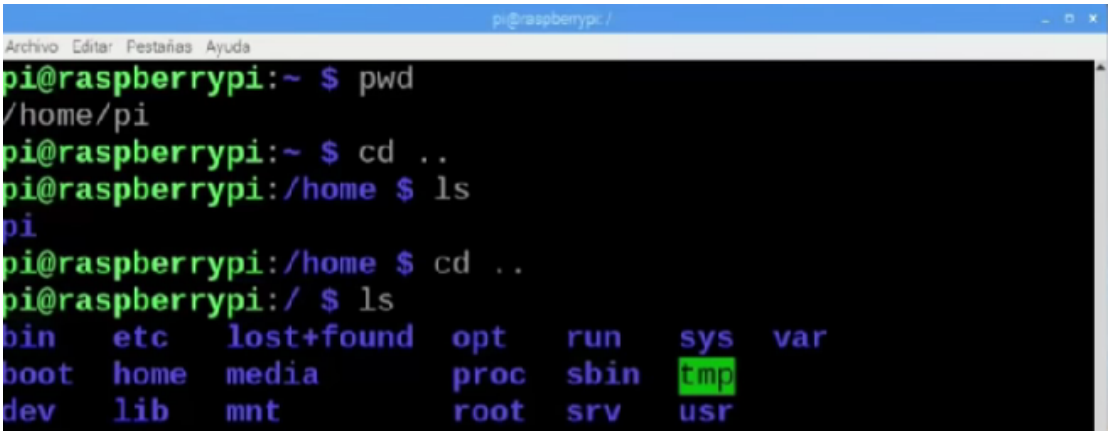
- cd Fotos.

Otro escenario es si deseas ir a un directorio completamente nuevo, por ejemplo, /home/nombredeusuario/Películas. En este caso, debes escribir cd seguido de la ruta absoluta del directorio:

- cd /home/ nombredeusuario/Películas.

Hay algunos atajos para ayudarte a navegar rápidamente:

- cd .. (con dos puntos) para ir un directorio hacia arriba
- cd para ir directamente a la carpeta de inicio
- cd- (con un guión) para ir al directorio anterior



mkdir para crear un nuevo directorio

sudo, abreviatura de «SuperUser Do» (SuperUsuario hace), este comando te permite realizar tareas que requieren permisos administrativos o raíz.

touch te permite crear un nuevo archivo vacío.

```
pi@raspberrypi:~/Desktop/micarpeta2 $ touch texto2
```

rm se usa para eliminar directorios y el contenido dentro de ellos.

```
pi@raspberrypi:~/Desktop/micarpeta2 $ rm texto1
```

rmdir elimina un directorio o carpeta.

```
pi@raspberrypi:~/Desktop $ rmdir micarpeta2
```

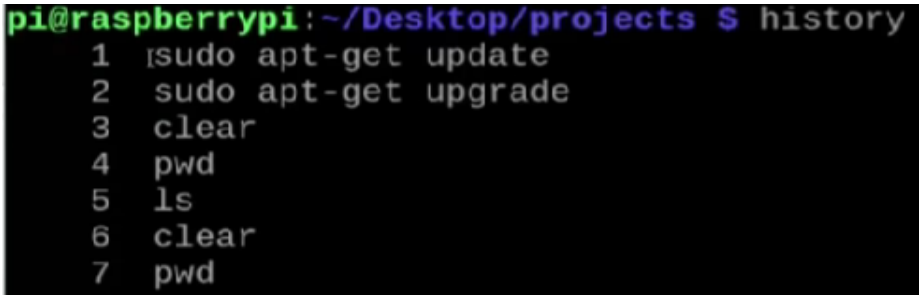
Si tenemos que editar un archivo podemos utilizar el editor **NANO**.

```
pi@raspberrypi:~/Desktop/projects $ nano texto.txt
```

cat se utiliza para listar el contenido de un archivo.

```
pi@raspberrypi:~/Desktop/projects $ cat texto.txt
este es mi primer archivo usando nano
este es la sengunda linea escrita con nano
pi@raspberrypi:~/Desktop/projects $
```

history se listan los comandos que han sido introducidos en el terminal.


pi@raspberrypi:~/Desktop/projects $ history
1 sudo apt-get update
2 sudo apt-get upgrade
3 clear
4 pwd
5 ls
6 clear
7 pwd

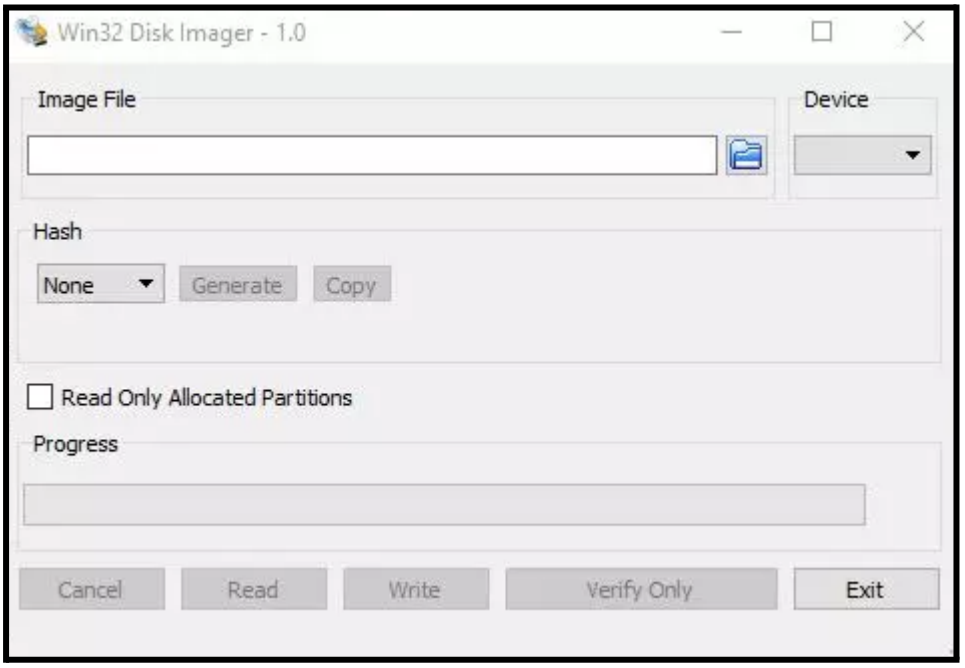
Hacer copia de seguridad

Hay varias formas de hacer copias de seguridad de la tarjeta de un Raspberry Pi. Dependiendo de si la hacemos desde Windows, o desde Linux (incluso desde el propio micro-ordenador) el proceso que debemos seguir será de una forma u otra. Vamos a verlo en detalle.

Desde Windows

Debemos conectar la tarjeta SD de la Raspberry a nuestro ordenador. Es probable que nos aparezca un mensaje que nos indica que la tarjeta de memoria no tiene formato y que la tengamos que formatear. Es importante NO hacerlo e ignorar el mensaje. Aparece porque la micro-sd estará en un sistema de archivos Linux y Windows no podrá leer sus archivos.

Descargamos el programa gratuito y de código abierto **Win32 Disk Imager** y lo instalamos en el ordenador. Una vez instalado, lo ejecutamos y veremos una ventana como la siguiente:



Hacemos clic sobre el botón de la carpeta que aparece en **Image File** y seleccionaremos el directorio donde queremos que se guarde la copia de seguridad que vamos a hacer. También seleccionaremos en el apartado **Device** la unidad de la que vamos a hacer la copia.

Este programa hace una copia «RAW«, es decir, en crudo, de los 1 y 0 de los datos. Da igual el contenido, se crea una copia literal de la unidad (incluso del espacio libre de la misma) para que no haya ningún problema.

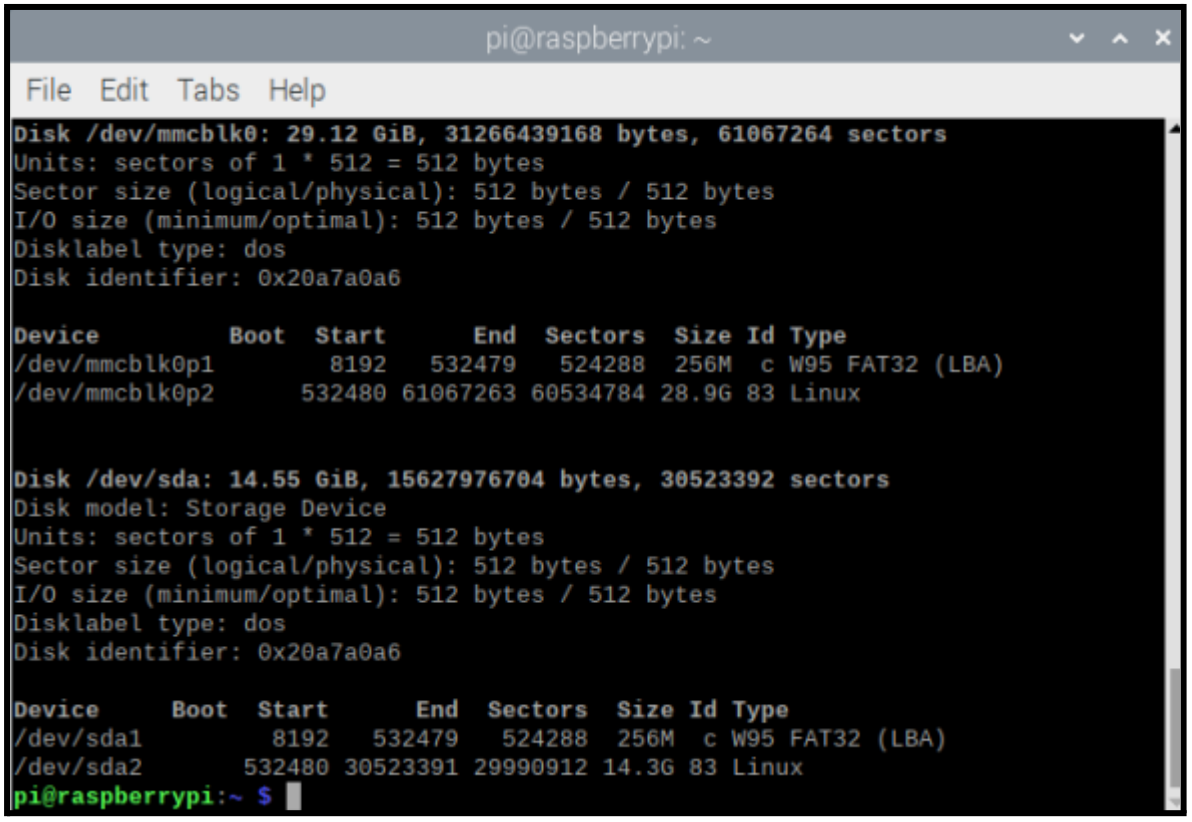
Una vez listo todo, hacemos clic sobre el botón **Read** y comenzará el proceso de creación de la imagen. Este proceso puede tardar varios minutos, y ocupará lo mismo que la capacidad de la tarjeta de memoria original.

Cuando termine el proceso de lectura, la tendremos la imagen (un archivo, con formato .img) en el directorio que le hayamos especificado. Ya podemos sacar la tarjeta del ordenador, y volverla a conectar al Raspberry Pi.

Desde Linux

Si lo vamos a hacer desde un sistema Linux (e incluso desde el propio Raspberry Pi a una unidad externa que tengamos, por ejemplo, por USB o en red), lo que debemos hacer es conectar la tarjeta al ordenador (si no la tenemos ya) y localizar su punto de montaje usando el siguiente comando:

sudo fdisk -l



Suponiendo que sea /dev/sdb (que variará en función de las unidades conectadas al PC), a continuación, haremos uso del comando **dd** para clonar los datos RAW de la unidad principal (la tarjeta) al directorio que queramos. Este comando tiene la siguiente estructura:

sudo dd if=/dev/sdb of=[directorio]/raspbackup.img

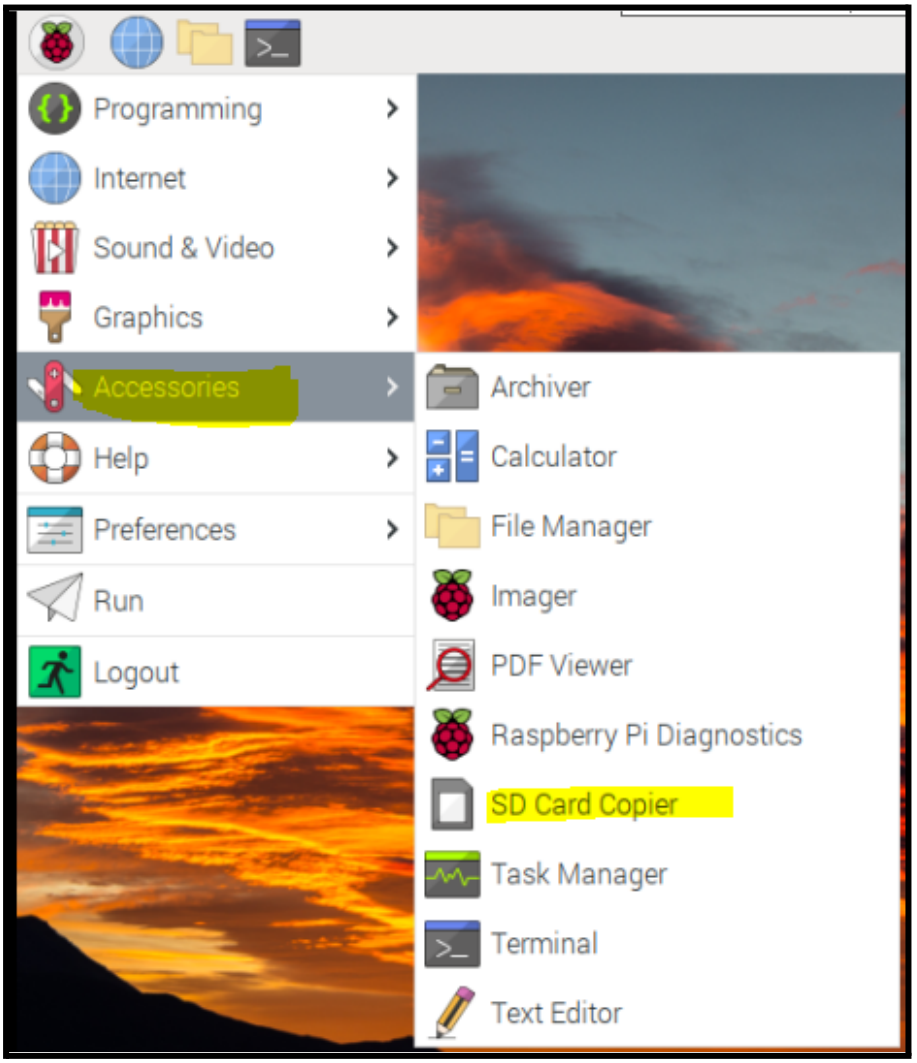
Debemos cambiar [directorio] por la ruta donde queramos guardar la imagen. Ya sea una unidad de red, u otro punto de montaje diferente. Lo que es lógico es que no debemos crear

la copia de seguridad en la misma micro-sd, ya que, si se pierden los datos, y la copia de seguridad, de poco nos sirve.

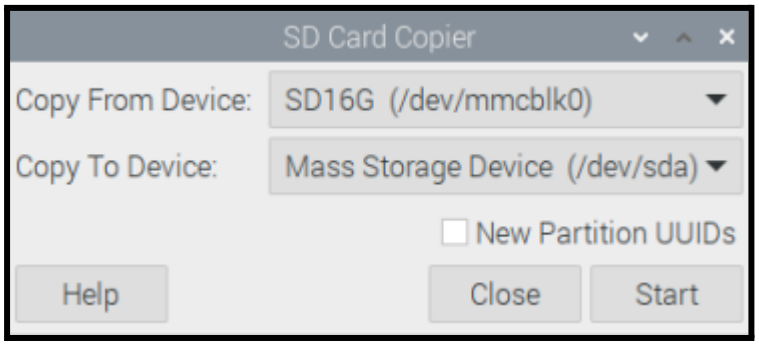
Desde Raspberry Pi

Desde el menú principal, accesorios y SD Card Copier, (piclone) es la herramienta que integra el sistema operativo Raspbian para el clonado de tarjetas microSD de forma gráfica, es decir, podemos copiar todo el contenido de una tarjeta microSD a otra haciendo un clon idéntico sin utilizar la línea de comandos.

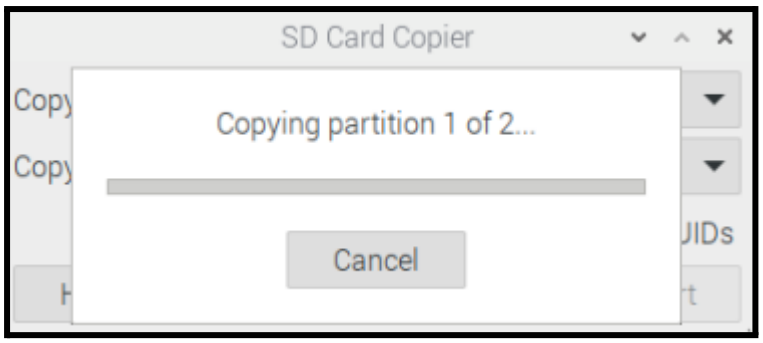
Para utilizar la aplicación deberás disponer de un hub USB de tarjetas SD.



Nos aparece una pantalla donde debemos seleccionar el dispositivo desde donde queremos copiar (seleccionando la actual SD /dev/mmcblk0) y hacia dónde, en nuestro caso, un Hub USB utilizando una tarjeta SM en /dev/sdd.



Pulsamos Start



Restaurar una copia de la SD

Para restaurar la copia de seguridad en caso de que la tarjeta original falle, o la cambiemos por una más grande, literalmente lo que debemos hacer es seguir los pasos a la inversa.

En el caso de Windows, por ejemplo, usaremos de nuevo el programa Win32 Disk Imager, seleccionando en **Image File** la imagen de la copia de seguridad, en **Device** la letra de la unidad de la micro-sd donde vamos a escribir la imagen, y haremos clic sobre el botón **write**. El proceso borrará todos los datos, y cuando termine tendremos la tarjeta de memoria lista para funcionar de nuevo en el Raspberry Pi.

En el caso de Linux, lo que debemos hacer es usar de nuevo el comando **dd**, pero de forma inversa. Es decir, en el parámetro **if** podremos el directorio de la imagen (img) de la copia de seguridad, y en el parámetro **of** el punto de montaje de la tarjeta de memoria donde vamos a volcar dicha imagen.

sudo dd if=[directorio]/raspbackup.img of=/dev/sdb

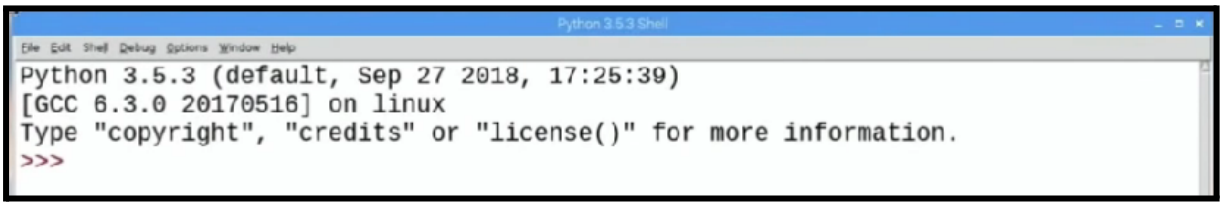
Programación

En Raspbian tenemos distintos editores y herramientas que nos permitirán desarrollar programas de forma sencilla que le indiquen que tareas realizar al Raspberry Pi.

Entre algunos de los lenguajes de programación que se pueden utilizar está Python, C, Scratch y Java.

Raspberry Pi es un ordenador por lo que podemos instalar prácticamente cualquier compilador de lenguaje de programación.

En el sistema operativo Raspbian que hemos instalado viene por defecto Python3.



Python permite trabajar con ciencia de datos, inteligencia artificial y visión por computadora.

Para programar la raspberry utilizaremos Node-Red.

Node-RED

Node-RED es una herramienta que sirve para comunicar hardware y servicios de una forma muy rápida y sencilla. Simplifica enormemente la tarea de programar del lado del servidor gracias a la programación visual.

Instalación

Primero ejecutaremos **sudo apt install build-essential git curl** para asegurar que npm pueda obtener y compilar cualquier módulo binario que necesite instalar.

Es conveniente realizar una actualización del sistema. Ejecutamos **sudo apt-get update** y **sudo apt-get upgrade.**

```
pi@raspberrypi:~ $ sudo apt-get update
```

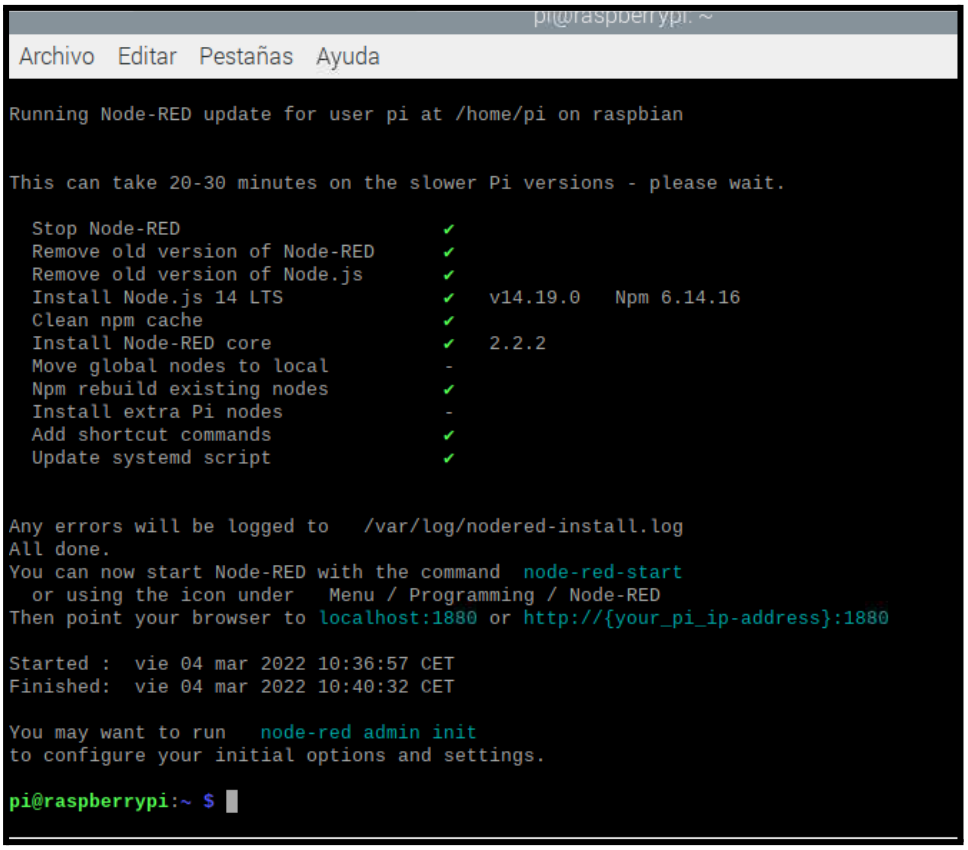
```
pi@raspberrypi:~ $ sudo apt-get upgrade
```

```
pi@raspberrypi:~ $ sudo apt install build-essential git curl
```

En la Raspberry abrimos la terminal y ejecutamos el siguiente script que instalará Node.js, npm y Node-RED.
pi@raspberrypi:~ $ **bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)**

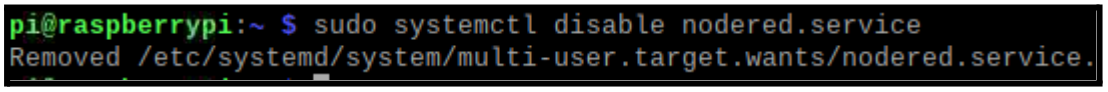
Nos realizará preguntas de confirmación de la instalación y se instalará Node.js, npm y Node-RED.

```
pi@raspberrypi:~ $ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/
update-nodejs-and-nodered)


This script checks the version of node.js installed is 12 or greater. It will try to
install node 14 if none is found. It can optionally install node 12, 14 or 16 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] ? y
```

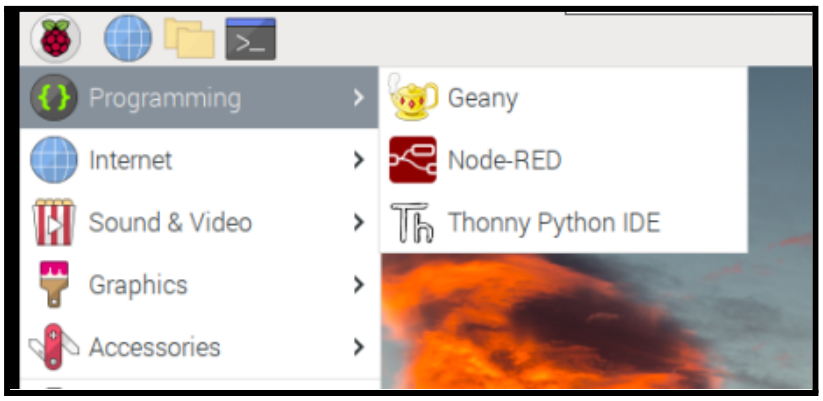


Si queremos que Node-RED se ejecute cuando la Raspberry se enciende o reinicie, habilitaremos el inicio automático del servicio ejecutando el comando
sudo systemctl enable nodered.service

Para deshabilitar el servicio ejecutamos el comando
sudo systemctl disable nodered.service



También se puede iniciar el servicio Node-RED desde el escritorio del sistema operativo Raspberry Pi seleccionando la opción -> Programación -> Nodo-RED.



Configuración de seguridad

Node-RED no está protegido y cualquier persona que pueda acceder a su dirección IP puede acceder al editor e implementar cambios.

Usuario y contraseña

Para crear un acceso a node-red con usuario y contraseña debemos realizar varios pasos.

- Lo primero se debe para el node-red, **sudo systemctl stop nodered**

```
sudo systemctl stop nodered
```

- Instalaremos el complemento **node-red-admin**.
  Ejecutaremos previamente ejecutaremos **sudo npm install -g npm**

```
sudo npm install -g npm
```

```
pi@raspberrypi:~ $ sudo npm install -g npm
/usr/bin/npm -> /usr/lib/node_modules/npm/bin/npm-cli.js
/usr/bin/npx -> /usr/lib/node_modules/npm/bin/npx-cli.js
+ npm@8.12.1
added 63 packages from 18 contributors, removed 299 packages and
```

  Ahora ejecutamos **sudo npm install -g --unsafe-perm node-red-admin**

```
pi@raspberrypi:~ $ sudo npm install -g --unsafe-perm node-red-admin
/usr/bin/node-red-admin -> /usr/lib/node_modules/node-red-admin/node-red-admin.js
+ node-red-admin@3.0.0
```

- Ejecutamos **node-red admin hash-pw,** se utiliza para cifrar una contraseña.

```
pi@raspberrypi:~ $ node-red admin hash-pw
Password:
$2b$08$DJA14/xbTe6KFy.4rrKkBOTj3tcdPAH1dJ/vaG89QkdNNmNWyFQkq
pi@raspberrypi:~ $
```

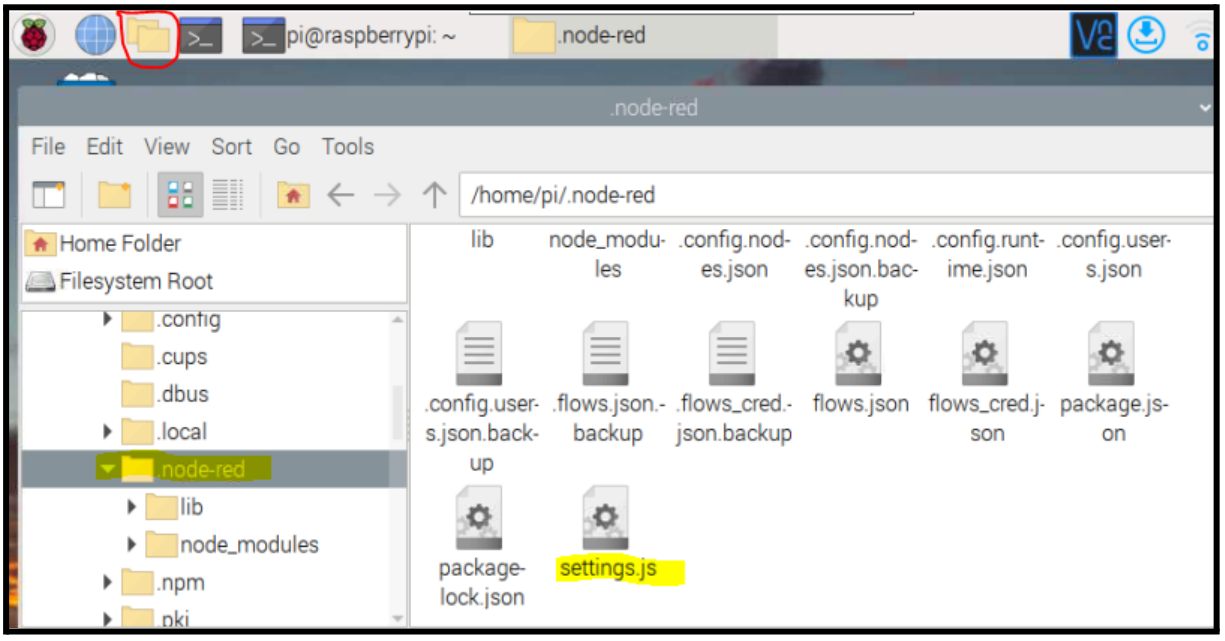
- Editar un fichero **settings.js** que se encuentra en el directorio oculto **.node-red**

```
pi@raspberrypi:~ $ ls -a
.               .bashrc     .cups       Downloads  .node-red  .pki        Videos            .xsession-errors.old
..              Bookshelf   .dbus       .local     .npm       .profile    .vnc
.bash_history   .cache      Desktop     modbus     .npmrc     Public      .Xauthority
.bash_logout    .config     Documents   Music      Pictures   Templates   .xsession-errors
pi@raspberrypi:~ $ cd .node-red
pi@raspberrypi:~/.node-red $ ls
flows.json  lib  node_modules  package.json  package-lock.json  settings.js
pi@raspberrypi:~/.node-red $
```

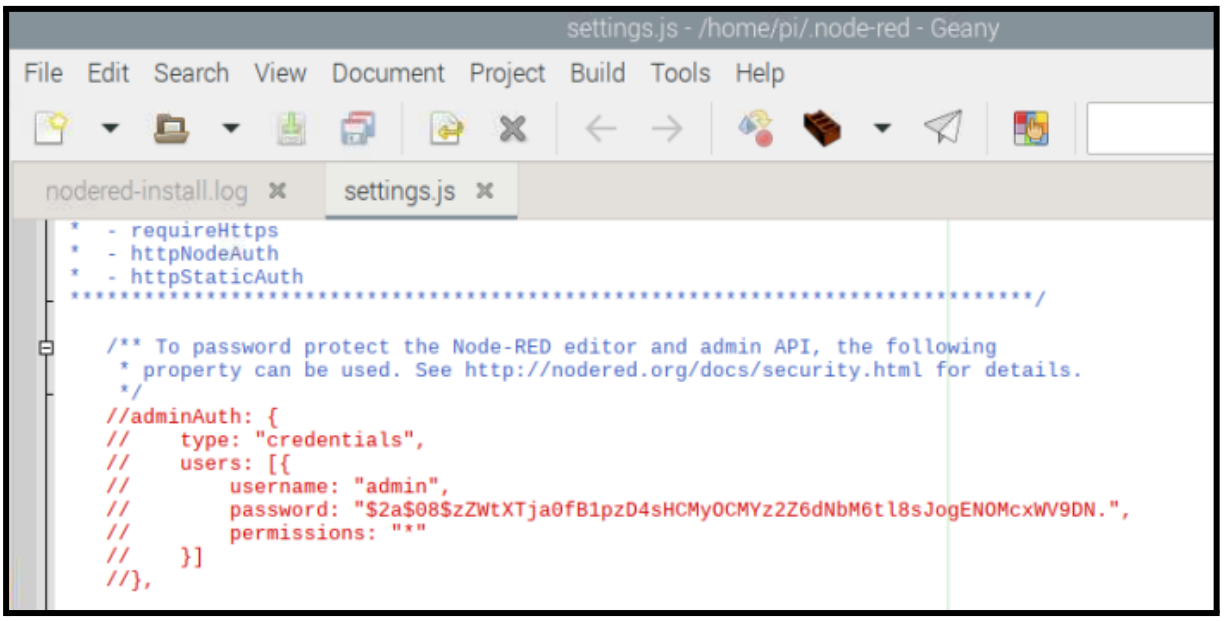
sudo nano /home/pi/.node-red/settings.js
Descomentamos las líneas que se muestran en la imagen y pondremos el usuario y la contraseña que acabamos de cifrar.

```
adminAuth: {
    type: "credentials",
    users: [{
        username: "pi",
        password: "$2b$08$M0UF12rJH7fV0Gq3xJ7kTuicDOV5viLQ7bWo6PWWs4VUNoxQWzG.m",
        permissions: "*"
    }]
},
```

Podemos realizar esta modificación desde el menú principal de Raspberry seleccionando administrador de archivos y accediendo a la carpeta pi/.node-red



Buscamos la sección adminAuth.



Descomentamos toda la sección y en username ponemos el usuario que queramos y en password copiaremos la contraseña que acabamos de cifrar.

```
adminAuth: {
    type: "credentials",
    users: [{
        username: "pi",
        password: "$2b$08$5kRlcRG273bsoN9Zcg1gbeyak1V6QchOysP4XoMBuPJjCs9nVteom",
        permissions: "*"
    }]
},
```

- Arrancamos node-red, **sudo systemctl start nodered**

```
sudo systemctl start nodered
```

Desde ahora al acceder a Node-Red se nos solicitará usuario y contraseña

Autenticación https

Para proteger las rutas HTTP expuestas por los nodos y el dashboard se puede usar una autenticación básica.

La propiedad httpNodeAuth en su archivo settings.js se puede usar para definir un nombre de usuario y contraseña únicos que podrán acceder a las rutas.

Primero paramos Node-red ejecutando s**udo systemctl stop nodered**

```
sudo systemctl stop nodered
```

Editamos el archivo **/home/pi/.node-red/settings.js** y descomentamos las líneas que se muestran en la imagen y pondremos el usuario y la contraseña que previamente hemos cifrado con **node-red admin hash-pw.**

```
httpNodeAuth: {user:"pi",pass:"$2b$08$JFO3/muax82xD54adn.y7uVPv2SuA0XH/8z4fIBb4JH7GTwGkdNhu"},
/*httpStaticAuth: {user:"",pass:""},
```

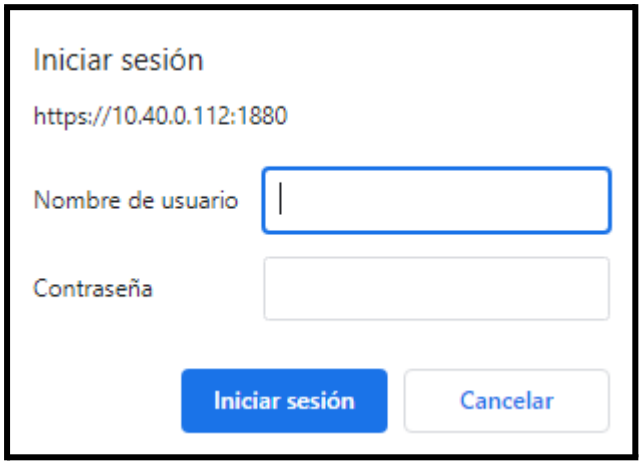
Podemos desde el menú de la raspberry editar el archivo y realizar la modificación.

```
/** To password protect the node-defined HTTP endpoints (httpNodeRoot),
 * including node-red-dashboard, or the static content (httpStatic), the
 * following properties can be used.
 * The `pass` field is a bcrypt hash of the password.
 * See http://nodered.org/docs/security.html#generating-the-password-hash
 */
httpNodeAuth: {user:"pi",pass:"$2b$08$5kRlcRG273bsoN9Zcg1gbeyak1V6QchOysP4XoMBuPJjCs9nVteom"},
//httpStaticAuth: {user:"user",pass:"$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN."},
```

Arrancamos node-red, **sudo systemctl start nodered**

```
sudo systemctl start nodered
```

Ahora al acceder nos pedirá usuario y contraseña



SSL, Certificados Autofirmados

Una conexión a través de SSL / TLS protege el tráfico de datos mediante el envío de paquetes cifrados. Para ello necesitamos un certificado de un proveedor confiable, que generalmente tienen un coste, o crearnos un certificado autofirmado.

Utilizaremos **openssl** para crear nuestra propia autoridad de certificación (**CA**), claves de servidor y certificados.

Estos pasos nos permitirán una conexión encriptada entre el servidor Node-Red y el cliente.

Primero paramos Node-red ejecutando s**udo systemctl stop nodered**

```
sudo systemctl stop nodered
```

El proceso para crear tu propia entidad certificadora es muy simple:

- Instalar openssl, **sudo apt-get install openssl**

```
pi@raspberrypi:~ $ sudo apt-get install openssl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssl is already the newest version (1.1.1n-0+deb11u1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

- Crear un directorio llamado certificados en /home/pi/, **mkdir Certificados** y entrar en el directorio para guardar los certificados y claves.

```
pi@raspberrypi:~ $ mkdir Certificados
pi@raspberrypi:~ $ cd Certificados
```

- Generamos la llave de CA propia, **openssl genrsa -des3 -out ca.key 2048**

```
pi@raspberrypi:~/Certificados $ openssl genrsa -des3 -out ca.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
.....................................................................
.....................+++++
e is 65537 (0x010001)
Enter pass phrase for ca.key:
Verifying - Enter pass phrase for ca.key:
```

- Generamos certificado de la propia autoridad certificadora (CA), **openssl req -new -x509 -days 1826 -key ca.key -out ca.crt**
  Se nos solicitan varios datos, el Common Name es descriptivo (Autoridad Certificadora Raspberry), el resto de datos no son obligatorios.

```
pi@raspberrypi:~/Certificados $ openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Guipuzcoa
Locality Name (eg, city) []:Donostia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:AEG
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Raspberry
Email Address []:
pi@raspberrypi:~/Certificados $
```

- Generamos una llave privada para el servidor, **openssl genrsa -out server.key 2048**

```
pi@raspberrypi:~/Certificados $ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
...+++++
e is 65537 (0x010001)
```

- Generamos una solicitud de certificado para el servidor, **openssl req -new -out server.csr -key server.key**

```
pi@raspberrypi:~/Certificados $  openssl req -new -out server.csr -key server.key
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Guipuzcoa
Locality Name (eg, city) []:Donostia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:AEG
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Raspberry
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
pi@raspberrypi:~/Certificados $
```

- Firmamos la solicitud de certificado con la llave privada de la autoridad CA **openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360**

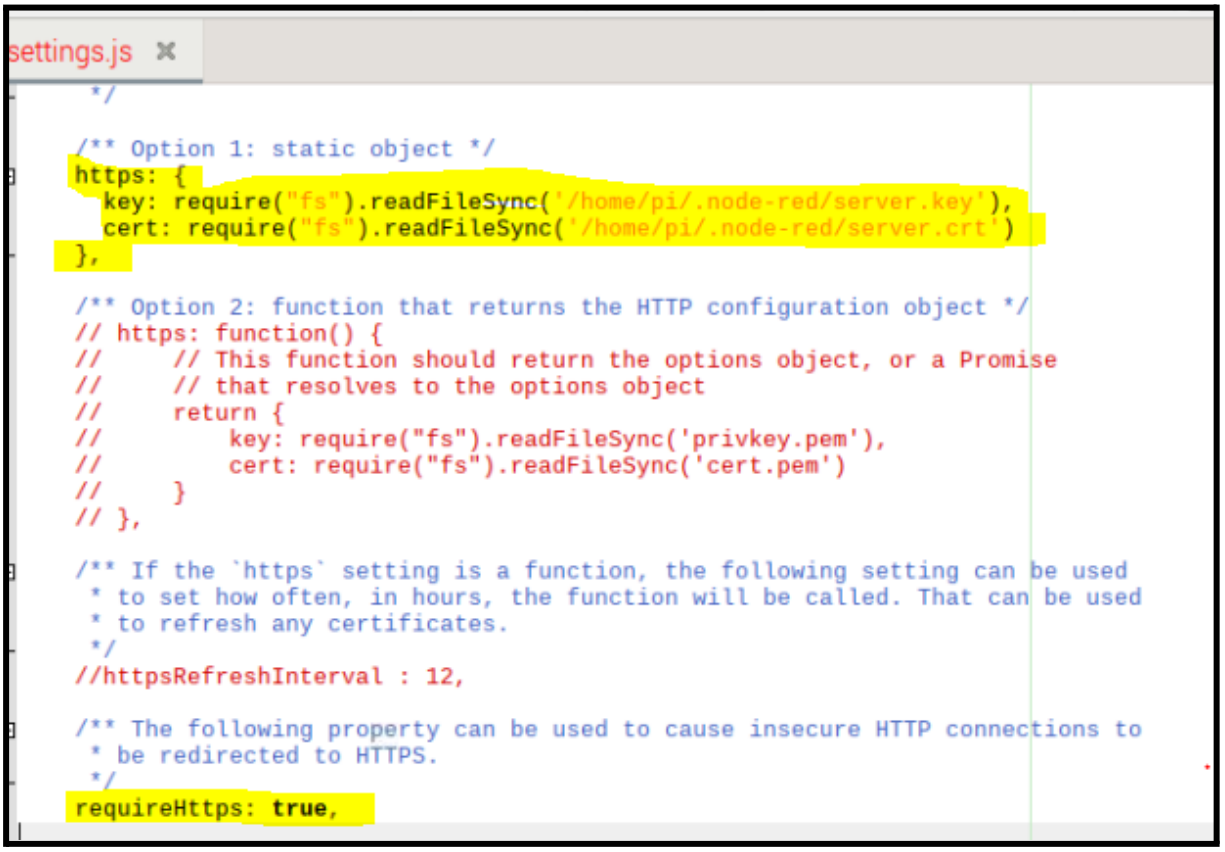
```
pi@raspberrypi:~/Certificados $ openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out ser
ver.crt -days 360
Signature ok
subject=C = ES, ST = Guipuzcoa, L = Donostia, O = AEG, CN = Raspberry
Getting CA Private Key
Enter pass phrase for ca.key:
pi@raspberrypi:~/Certificados $
```

- Copiamos los certificados a la carpeta de usuario donde corre node-red. **cp /home/pi/Certificados/server.key /home/pi/.node-red/server.key**

cp /home/pi/Certificados/server.crt /home/pi/.node-red/server.crt

```
pi@raspberrypi:~ $ cp /home/pi/Certificados/server.key /home/pi/.node-red/server.key
pi@raspberrypi:~ $ cp /home/pi/Certificados/server.crt /home/pi/.node-red/server.crt
```

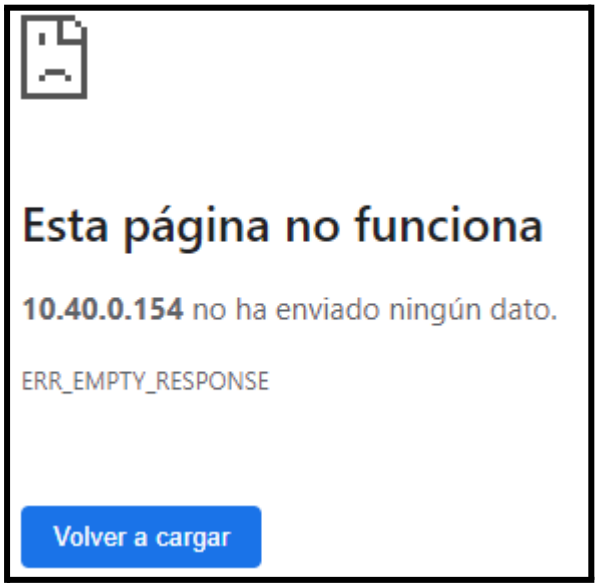
- Editamos el archivo **/home/pi/.node-red/settings.js** y descomentamos las líneas que se muestran en la imagen.



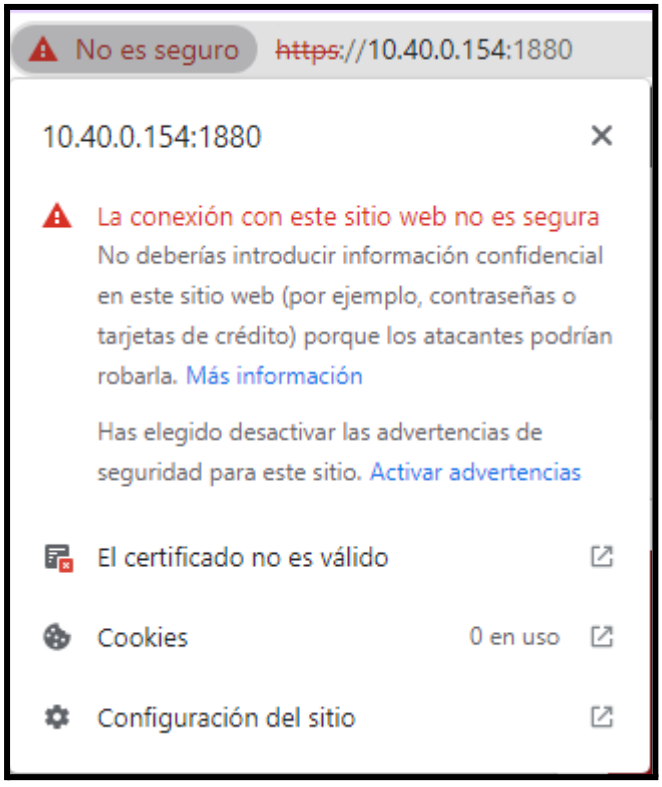
Arrancamos node-red, **sudo systemctl start nodered**

```
sudo systemctl start nodered
```

Entrar en Node-RED de nuevo en **http://{pi_IP}:1880** y comprobar que no funciona.



Ahora debemos acceder con **https://{pi_IP}:1880**

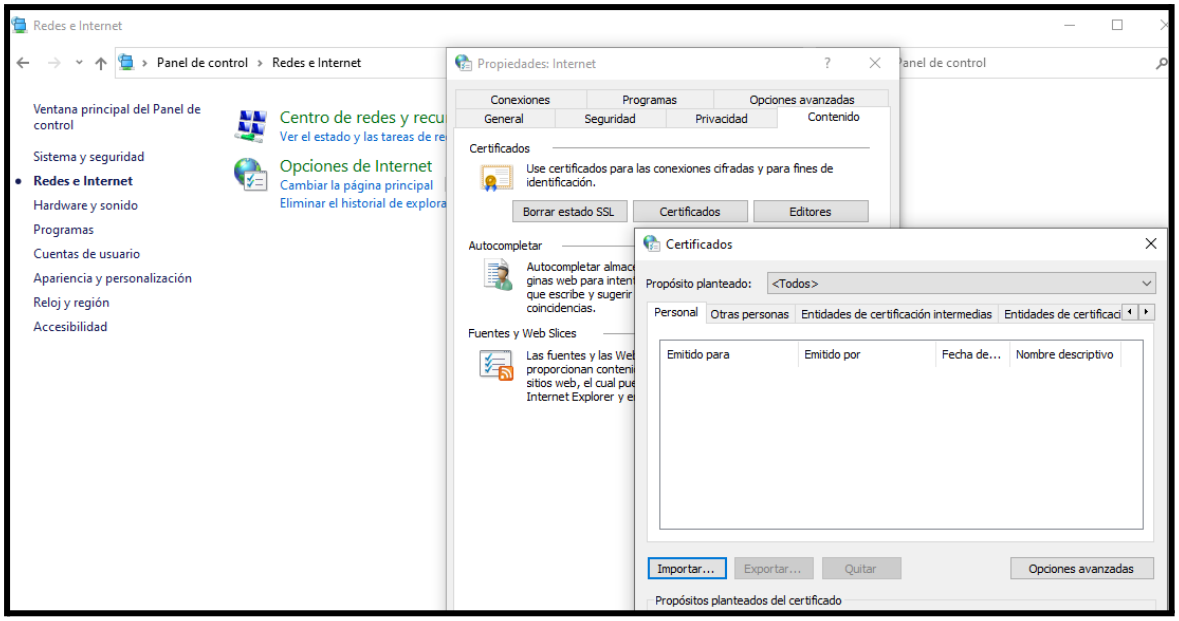


La web aparecerá como no segura en el navegador, podemos añadir el certificado en nuestro ordenador debemos realizar copiar el certificado desde la raspberry.
En la terminal de comando vamos al directorio .ssh y copiamos los archivos **ca.crt, server.key y server.crt.**

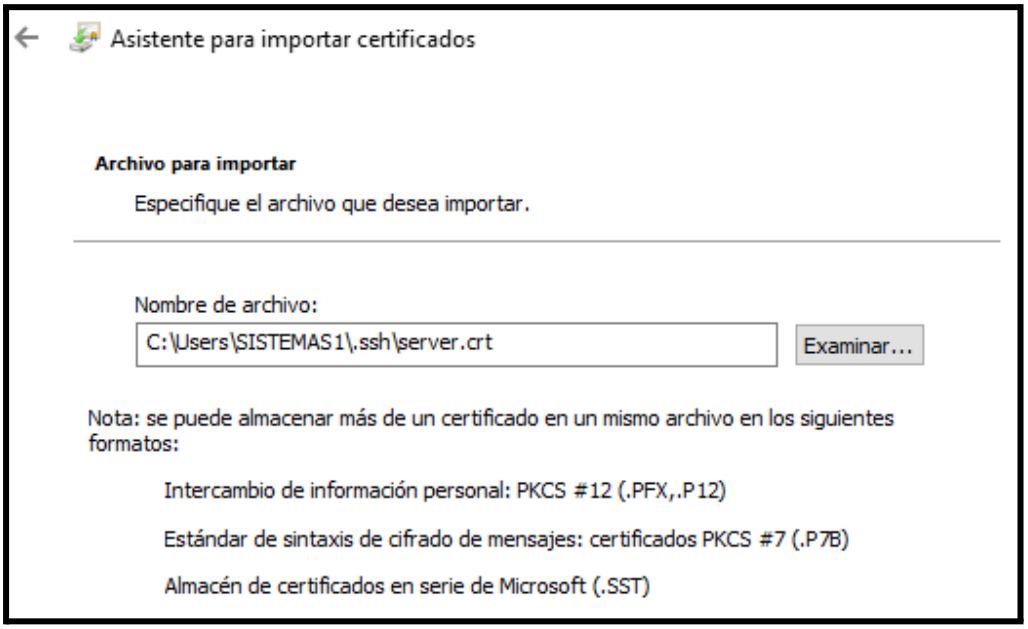
```
C:\Users\SISTEMAS1\.ssh>scp pi@10.40.0.154:/home/pi/Certificados/ca.crt ca.crt
pi@10.40.0.154's password:
ca.crt
```

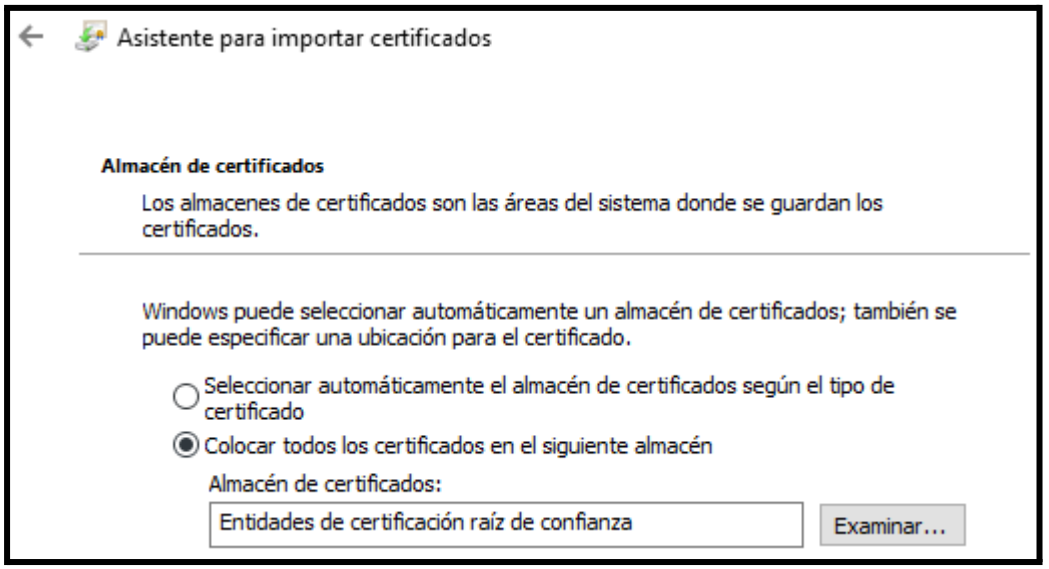
```
C:\Users\SISTEMAS1\.ssh>scp pi@10.40.0.154:/home/pi/.node-red/server.crt server.crt
pi@10.40.0.154's password:
server.crt
```

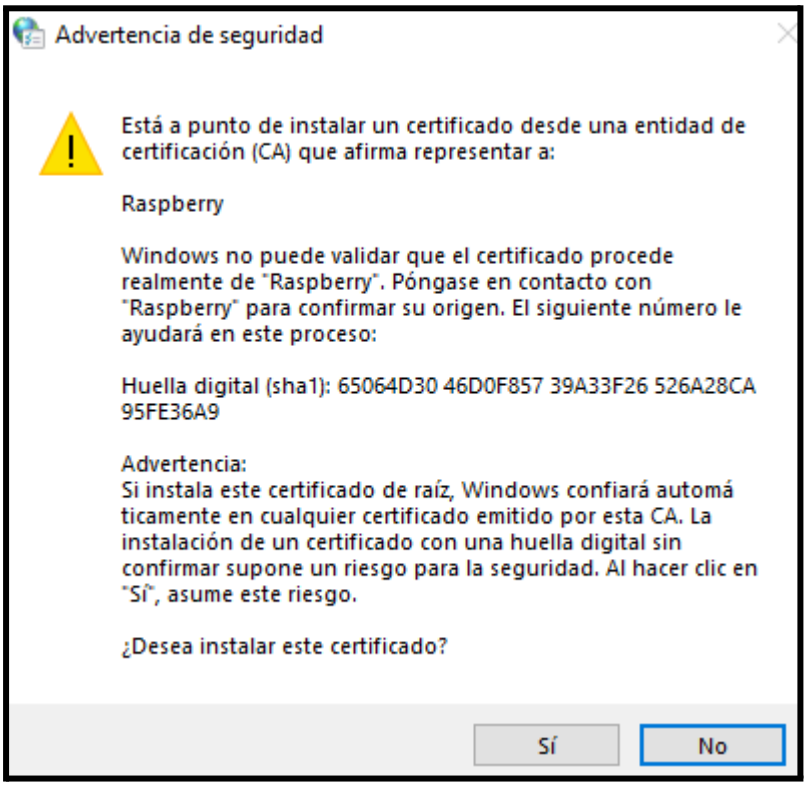
Desde windows, panel de control, opciones de internet, contenido, certificados, importar



Se nos abrirá el asistente de instalación de certificados, seleccionamos el archivo que hemos copiado desde la raspberry que contiene el certificado







Uso de la aplicación Node-Red

Una vez que Node-RED se está ejecutando, puede acceder desde:

- desde el navegador web de la propia Raspberry, localhost:1880
- desde cualquier otro equipo, https://{your_pi_ip-address}:1880

Al abrir la aplicación aparecen las siguientes secciones principales.

Los nodos son la unidad mínima que podemos encontrar en Node-RED. En la parte izquierda de la interfaz podemos ver la lista de nodos que vienen instalados por defecto y organizados en categorías según su funcionalidad.

Hay nodos de entrada, salida, funciones, para almacenar datos, etc… Esto muestra la capacidad de Node-RED de comunicarse con otros servicios.

Se pueden clasificar en tres tipos de nodos:

- Sólo admiten entradas: sólo admiten datos de entrada para ser enviados a algún sitio como pueda ser una base de datos o un panel de control.

- Sólo admiten salidas: son los nodos que sólo ofrecen datos tras recibirlos a través de diferentes métodos como por ejemplo un mensaje MQTT.

- Admiten entradas y salidas: estos nodos nos permiten la entrada de datos y luego ofrecen una o varias salidas. Por ejemplo, podemos leer una temperatura, transformarla en grados Celsius y enviarla a otro nodo.

- Los nodos los arrastramos al flujo (o flow, en inglés). Aquí es donde tendremos la lógica de nuestro programa mediante la concatenación de los diferentes nodos utilizados.

Podemos ver más información sobre lo que hace un nodo en el panel de información si hacemos click sobre cualquiera de ellos.

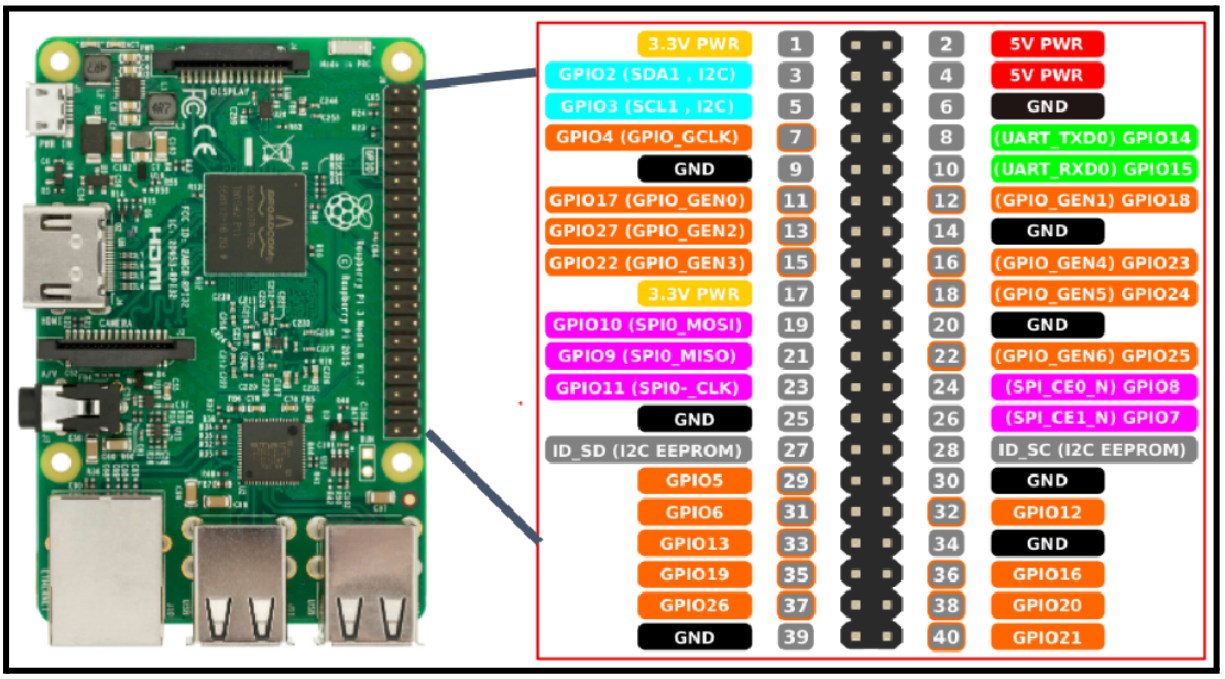
El panel de debug sirve para mostrar mensajes de lo que está ocurriendo dentro de cada flujo o tratamiento de datos.

Para guardar la aplicación sólo tienes que dar a Deploy, que pasará de color rojo a gris.

GPIO

Vamos a usar Node-RED para controlar el estado de los pines GPIO de Raspberry.

En Raspberry tenemos una fila de pines a lo largo del borde superior de la placa, estos pines son una interfaz física entre la Raspberry y el mundo exterior.



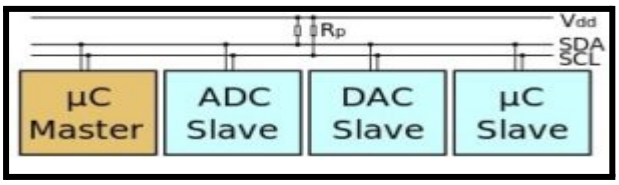
Según su función estos pines se clasifican como:

***Voltaje**
La Raspberry Pi puede proporcionar 5v **(pines 2 y 4)** y 3.3v (**pines 1 y 17**) de potencia. También proporciona una toma de tierra (**GND**) en los pines **6, 9, 14, 20, 25, 30, 34** y **39**.

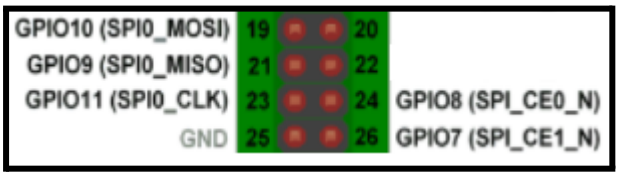
***UART, (color azul claro)** Utiliza los **pines 8 y 10 (GPIO 14 y 15)**, utilizados para comunicaciones serie a través del puerto serie RS232.

***I2C:** Inter-Integrated Circuit, (color gris claro) utiliza los pines **3 (GPIO 2)** y **5 (GPIO 3)** referenciados como SDA (datos) y SCL (clock)

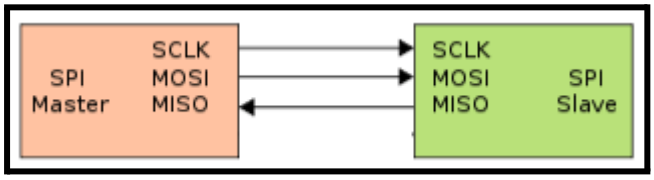
Bajo la arquitectura maestro/esclavo, y comunicación serie, es capaz de mantener tantos dispositivos diferentes como sean necesarios, siempre y cuando cada uno de ellos tenga una dirección única en el bus I2C.



***SPI:** Serial Peripheral Interface, (color naranja) utiliza Los **pines 19, 21, 23, 24, 25 y 26 (GPIO 10, 9, 11, 8, GND y GPIO 26)**



Similar al bus I2C utiliza 4 hilos (MOSI, MISO,SCLK; SS), permitiendo mayor flujo de datos y velocidad, aunque menor distancia entre los dispositivos y la raspberry



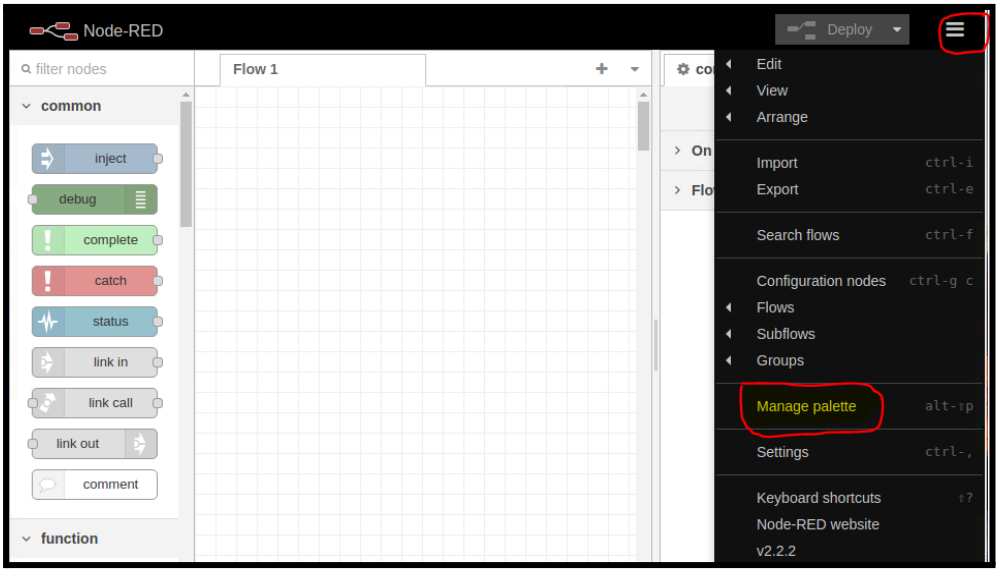
SCLK: Serial Clock (controlled by the main device)
MOSI: Master Out Slave In (data output from the main device)
MISO: Master In Slave Out (data output from controlled device)

***GPIO**
El otro gran bloque lo constituyen los pines denominado **GPIO** (General Purpose Input Output) (color amarillo) que pueden ser programados como entradas o salidas digitales con voltajes de 3.3/5v utilizados por ej para reconocer la pulsación de un botón (entrada) o encender un led (salida).

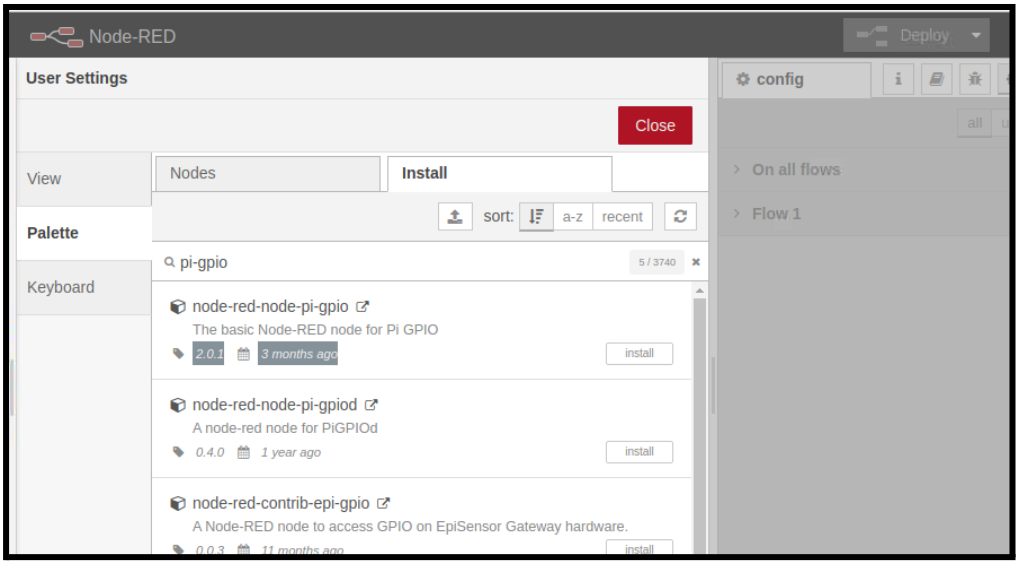
Estos pines son los que se utilizan para la mayoría de los proyectos iniciales.

Control de los pines GPIO con Node-Red

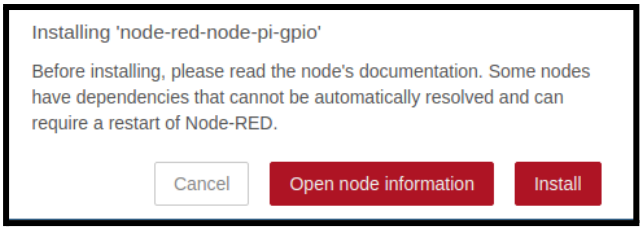
Para el control de los pines GPIO se necesita añadir un “nodo” nuevo que buscaremos y seleccionaremos desde la opción “manage palette”, es posible que ya estén instalados.



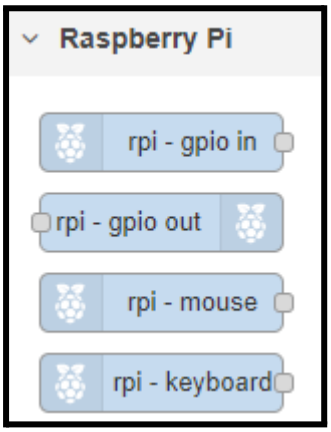
Marcamos la pestaña de Install y buscamos **pi-gpio**



Aparece una lista y seleccionamos el primero de ellos: **node-red-node-pi-gpio** e install.



Una vez instalado, en el menú izquierdo podemos encontrar el grupo GPIO bajo la etiqueta “Raspberry Pi” donde encontramos los nodos **rpi-gpio-in** (entrada) ,**rpi-gpio-out** (salida), además de **rpi-mouse** y **rpi-keyboard**.



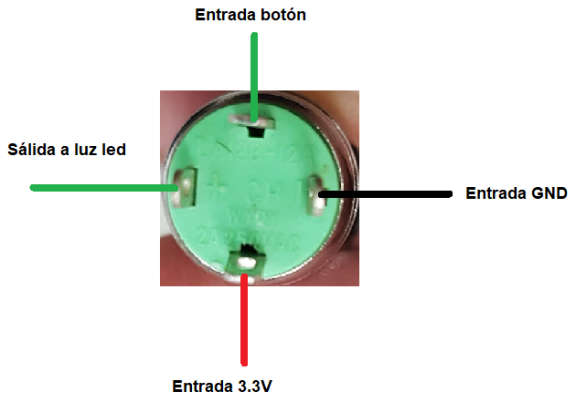
Los dos nodos con la etiqueta **rpi gpio** son los que usamos para hablar con los pines GPIO en la Raspberry.

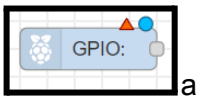
- **rpi-gpio in** es para entradas, usar un pulsador para controlar algo sería un ejemplo de una entrada.
- **rpi-gpio out** es para salidas, encender un LED sería un ejemplo de una salida.

Práctica, encendido led

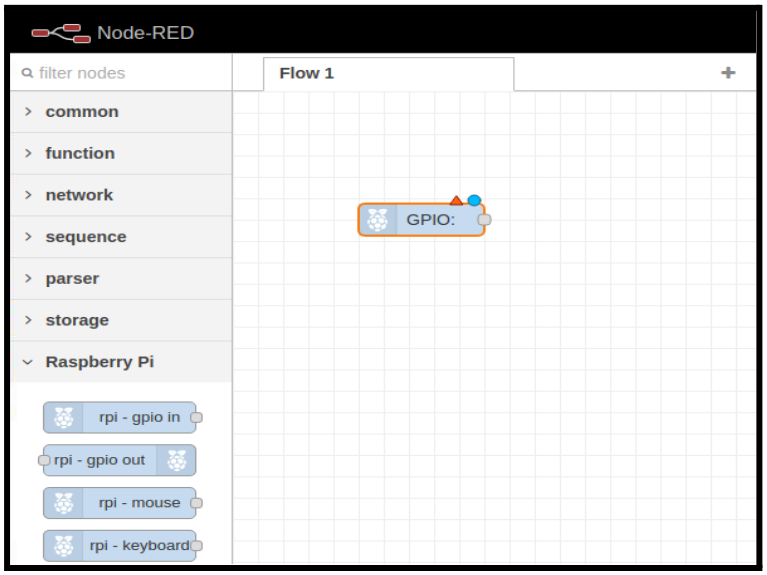
Vamos a comenzar programando el encendido de un led mientras pulsamos un botón.

Físicamente vamos a utilizar un modelo de micro pulsador que incorpora un led en su exterior cuyos contactos se utilizan tal y como se muestra en la imagen derecha.
El botón tiene un enlace para ver sus características.

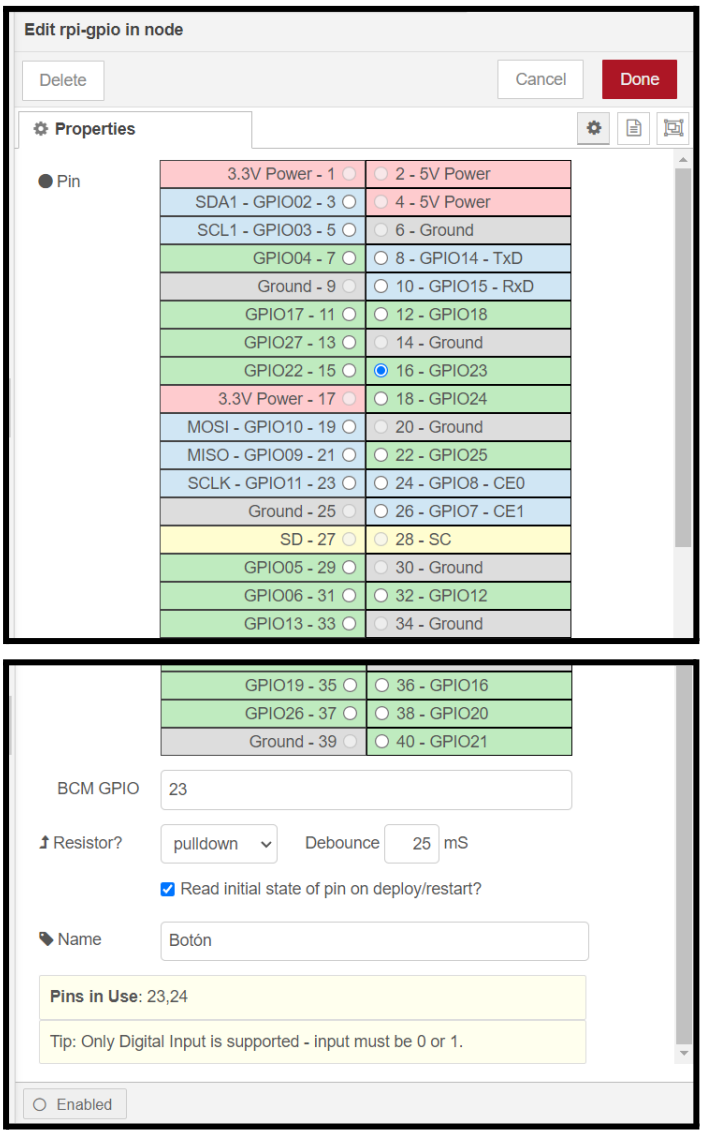


Arrastramos el nodo **rpi-gpio in** 

a la zona marcada como “flow”.



Haciendo doble clic en el nodo aparecerá un cuadro para configurar el nodo. Marcamos el pin GPIO que queremos utilizar. En el ejemplo utilizamos el pin 16, (GPIO23) y lo utilizaremos con entrada para conectar el botón.

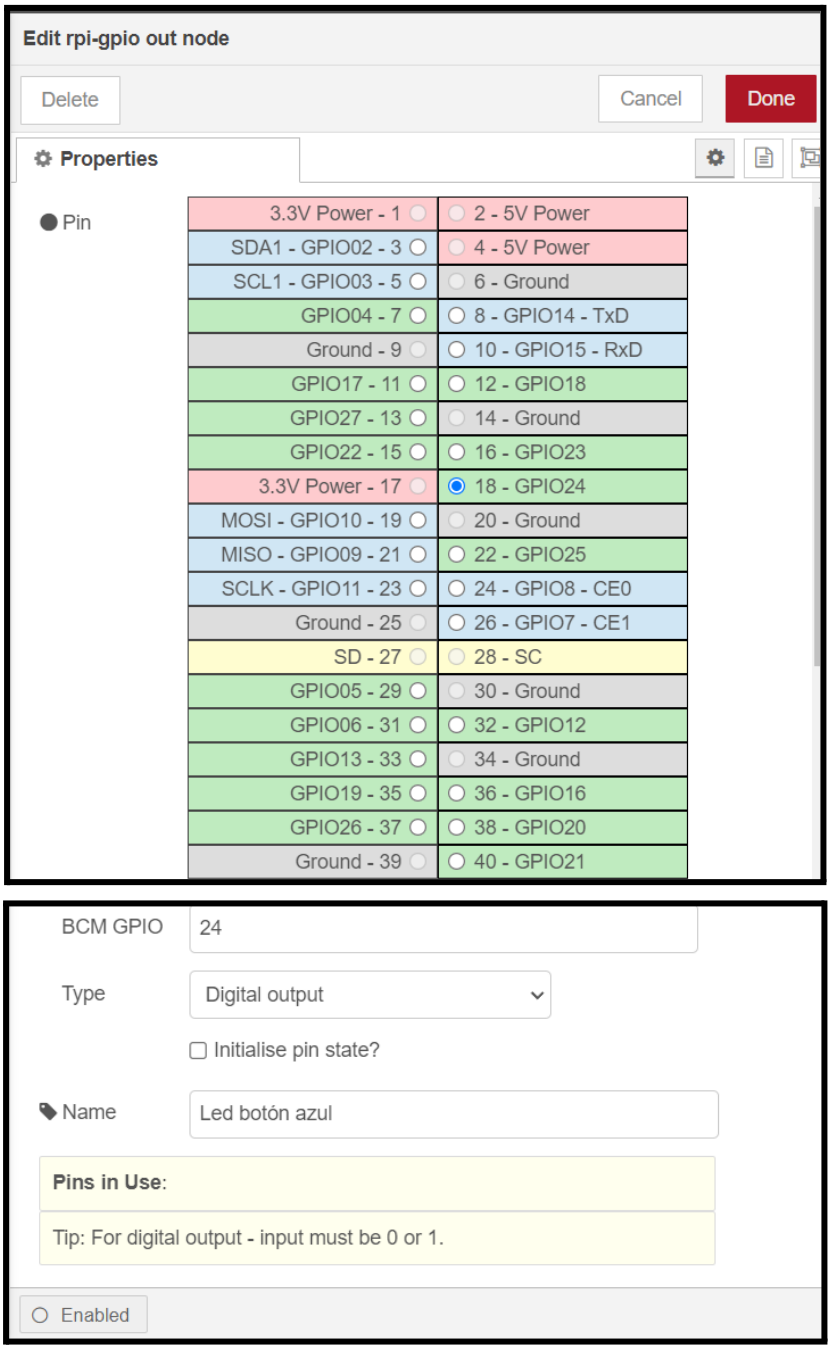


Hemos establecido el estado inicial al inicio en 0 (apagado), hemos denominado el nodo "Botón" y establecido el estado del botón en Pulldown que es un estado 0 de no presionado, al presionar el botón pasará a estado 1.

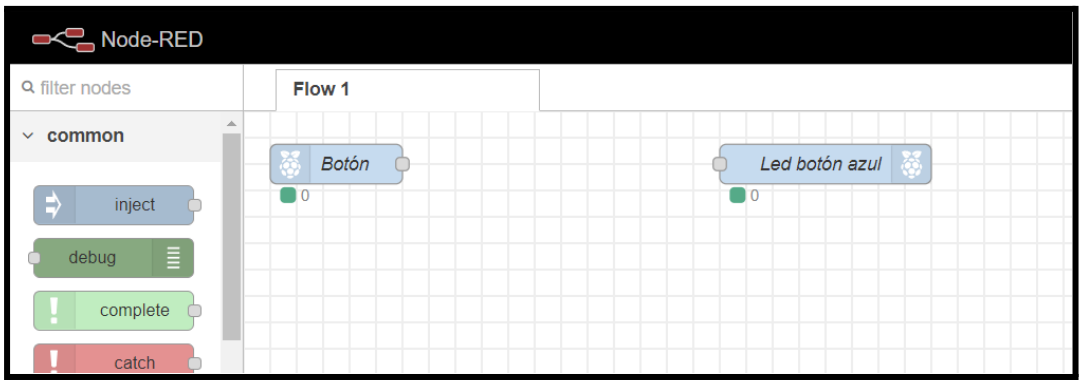
Arrastramos un nodo de entrada **rpi-gpio out** 

a la página flow y hacemos doble clic para editar.

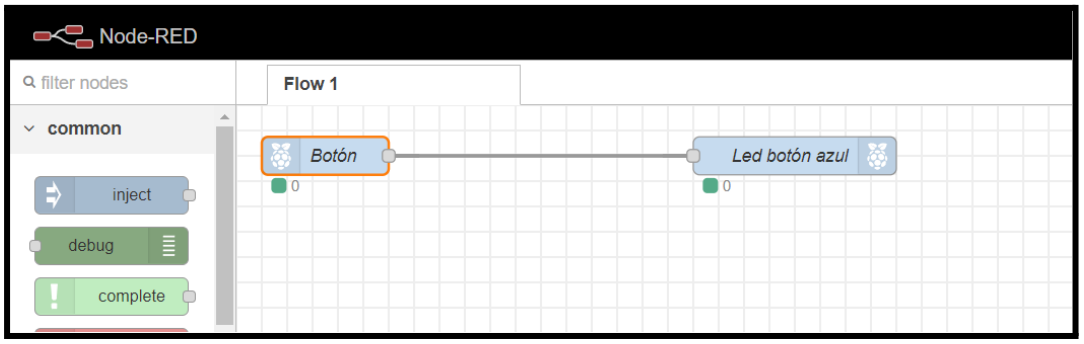
Marcamos el pin GPIO que queremos utilizar. En el ejemplo utilizamos el pin 18, (GPIO24) y lo utilizaremos con salida para el led.



Ya tenemos definidos el botón de entrada y la luz led de salida.

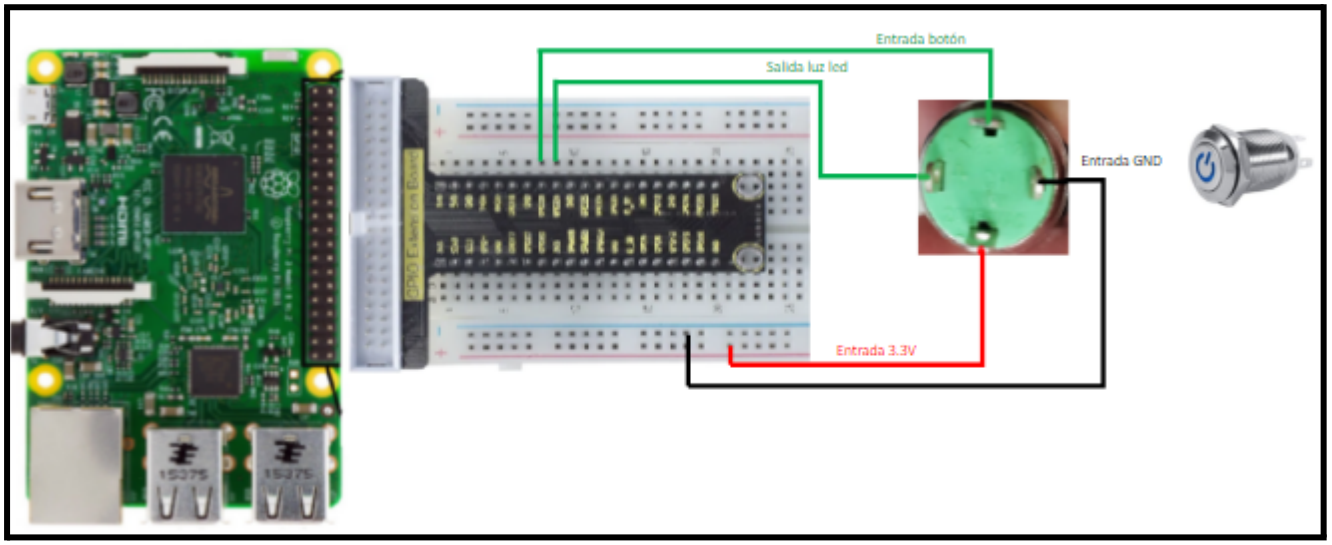


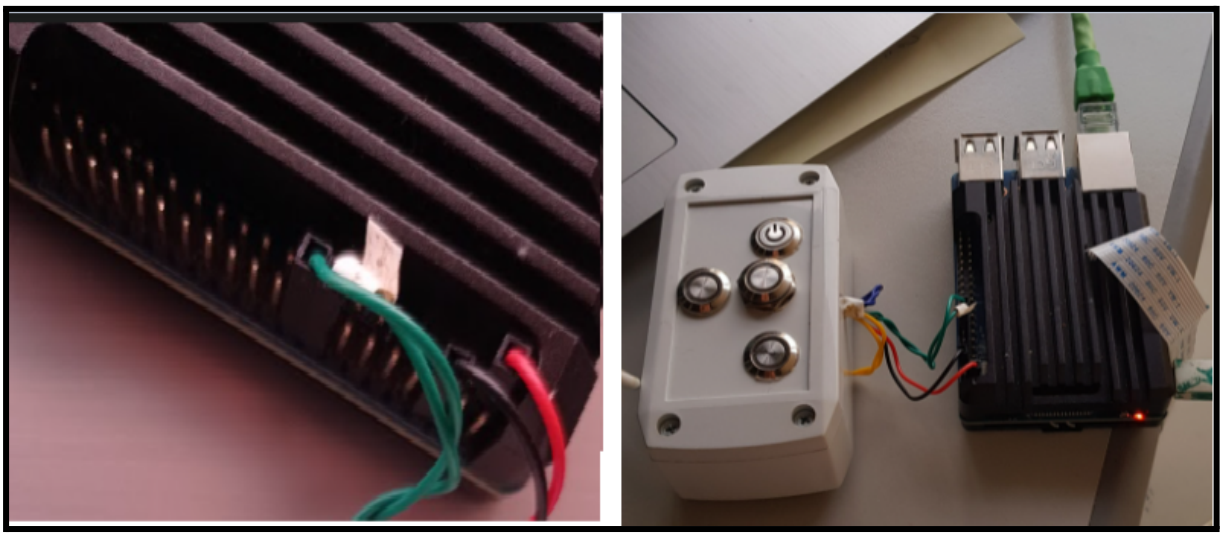
Uniremos el nodo de entrada “Botón” con el nodo de salida “Led” .



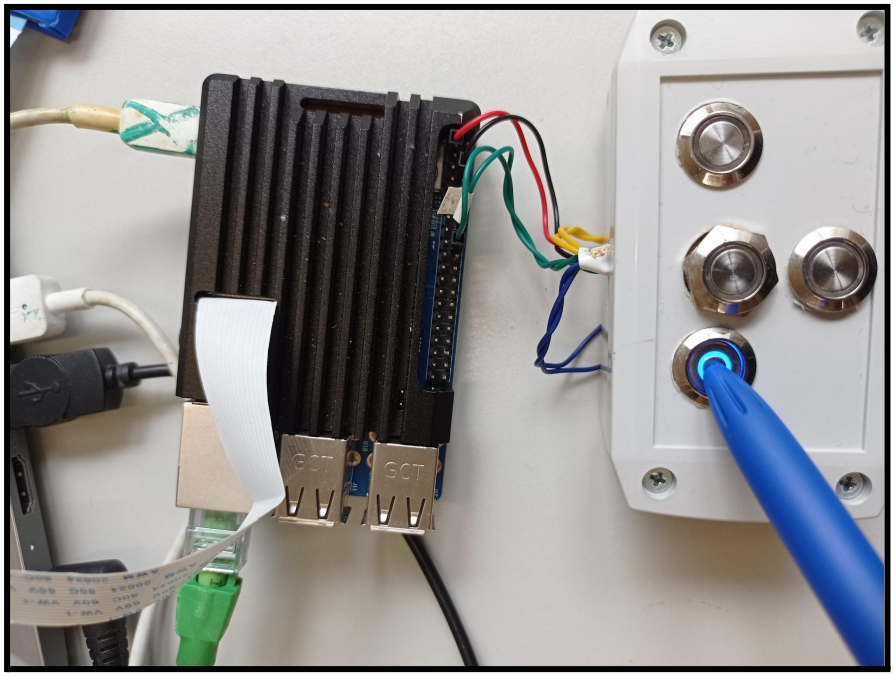
Para guardar el diseño sólo tienes que dar a **Deploy,** pasa de estar en un color gris oscuro a rojo.

Las siguientes imágenes siguientes muestran como se ha cableado la Raspberry y el botón/led

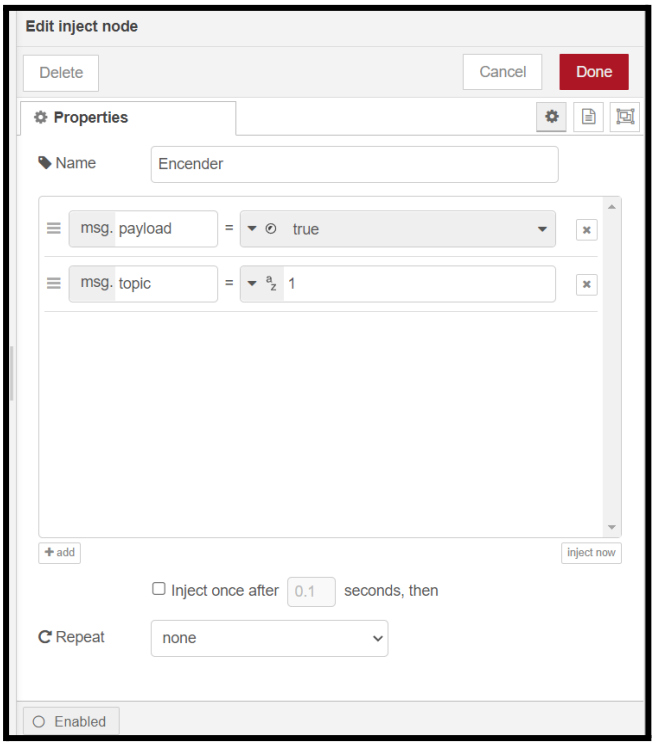




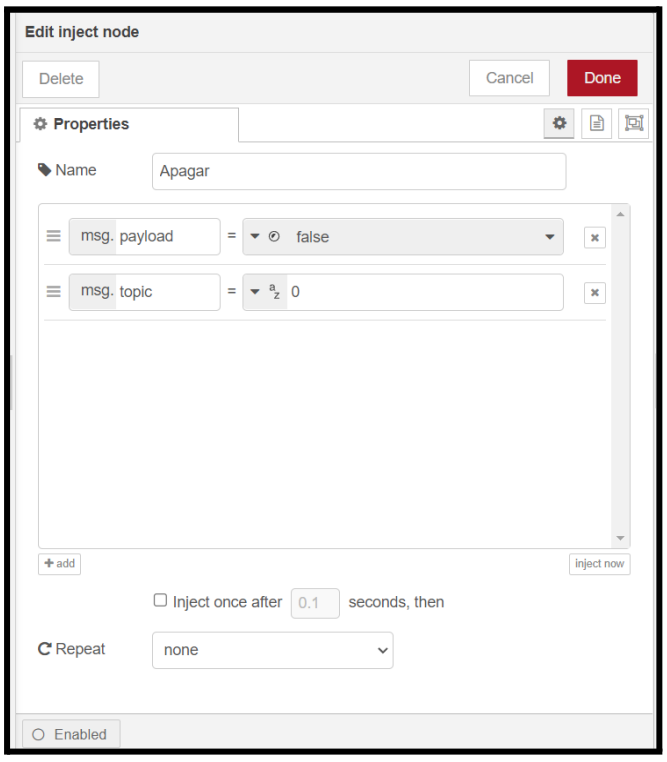
Al presionar el botón se enciende el led (azul) asociado al botón.

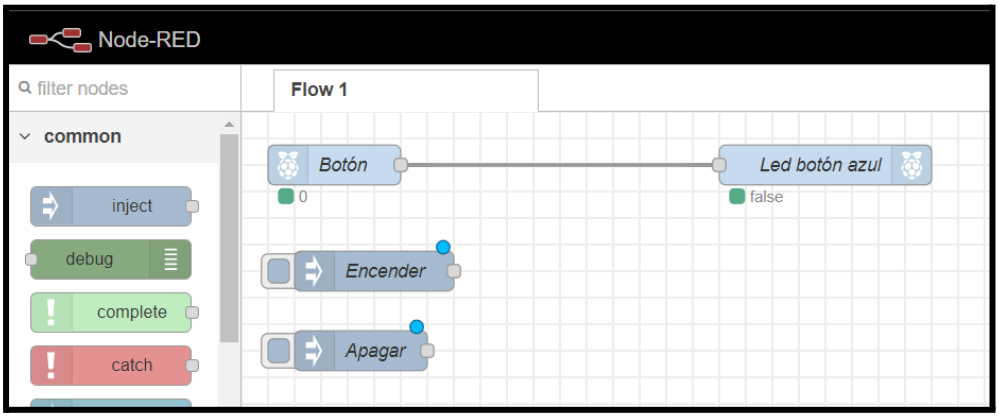


Se puede realizar el encendido y apagado del led desde un ordenador o panel HMI utilizando Node-RED sin necesidad de que nadie presione el botón. Para ello crearemos dos nodos inject, uno para encender y otro para apagar.

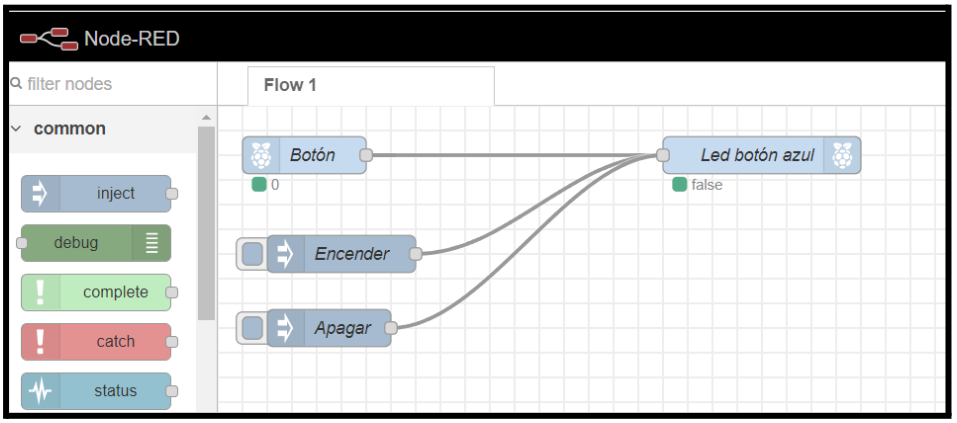


Inyecta un mensaje en un flujo de forma manual o a intervalos regulares. La carga útil del mensaje puede ser de varios tipos, incluidas cadenas, objetos JavaScript o la hora actual. Aquí definimos los nodos como booleanos y encender tendrá valor TRUE/1 y apagar FALSE/0.





Ahora debemos unir estos dos nodos con la salida Led

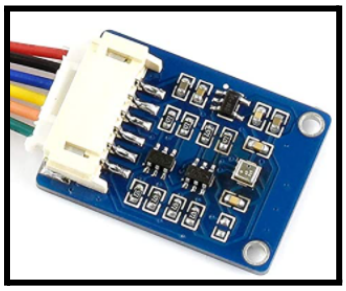


Ahora desde nuestro ordenador a modo de panel HMI podemos igualmente encender y apagar el botón.

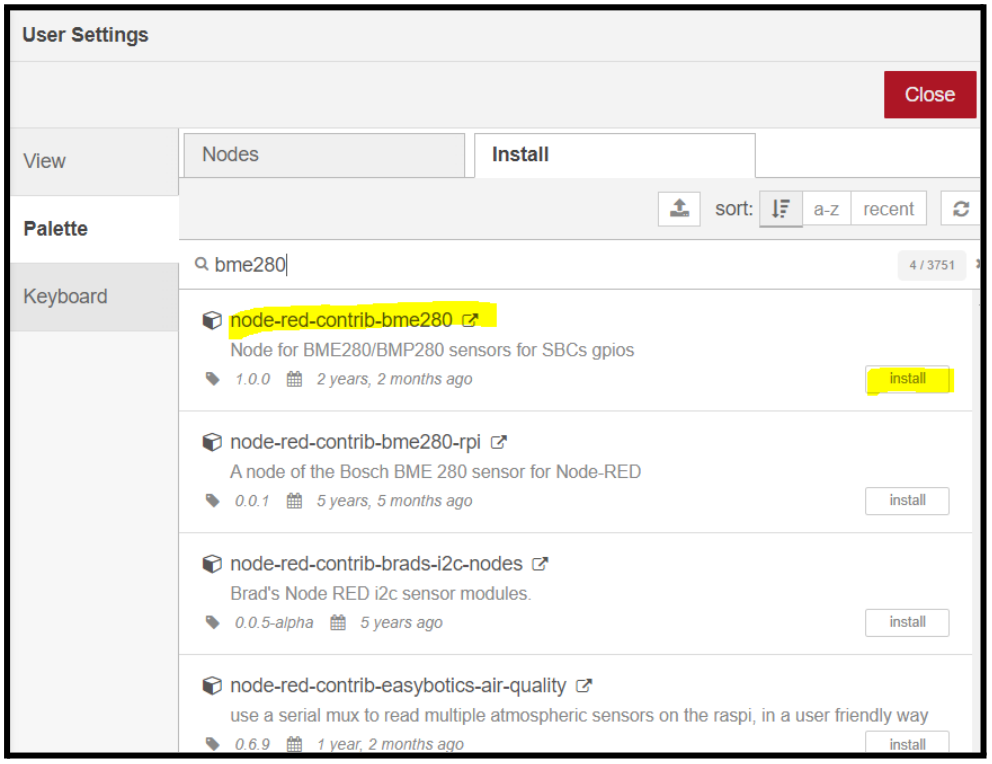
Práctica, lectura sensor temperatura BME280

Utilizaremos un sensor de temperatura del que que exista su módulo en Node-RED para facilitar la programación.

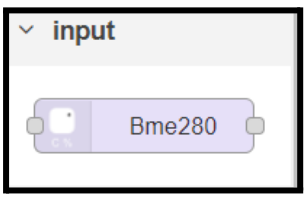
Elegimos uno BME280, en la foto dejo enlace para ver sus características.



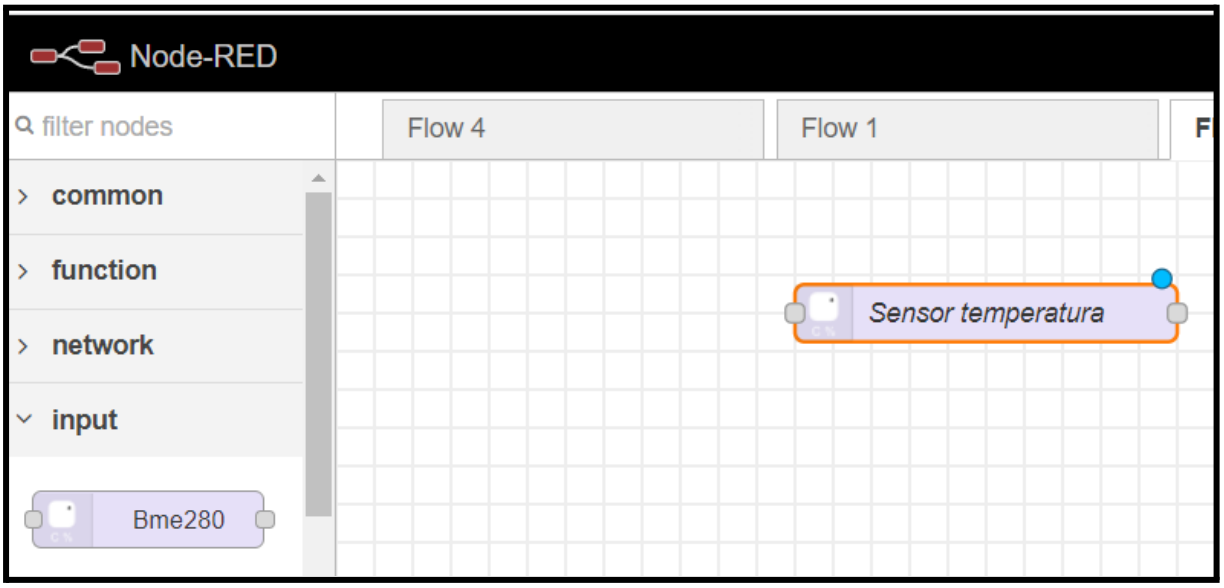
En Node-RED iremos a Manage Palette y en la pestaña Install buscamos **bme280**. Seleccionamos el más actual e instalamos,

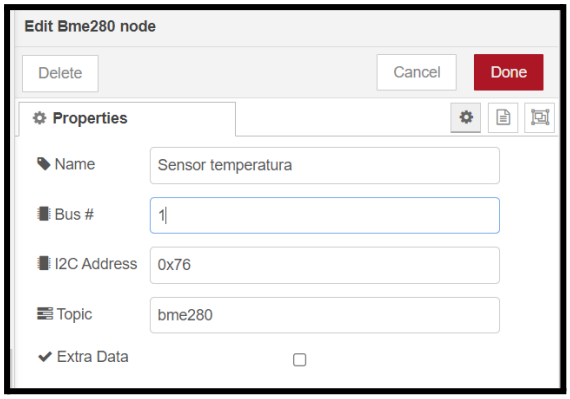


Ya tenemos instalado el nodo Bme280 para poder leer datos ambientales de sensores BME280/BMP280.



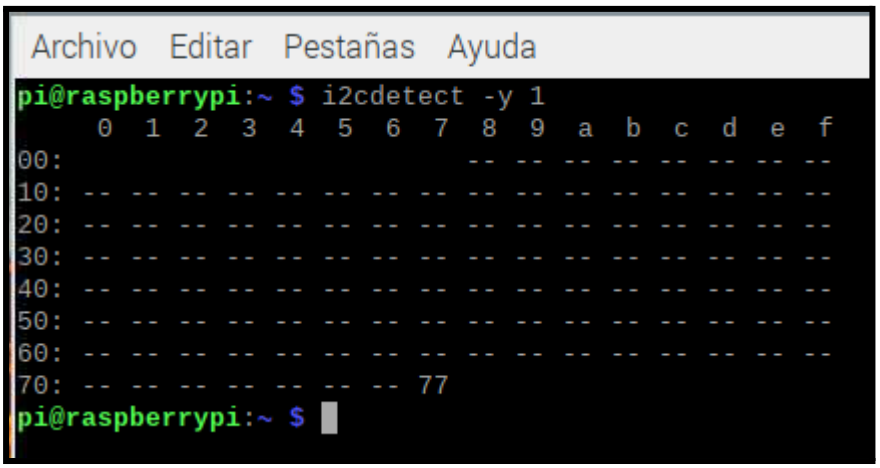
Marcamos, lo insertamos y damos doble clic para propiedades.





Ponemos un nombre y en Bus el número de bus que utilizaremos en la raspberry.
I2C Address, por defecto viene con la 76 aunque dependiendo de las características del sensor esta dirección puede ser modificada. Si tuviésemos más sensores de este tipo tenemos que cambiar la dirección ya que no se pueden repetir la dirección.

Si no sabemos la dirección, una vez pinchado el sensor en la raspberry, ejecutamos **i2cdetect -y 1** (donde 1 es el número del bus).
En el ejemplo comprobamos que el sistema devuelve el valor **77**



Modificamos por tanto el nodo del sensor y cambiamos la dirección a 77.

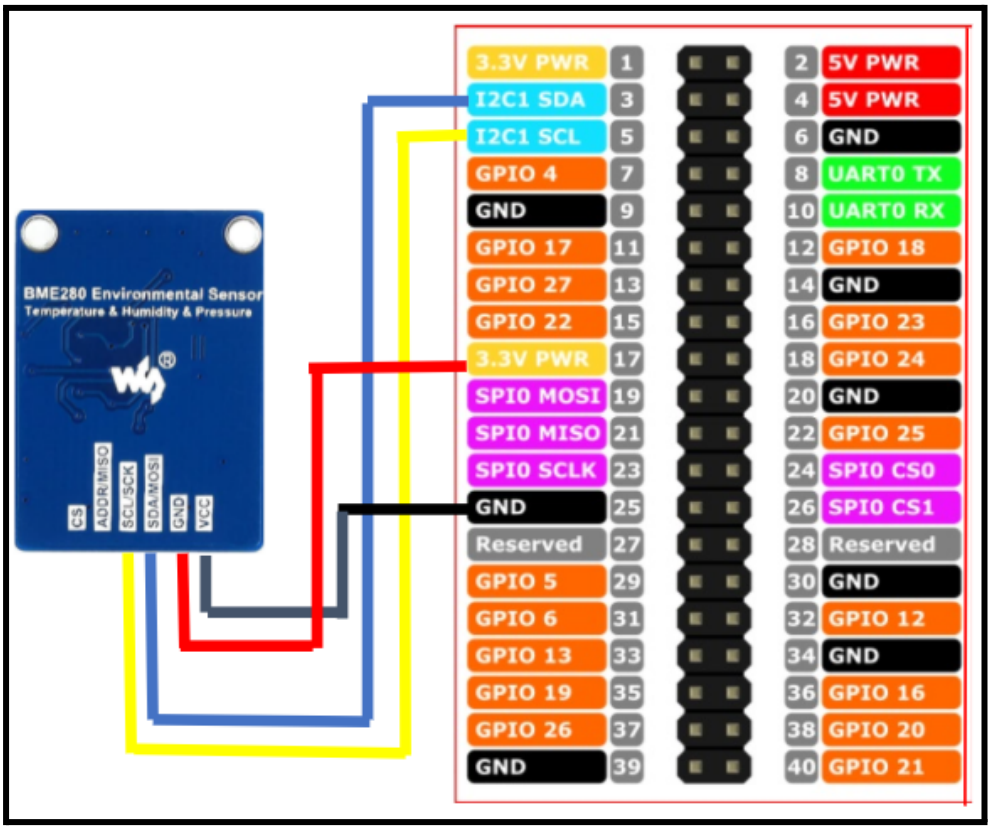
En las características del sensor vemos que admite I2C (predeterminado) con dirección esclava configurable y SPI para comunicarse con la raspberry.

I2C

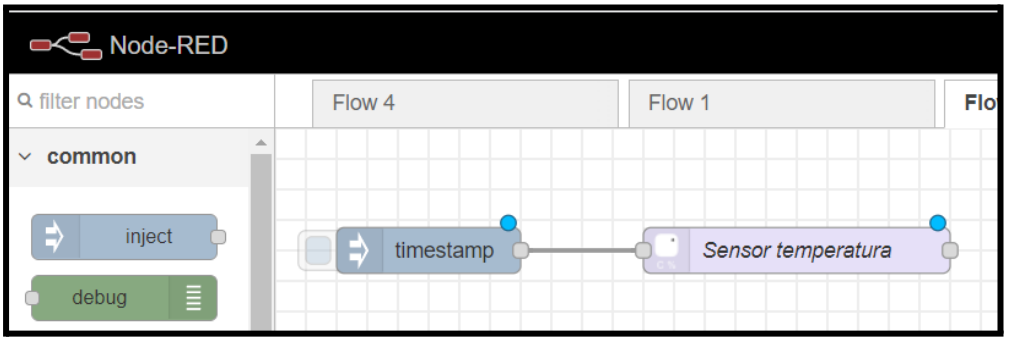
I2C es un bus de comunicación muy sencillo con solo dos hilos, una línea de datos (SDA) y una línea de reloj (SCL).

Conectamos el sensor a las raspberry en el pin 3,**GPIO02,** el cable SDA y en el pin 5, **GPIO03,** el cable SLC.

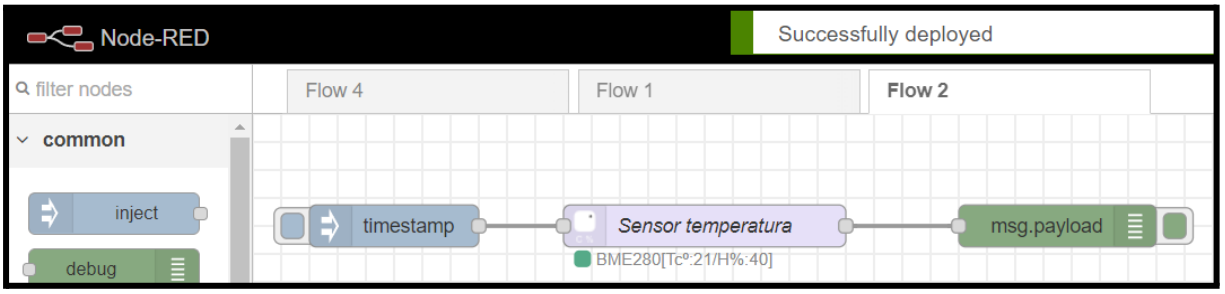
Solo se necesitan cuatro cables. dos para alimentación (VCC 3.3V y GND) y dos para transmitir datos (SLC y SDA).



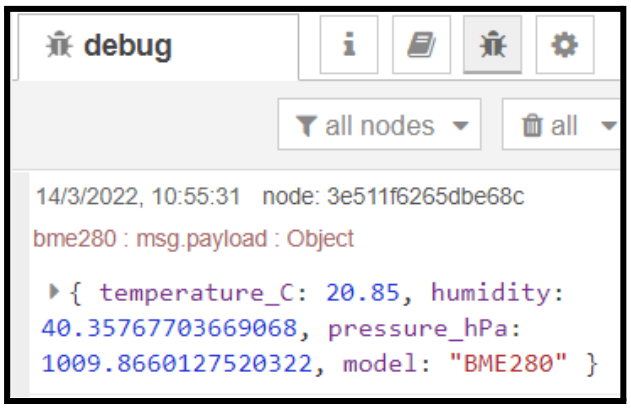
Para que el BME280 envíe datos a intervalos, colocamos un nodo de inyección desde la pestaña de entrada en el flujo de trabajo.



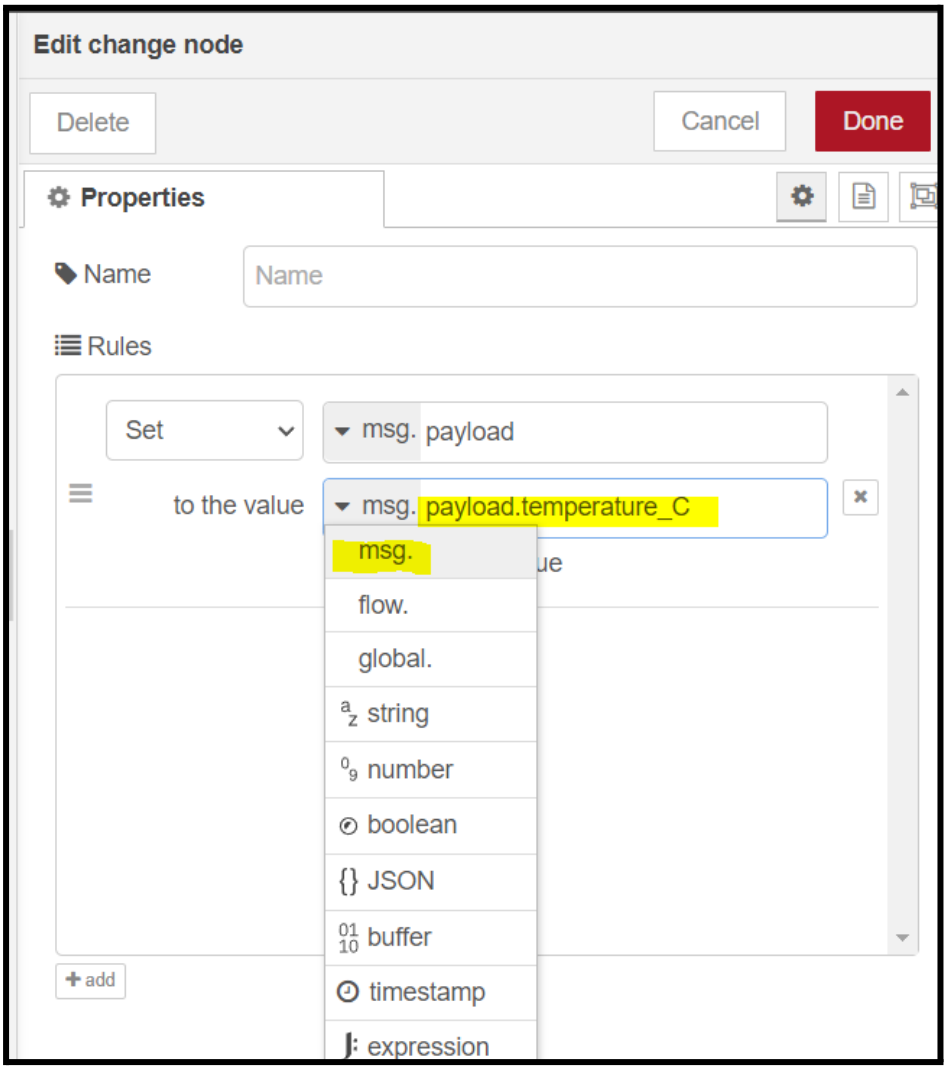
Añadimos un nodo de debug para visualizar los datos.

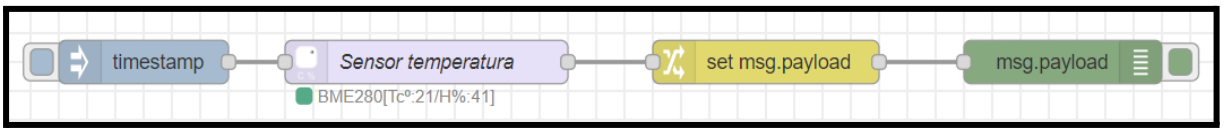


Cada vez que pulsemos en el nodo de inyección veremos los datos del sensor en la pestaña de depuración.

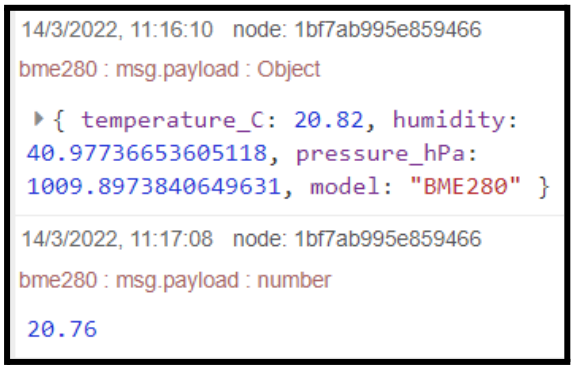


Si queremos usar un campo específico (como la temperatura) sin traer todo lo demás, tendremos que separarlo. Usaremos un nodo de **cambio**,  conectado entre el nodo bme y el nodo debug.
En las propiedades seleccionamos de tipo msg y en valor ponemos el nombre de la variable que vemos en el debug.

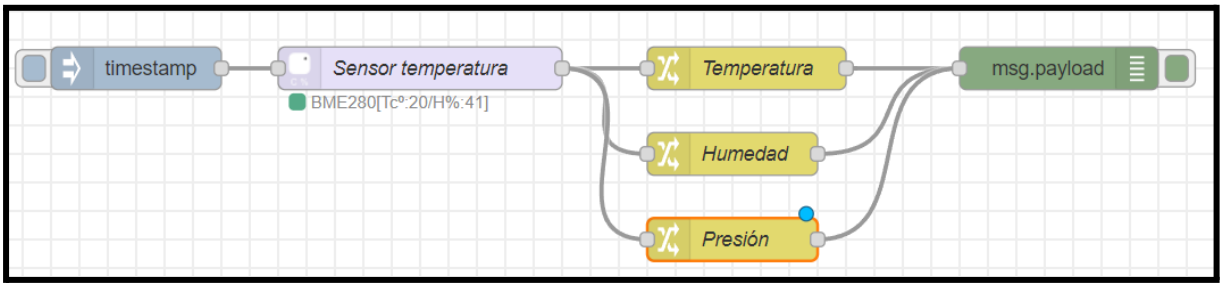




En depuración vemos el valor de la temperatura



Creamos otros nodos para la humedad y para la presión.



Ahora en debug vemos los tres datos del sensor por separado.

Introducción a la programación LADDER en Node-RED

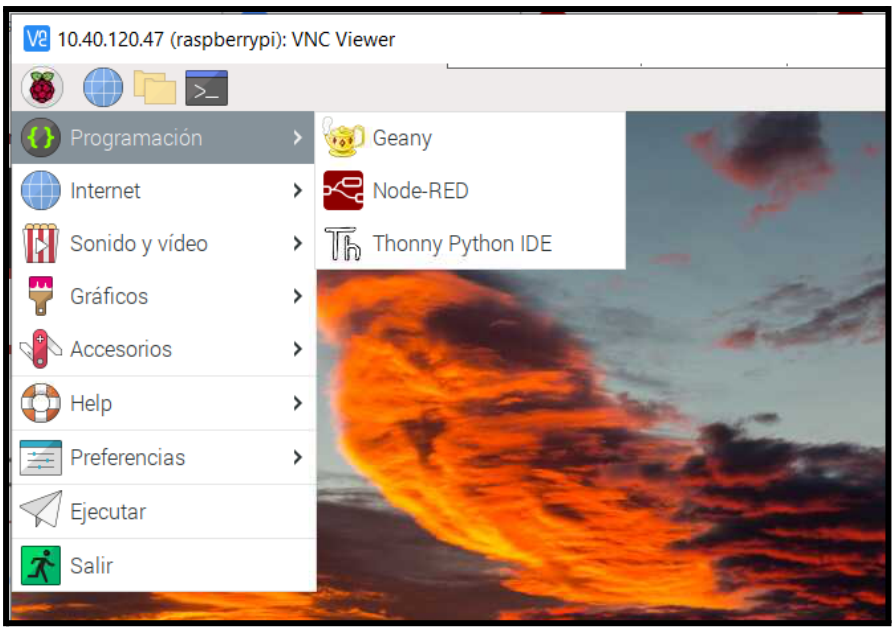
Los nodos **redPlc** implementan la funcionalidad del software PLC en Node-Red, utilizan el entorno gráfico de Node-Red para programación Ladder.

Si deseas introducirte un poco en la programación Ladder pulsa en el enlace.

Instalación

Necesitamos añadir dos nodos en Node-RED.

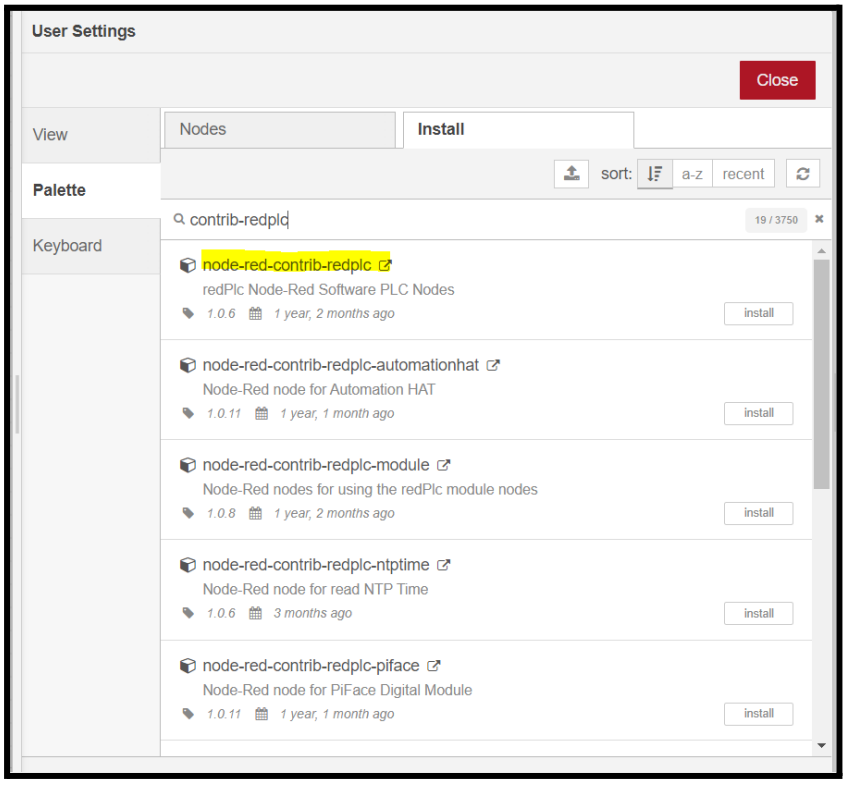
Iniciamos Node-RED en Raspberry



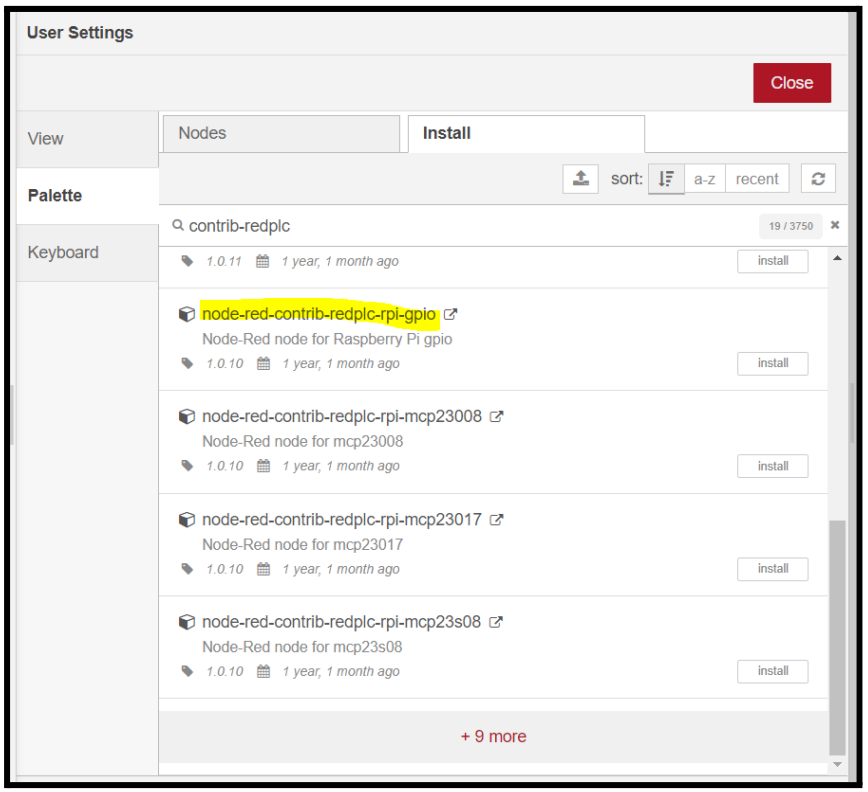
Accedemos desde nuestro ordenador https://IP(Raspberry):1880

Marcamos el menú general y pulsamos Manage Palette.

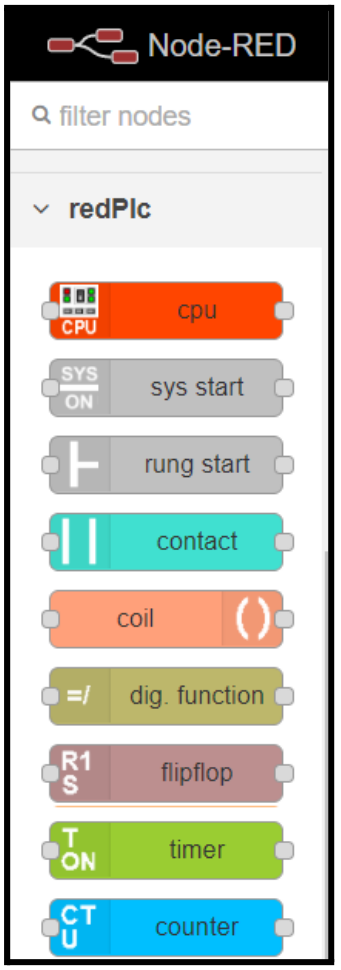
Seleccionamos la pestaña Install, buscamos contrib-redplc y pulsamos Install en el nodo **node-red-contrib-redplc**



Seguido buscamos el nodo **node-red-contrib-redplc-rpi-gpio** lo seleccionamos e instalamos.



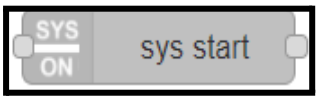
Ya instalados los dos paquetes de nodos vemos en la pantalla inicial a la izquierda los nuevos nodos.



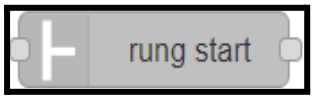
Uso

Los nodos del módulo asignan hardware o datos de comunicación a variables globales. Los nodos del módulo deben instalarse según el hardware utilizado o la comunicación. Las variables globales usan nombres y formatos únicos predefinidos. Cada variable es única con el número de dirección subsiguiente.

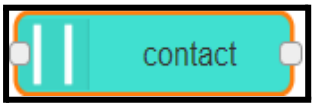
Realiza un orden secuencial cíclico. Los nodos del módulo están cableados a la primera salida. Los nodos de lógica de escalera están conectados al segundo y siguientes salidas (peldaños).



Envía un mensaje al inicio. Se puede utilizar para inicializar.



Muestra un riel de alimentación de escalera. El estado muestra el número de peldaño o el comentario. Conecte este nodo a las salidas de cpu.



Contact: implementa un contacto de escalera. Si está cerrado, la potencia de entrada (verdadera o falsa) se envía a la salida. Si está abierto, no se envía energía (falso) a la salida. El cableado del nodo de contacto en serie funciona como lógica AND. El cableado del nodo de contacto en paralelo funciona como lógica OR. Cualquier combinación lógica de cableado es posible.

Implementa una bobina de escalera. Energizado si la condición de entrada es verdadera. Desenergizado si la condición de entrada es falsa.

Implementa funciones digitales. Estos son NOP, NOT, OR, AND y XOR. P_TRIGGER y N_TRIGGER para detección de bordes.

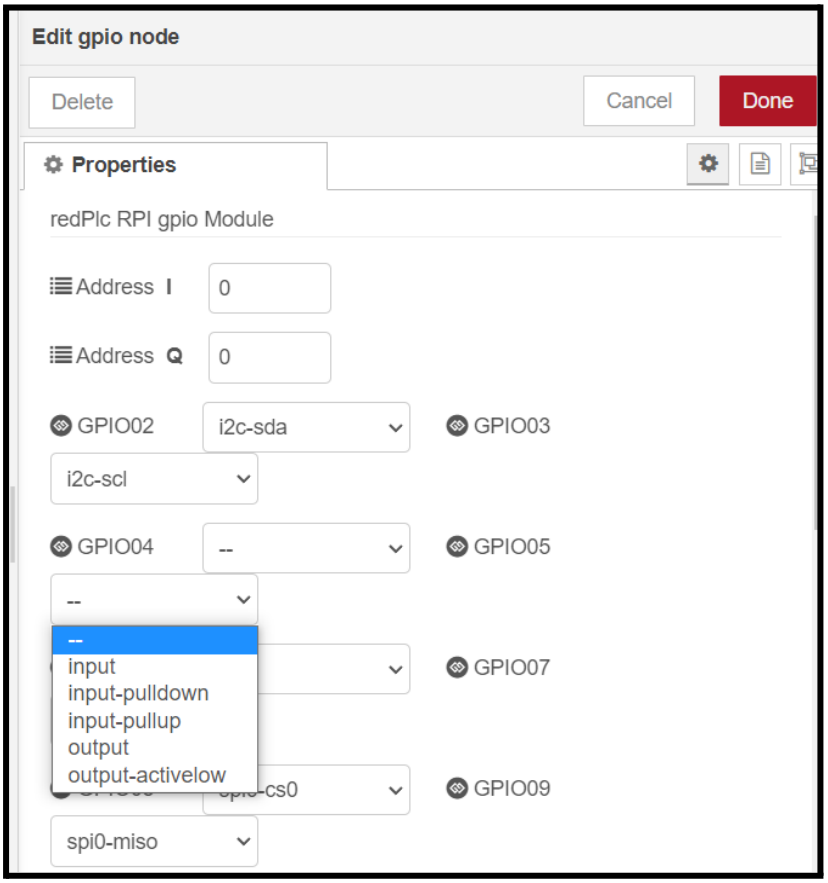
Establece la salida con restablecimiento dominante.

TON se retrasa. TOF se retrasa. TP produce un pulso de encendido/apagado. TPI produce un intervalo de pulso de encendido/apagado. TONR es un temporizador remanente.

CTU cuenta hacia arriba. CTD cuenta atrás. CTUD cuenta arriba/abajo.

Cada pin gpio se puede configurar como entrada o salida.

Si pulsamos doble click aparecen las propiedades.



La “payload” I, los pines gpio se establecen como entrada y se actualizan I bits de variables globales.

En la “payload” O, los pines gpio se establecen como salida y se actualizan a partir de Q bits de variables globales.

Al inicio, se crean variables globales.

Modos GPIO:

- --: El pin está intacto.
- **input**: entrada flotante. Debe conectarse a 3.3V o tierra.
- **input-pulldown**: Entrada con pulldown (resistencia). “True” si está conectado a 3.3V.
- **input-pullup**: Entrada con pullup (resistencia). “True” si está conectado a tierra.
- **output**: Salida. Está conectado a 3.3V en “true”.
- **output-activelow**: Salida activa baja. Está conectado a tierra en “true”.

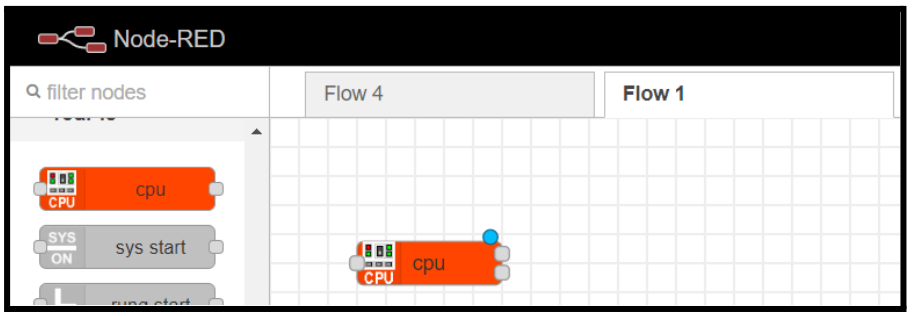
Este nodo sólo funciona en Raspberry Pi. Compruebe si los pines están configurados como I2C, SPI o UART. !! Más de 3,3 V daña la entrada GPIO. !! La sobrecorriente daña la salida GPIO.

Bit	Funct.	Pin	Pin	Funct.	Bit
	3V3	1	2	**5V**	
2	GPIO02	3	4	**5V**	
3	GPIO03	5	6	**GND**	
4	GPIO04	7	8	GPIO14	14
	GND	9	10	GPIO15	15
17	GPIO17	11	12	GPIO18	18
27	GPIO27	13	14	**GND**	
22	GPIO22	15	16	GPIO23	23
	3V3	17	18	GPIO24	24
10	GPIO10	19	20	**GND**	
9	GPIO09	21	22	GPIO25	25
11	GPIO11	23	24	GPIO08	8
	GND	25	26	GPIO07	7
		27	28		
5	GPIO05	29	30	**GND**	
6	GPIO06	31	32	GPIO12	12
13	GPIO13	33	34	**GND**	
19	GPIO19	35	36	GPIO16	15
26	GPIO26	37	38	GPIO20	20
	GND	39	40	GPIO21	21

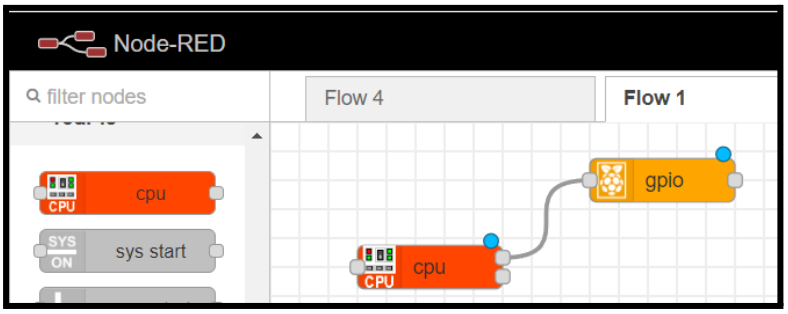
Práctica, encendido/apagado de led con botones

Como ya vimos en la programación Ladder vamos ha realizar el encendido y apagado de una luz con dos botones.

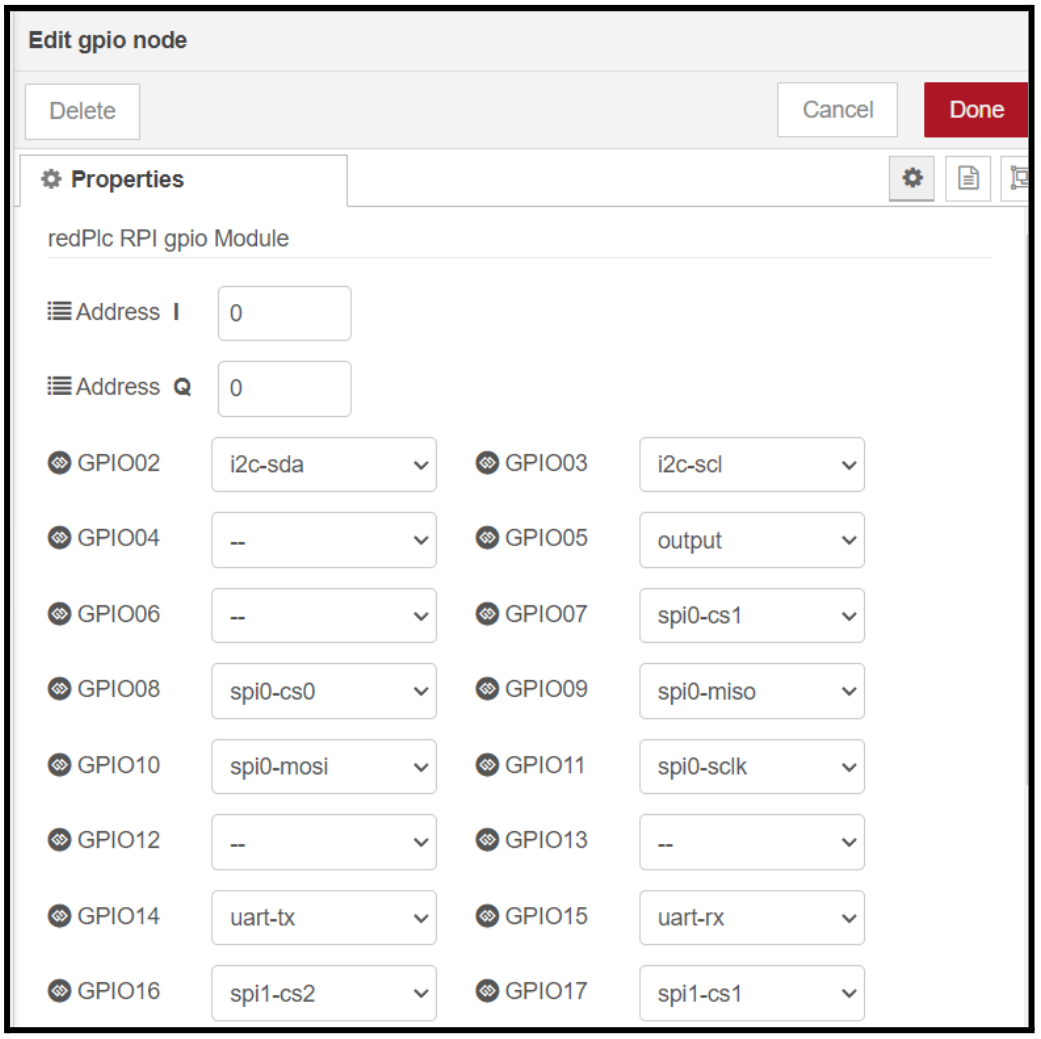
Lo primero que insertamos es el nodo CPU

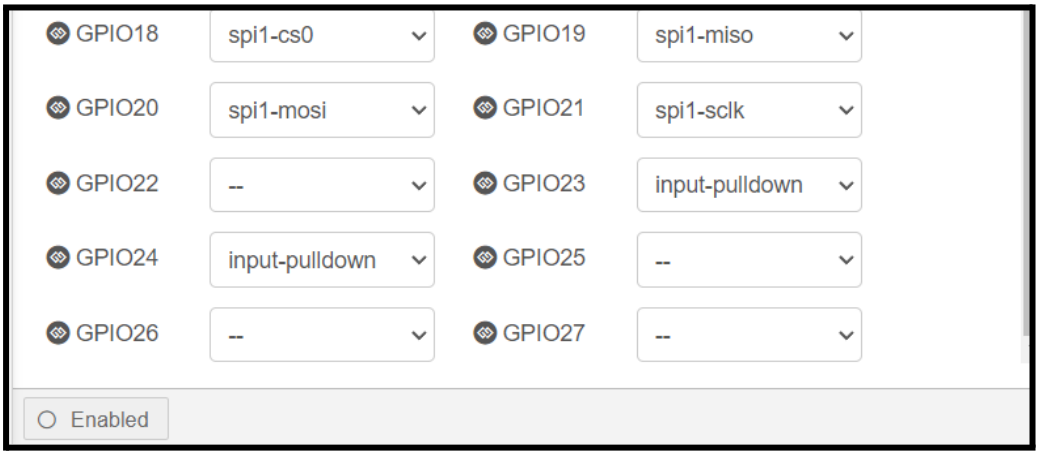


Añadimos el nodo **gpio,** unimos a la cpu y pulsando doble click aparecen las propiedades.



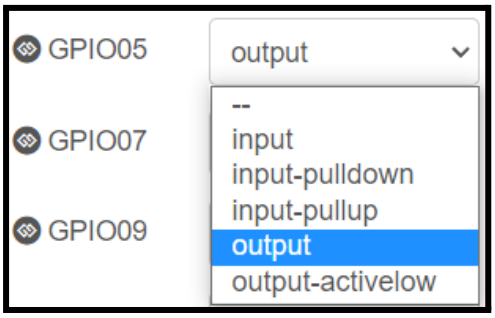
En propiedades definimos los GPIO que vamos a utilizar y sus características. Utilizamos GPIO que no estén reservados para spi, uart, spi..



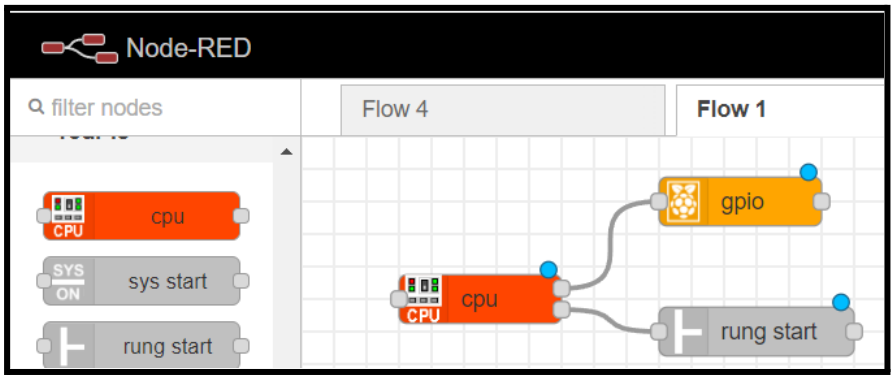


Para el ejemplo seleccionamos los pines **GPIO23** y **GPIO24** como entradas

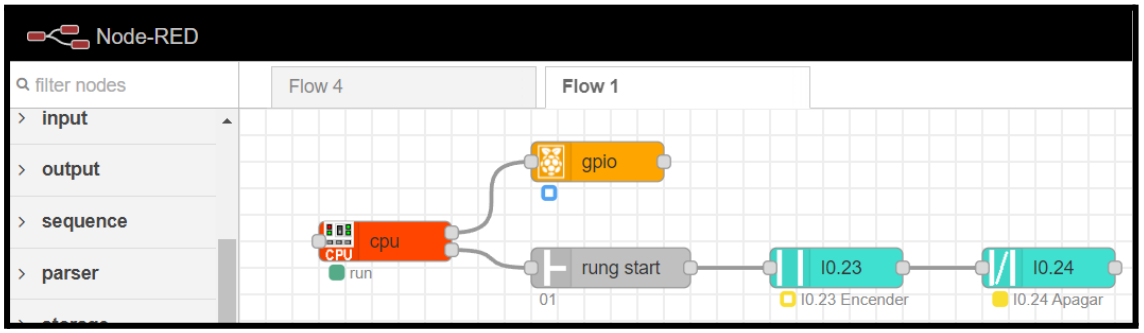
Una vez seleccionado el GPIO a utilizar debemos definir el tipo. Tenemos en el ejemplo dos botones de entrada tipo pulldown, encender y apagar. Otro GPIO será de salida, es la luz.



Añadimos un riel de alimentación, **rung start,** y lo unimos a la salida de cpu.



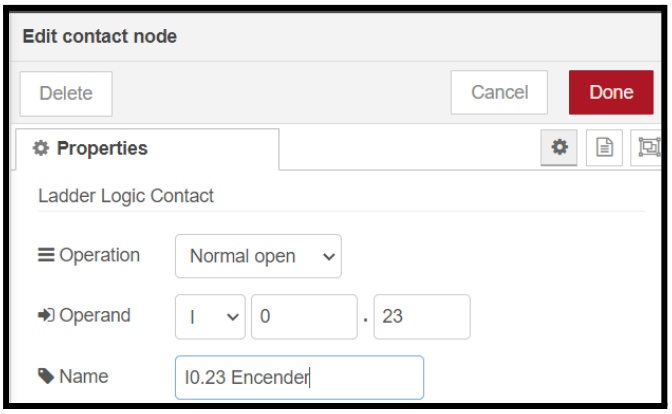
Añadimos los dos contactos, uno abierto (Encender) y otro cerrado (Apagar). Estos son de entrada de tipo pulldown.



Pulsaremos doble clic sobre cada uno de ellos para definir sus propiedades.

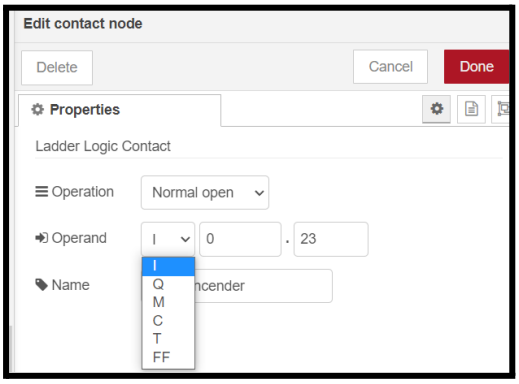
En **Operación** tenemos 4 posibilidades:

- Normalmente abierto: Contacto cerrado en bit verdadero.
- Normalmente cerrado: Contacto cerrado por bit falso.
- Borde positivo: Contacto cerrado en el borde de subida.
- Borde negativo: Contacto cerrado en el borde descendente.



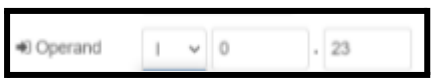
En **Operand** definimos el tipo de variable que puede ser:

- I, entrada
- Q, salida
- M, memoria
- C, contador
- T, reloj
- FF, flip flop



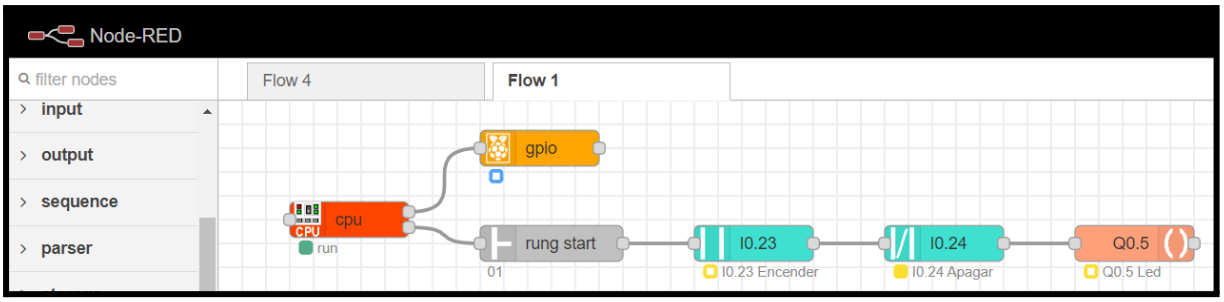
En el siguiente número definimos la dirección de entrada o salida del módulo GPIO que tiene que ser igual a cómo lo definimos en el nodo **rpi GPIO.**

El último número a introducir es el GPIO utilizado, en este caso 23.

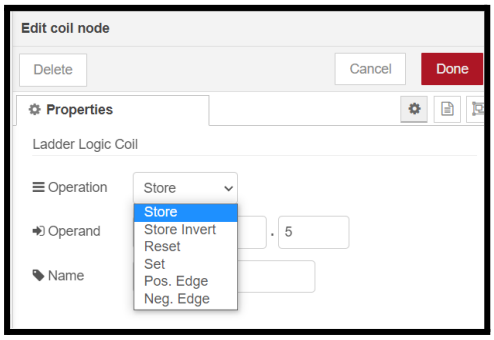


Asignaremos un nombre que identifique el gpio utilizado y su función.

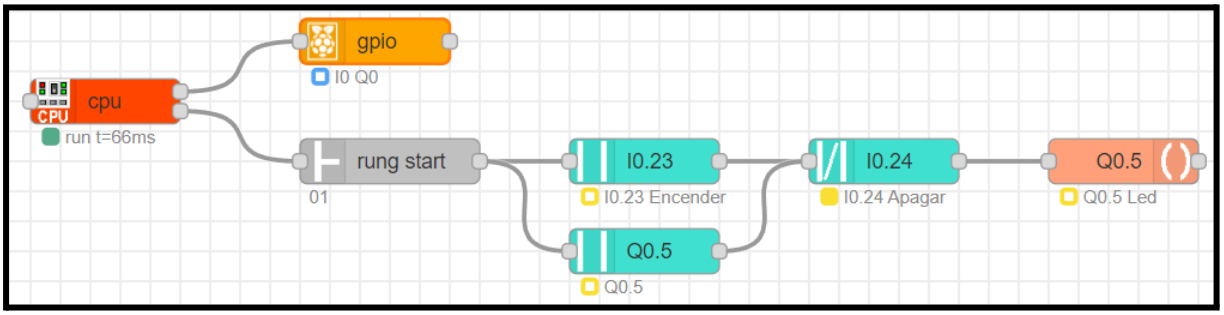
Ya tenemos el botón de encender y el apagar, ahora necesitamos definir la led. Insertamos un nodo de tipo Coil, bobina.



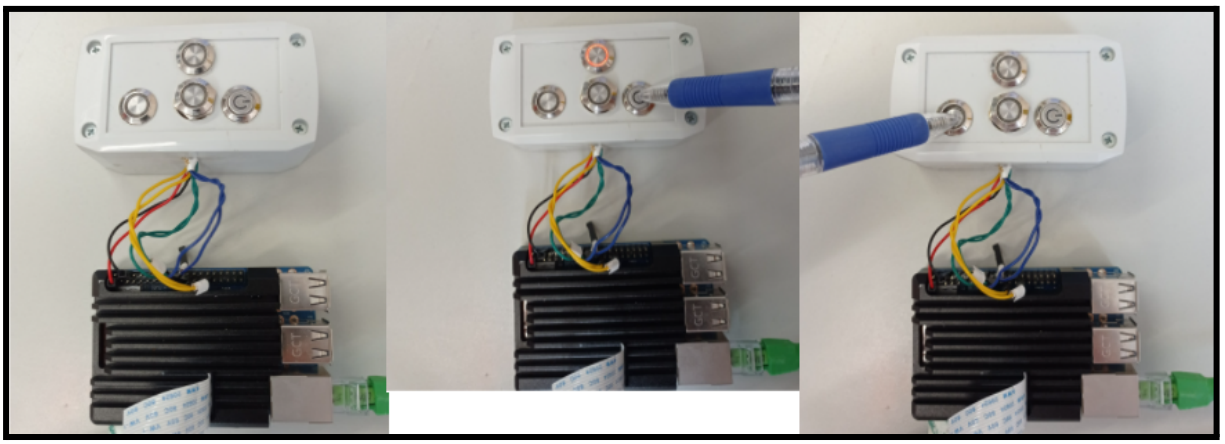
Al pulsar doble clic aparecen las propiedades, definiremos el tipo de operación, el gpio que utiliza y un nombre identificador.



Para que la luz se mantenga encendida hasta que el botón de apagar se pulse debemos añadir un contacto normalmente abierto con el mismo nombre que la bobina de salida utilizada para que se produzca el enclavamiento.



Con esta práctica hemos visto como podemos utilizar la raspberry como un PLC y programar utilizando Ladder.

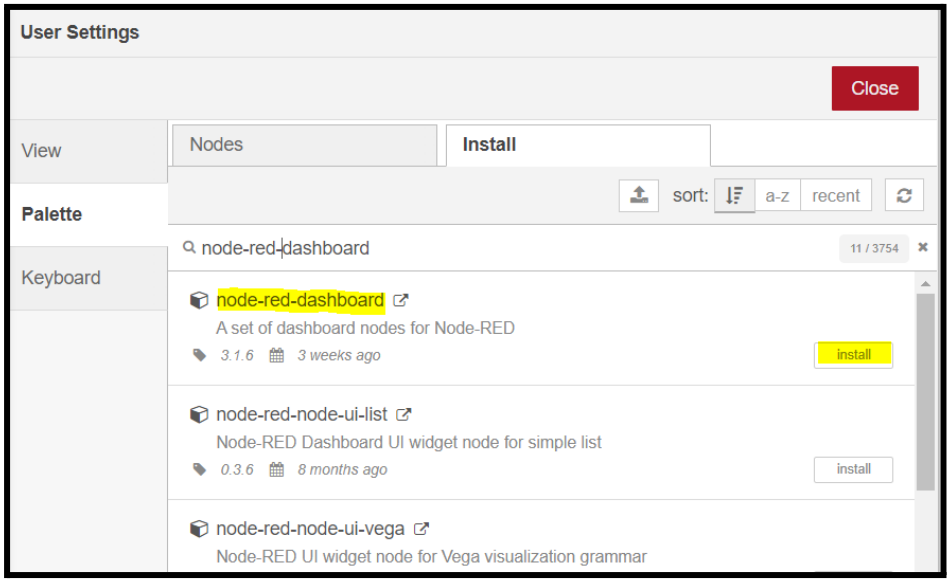


Dashboard

Instalación

Si queremos visualizar estos datos en un tablero de datos en vivo, similar a un HMI, podemos instalar el módulo **dashboard**.

Desde la esquina superior derecha accedemos al menú general y buscamos **Manage Palette** y en la pestaña Install buscamos **node-red-dashboard** e instalamos.



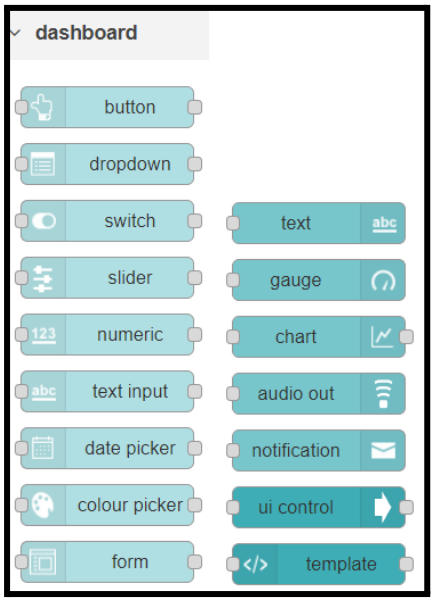
El Dashboard de Node-RED se distribuye en una especie de rejilla, cada elemento grupo tiene un tamaño, por defecto se le asigna una anchura de 6 unidades, cada una se compone de 48 píxeles.

En cada grupo existen widgets, los cuales también poseen la propiedad de la anchura, por defecto será automática y ocupará la anchura del elemento grupo.

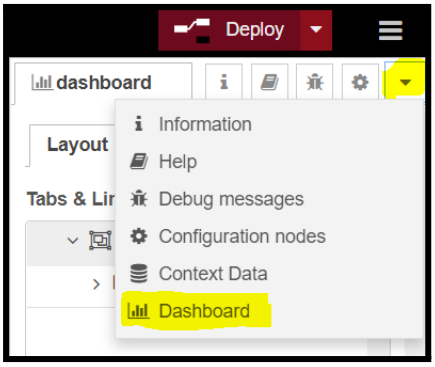
Existen diversos nodos que ofrecen una gran utilidad a la hora de combinar una interfaz gráfica de visualización de resultados con la interactuación entre el usuario y el flujo de datos diseñado.

El diseño del widget se administra mediante una pestaña del tablero en la barra lateral del editor de Node-RED.

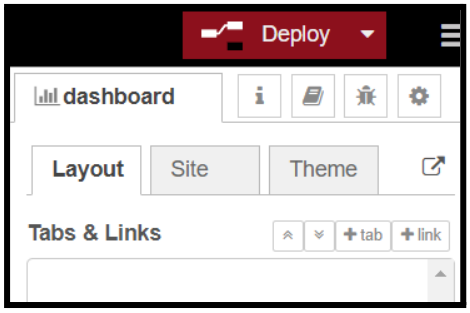
Se ha añadido la pestaña dashboard con los siguientes nodos:



En la parte derecha de node-red se ha añadido una pestaña **dashboard.**



Dentro de dashboard aparecen tres subpestañas: Layout, Site y Theme.

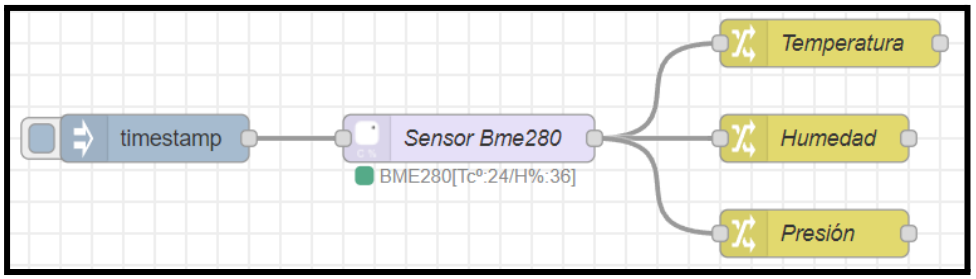


Ahora trabajaremos con **Layout.** Se distinguen:

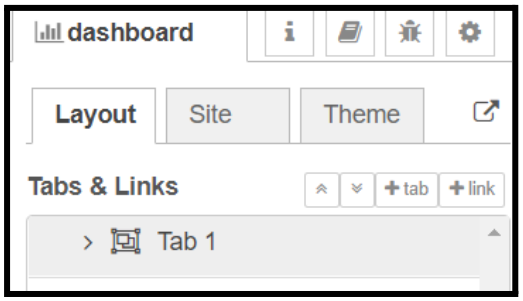
- Tabs: desde aquí puede reordenar las pestañas, grupos y widgets, y agregar y editar sus propiedades. También puede abrir las herramientas de diseño que pueden ayudarlo a organizar los widgets más fácilmente que a través de la barra lateral.
- Links: también se pueden agregar al menú enlaces a otras páginas web.

Práctica, visualización datos sensor BME280 en dashboard

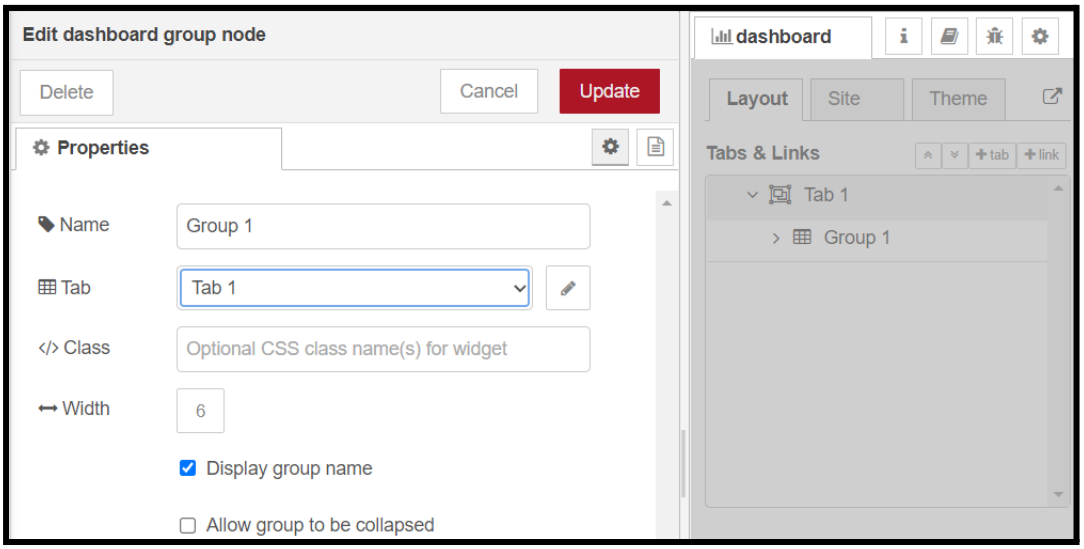
Partimos del último ejemplo en el que utilizamos un sensor para medir temperatura, humedad y presión.



Añadimos un tab , es como un “menú”.

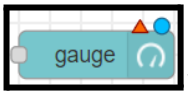


Dentro de este tab o menú añadimos un grupo.

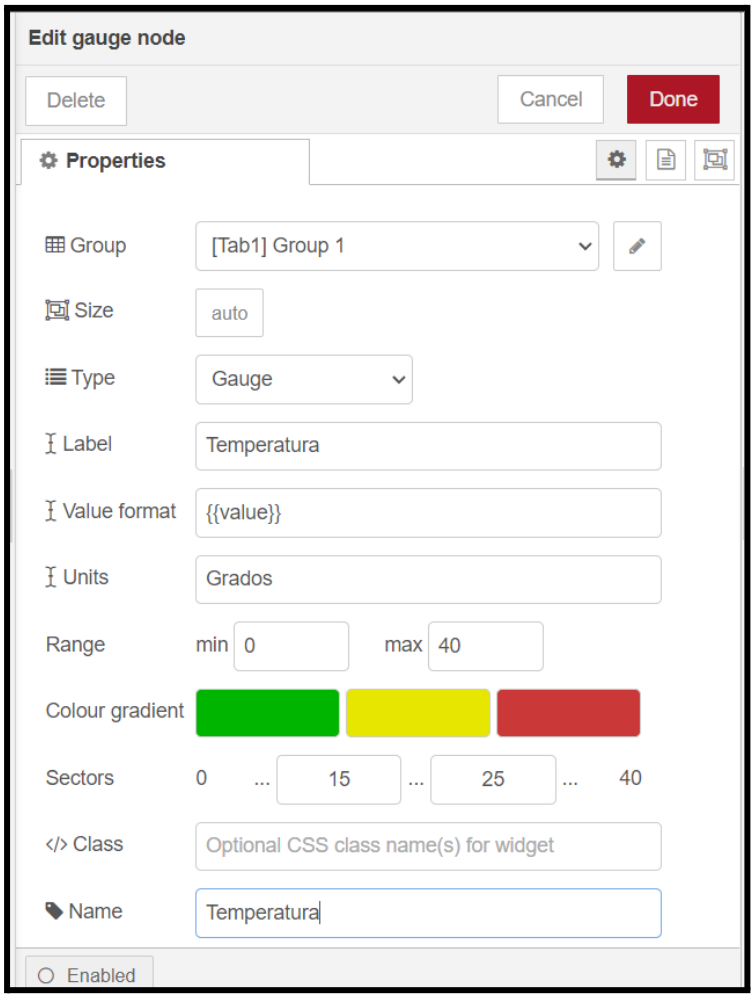


De momento dejamos los nombres por defecto para ver en el ejemplo como aparece.

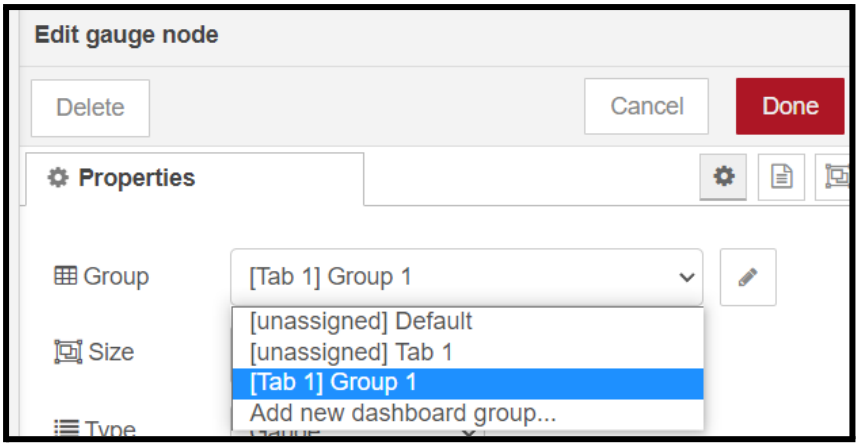
Ahora podemos añadir los nodos de dashboard que deseemos.

Agregamos un nodo de tipo indicador 

y editamos.

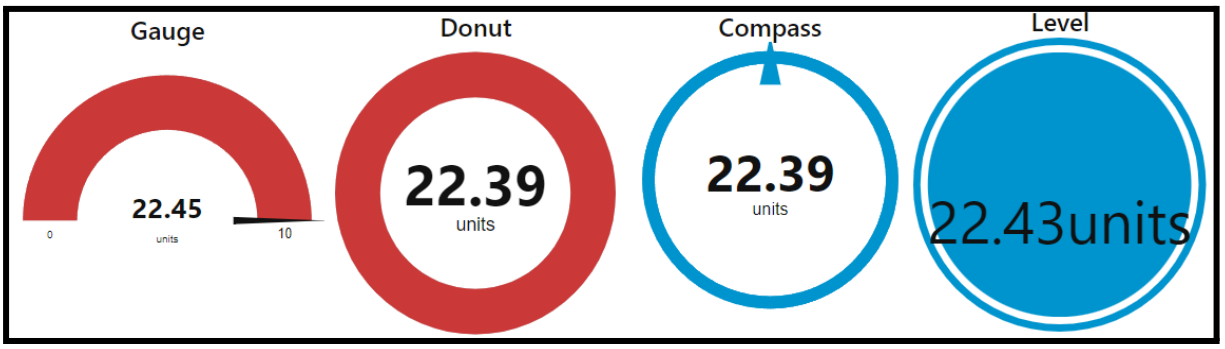


Seleccionamos el grupo o menú donde deseamos que aparezca.



Podemos cambiar el tamaño o dejar en automático.

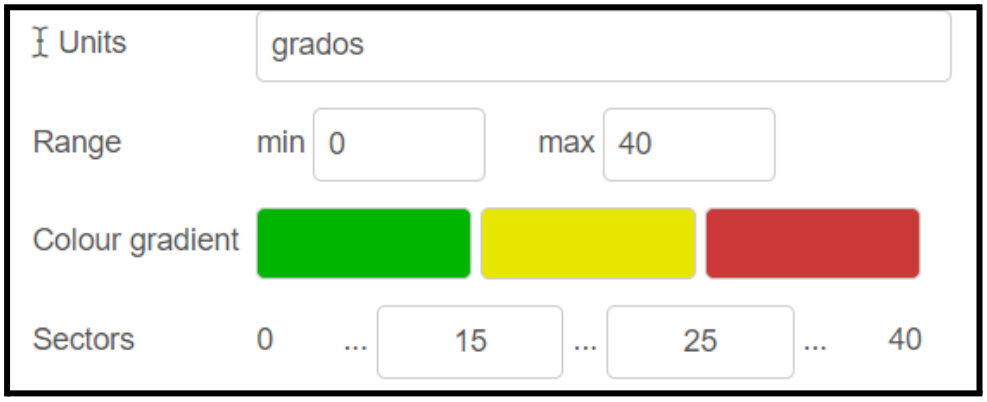
El indicador tiene varios tipos: gauge, donut, compass y level.



Units ponemos un texto con la unidad de medida.

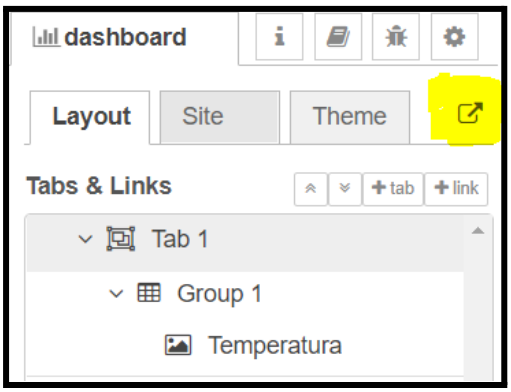
En rango marcamos un mínimo y un máximo en la medida. En el ejemplo va de 0º a 40º. Se pueden especificar los colores de cada uno de los 3 sectores y el indicador se mezclará entre ellos. Los colores deben especificarse en formato hexadecimal (#rrggbb).

Si especifica números para los sectores, los colores cambian por sector. Si no se especifica, los colores se mezclan en toda la gama.

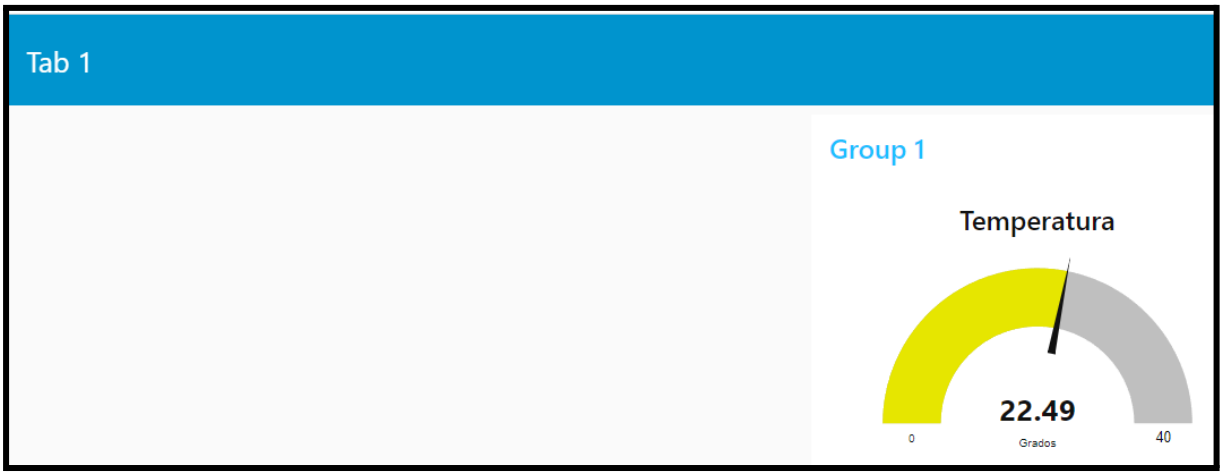


De 0 a 15 saldrá verde, de 16 a 25 amarillo y de 26 a 40 rojo.

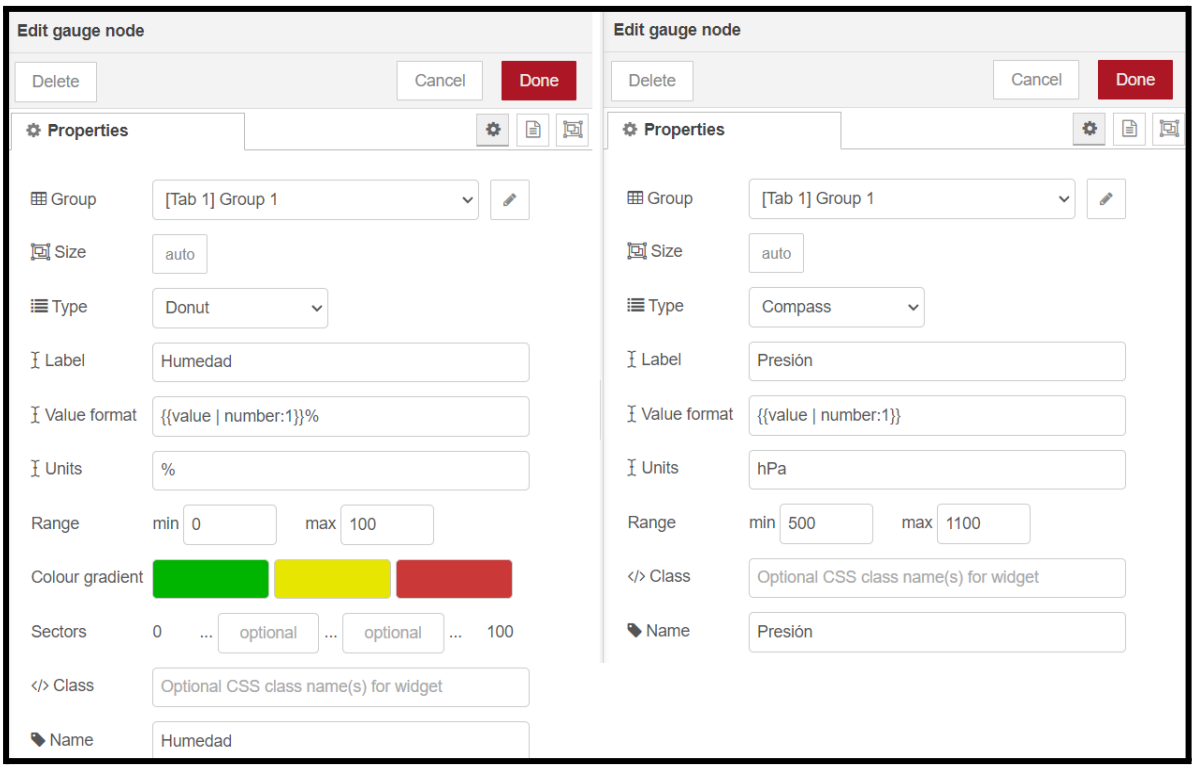
Para acceder a la dashboard podemos desde el navegador https://ip:1880/ui o acceso rápido desde node-red en la parte derecha en la pestaña dashboard.



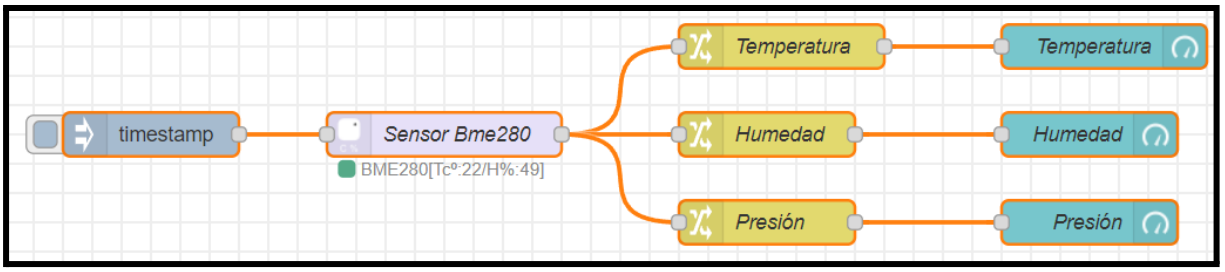
Así es su apariencia.



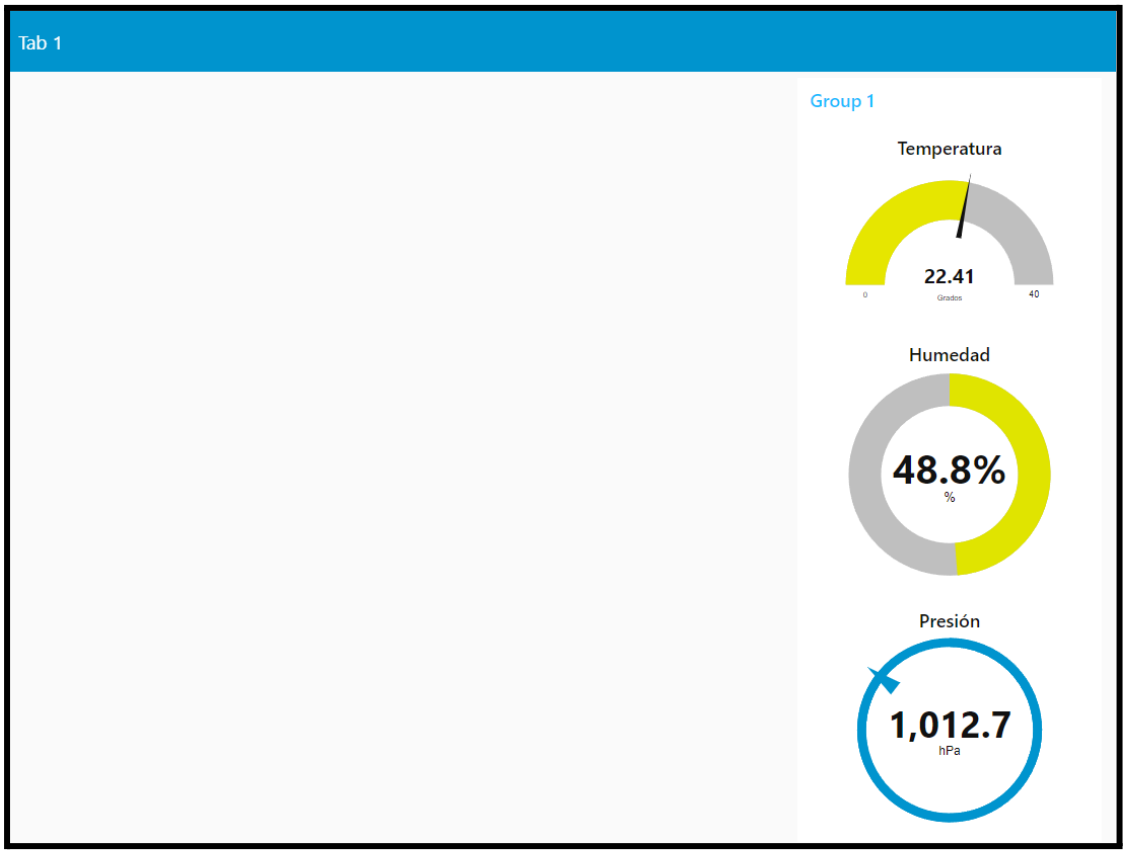
Añadimos un nodo que muestre la humedad y otro la presión.



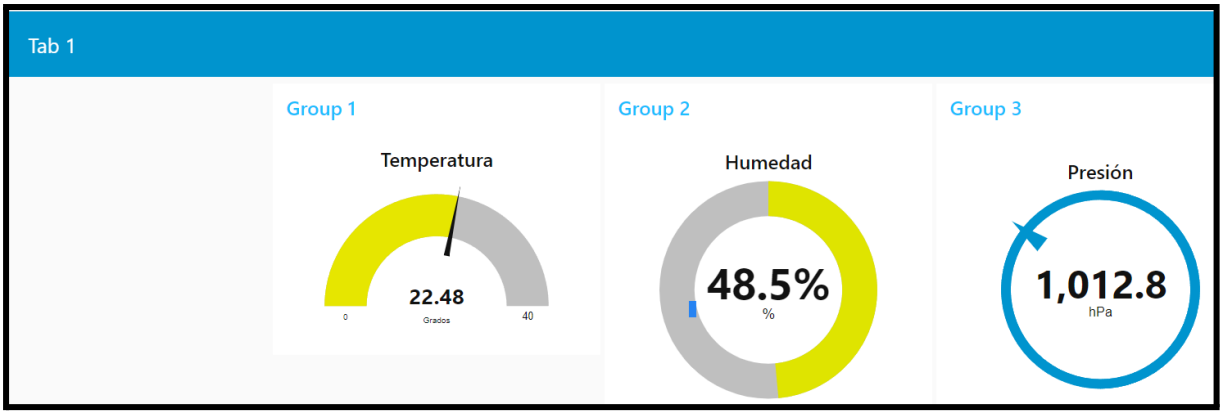
Unimos estos nodos con los nodos que tienen el valor del sensor.



Después de todas las modificaciones en node-red pulsaremos en Deploy y accionamos timestamp para actualizar los datos.
Así se me mostrará Dashboard.



Si creamos más grupos y desplazamos los nodos a esos grupos se muestran en línea.



El siguiente paso puede ser disponer de la información del sensor desde cualquier lugar y dispositivo.

Utilizaremos para ello el protocolo de comunicación, MQTT, que nos permite subir los datos a un servidor (broker) MQTT

MQTT

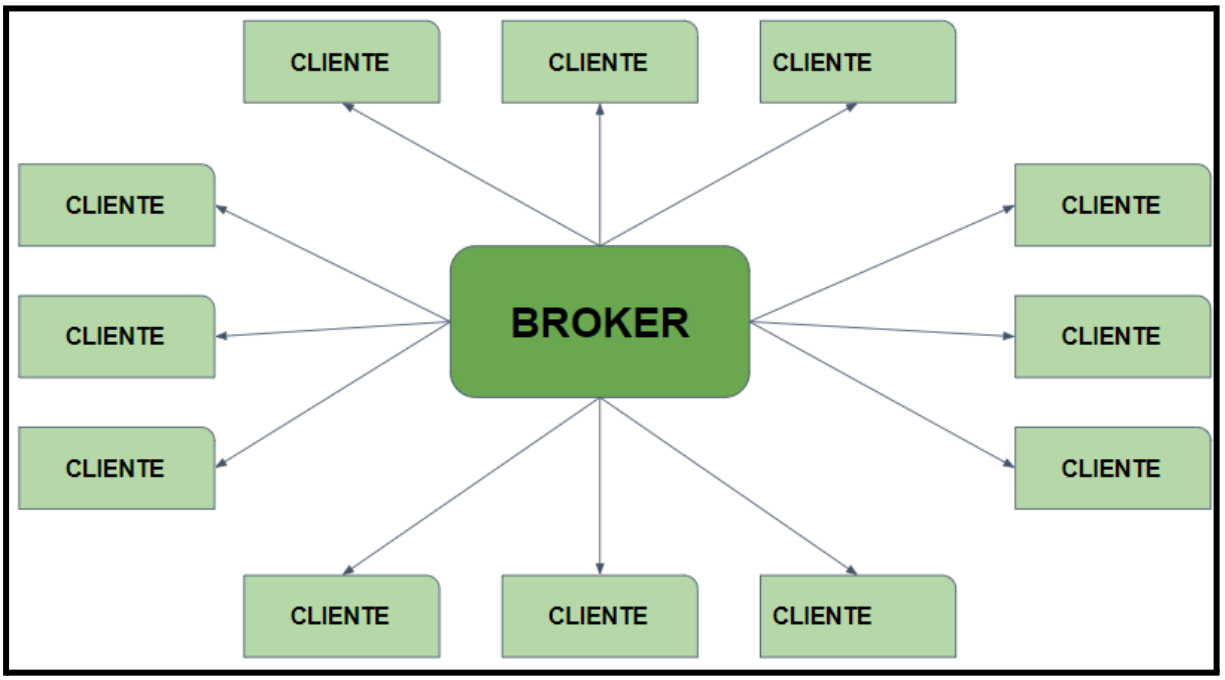
MQTT es un protocolo M2M y son siglas en inglés, Message Queue Telemetry Transport. Está basado en un protocolo de mensajería publicación/suscripción, al contrario que HTTP que es petición/respuesta.

Es un protocolo simple y ligero. Es muy interesante para sistemas que requieren poco ancho de banda, tienen una alta latencia y requieren de poco consumo de los dispositivos. La latencia es el tiempo desde que se envía un mensaje hasta que llega a su destino. Cuanto más tiempo, más latencia.

Los objetivos del protocolo MQTT es minimizar el ancho de banda, comunicación bidireccional entre dispositivos, minimizar los requerimientos de los dispositivos tanto recursos como consumo y garantizar la fiabilidad y cierto grado de seguridad.

Los dispositivos se conectan utilizando una topología en estrella, todos los clientes se conectan directamente a un punto central que hace de servidor. En MQTT este servidor se llama Broker y todos los mensajes pasan por él.

Los clientes que se conectan a él ya sea para enviar o recibir mensajes. Los Clientes, al iniciar se suscriben a tópicos, y comenzarán a recibir todos los mensajes que se publiquen en ese tópico.

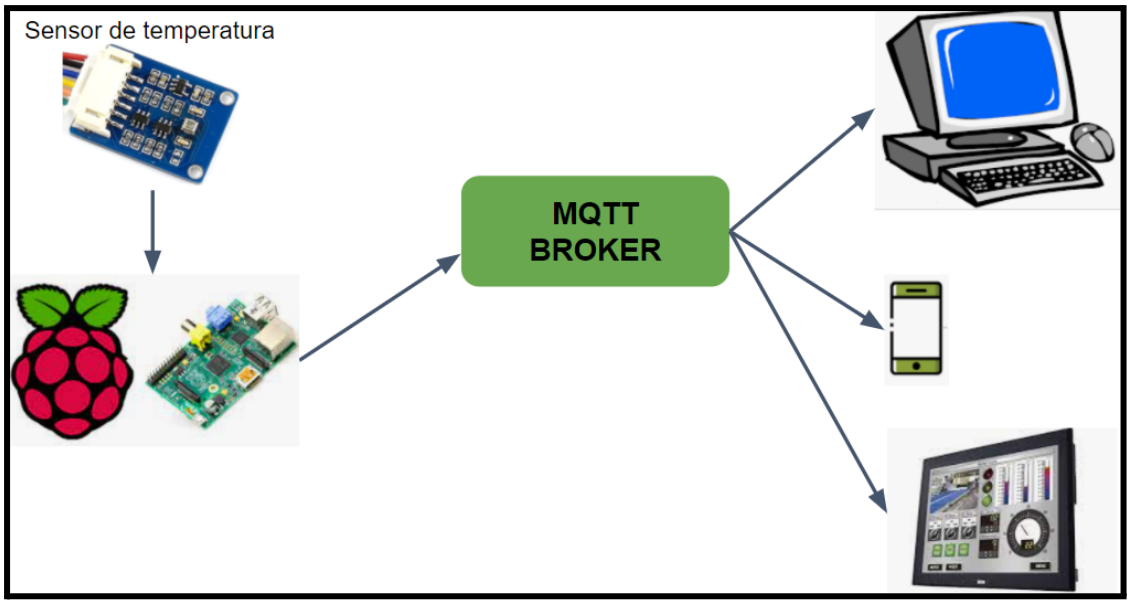


El topic es el tema donde se suscriben los receptores para recibir el mensaje.

El Broker se encarga de distribuir los mensajes a los receptores, cuando llega un nuevo mensaje se envía a los clientes que estén suscritos al topic.

Un cliente se suscribe a un topic que es como el tema, a quién va dirigido el mensaje. En la imagen siguiente tenemos un servidor MQTT Broker y cuatro clientes que son un Sensor de Temperatura, un ordenador, un teléfono móvil y un HMI..

El ordenador, el teléfono y HMI se conectan al Servidor MQTT Broker y se suscriben al tópico «temperatura».

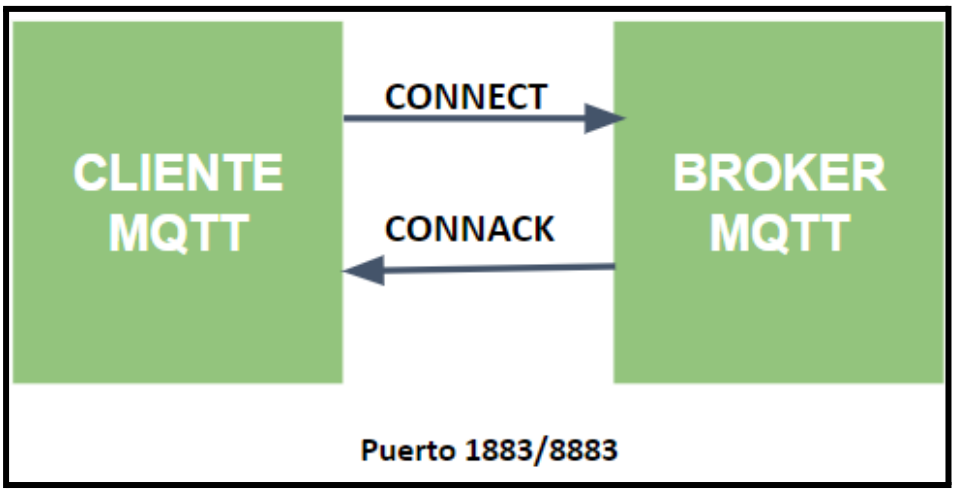


Funcionamiento

Cada cliente inicia una conexión TCP/IP con el broker, el cual mantiene un registro de cada cliente conectado.

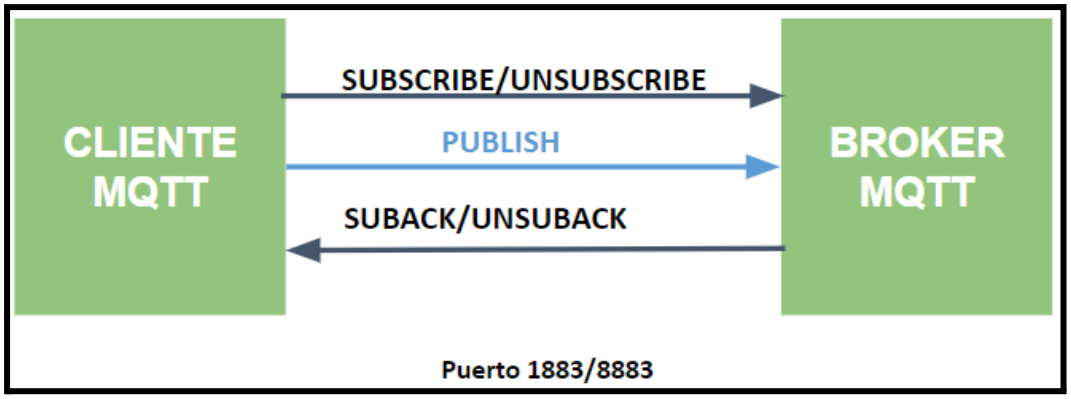
Esta conexión se mantiene abierta hasta que el cliente la finaliza. MQTT utiliza el puerto 1883 o el 8883 (cuando funciona sobre protocolo TLS).

El cliente envía un mensaje CONNECT que contiene datos identificativos como nombre de usuario, contraseña, cliente-id,.. El broker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc).



Para enviar mensajes el cliente utiliza mensajes PUBLISH que contienen el topic y el payload.

Para suscribirse o desuscribirse el cliente utiliza mensajes SUBSCRIBE o UNSUBSCRIBE, que el broker responde suscrito o desuscrito con SUBACK o UNSUBACK.



Para asegurar que la conexión está activa los clientes envían periódicamente un mensaje PINGREQ que es respondido por el broker con un PINGRESP. Al final el cliente se desconecta enviando un mensaje de DISCONNECT.

Los mensajes constan de 3 partes:

- Cabecera fija, está formado por un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje.
- Cabecera variable, es opcional y contiene información adicional que es necesaria en ciertos mensajes o situaciones.
- Contenido (payload), es el contenido real del mensaje.

Un **topic** dentro de MQTT es una cadena de caracteres que representa el tema del mensaje, sirve para que el broker identifique hacía que cliente pudiera ir dirigido un mensaje publicado por otro cliente.

El **broker** es el “servidor” que establece la red MQTT, acepta y recibe mensajes publicados por los clientes (publicadores) en **topics** específicos, identifica y distribuye estos mensajes entre los clientes suscritos (suscriptores) que correspondan con el **topic**.

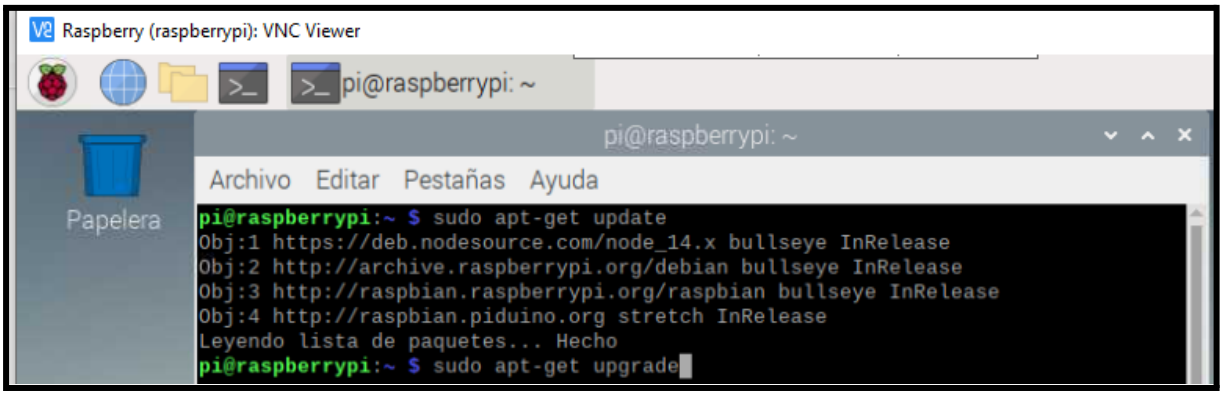
Instalación de un broker MQTT en la Raspberry

Accedemos a la Raspberry desde VNC y abrimos una terminal de comandos.

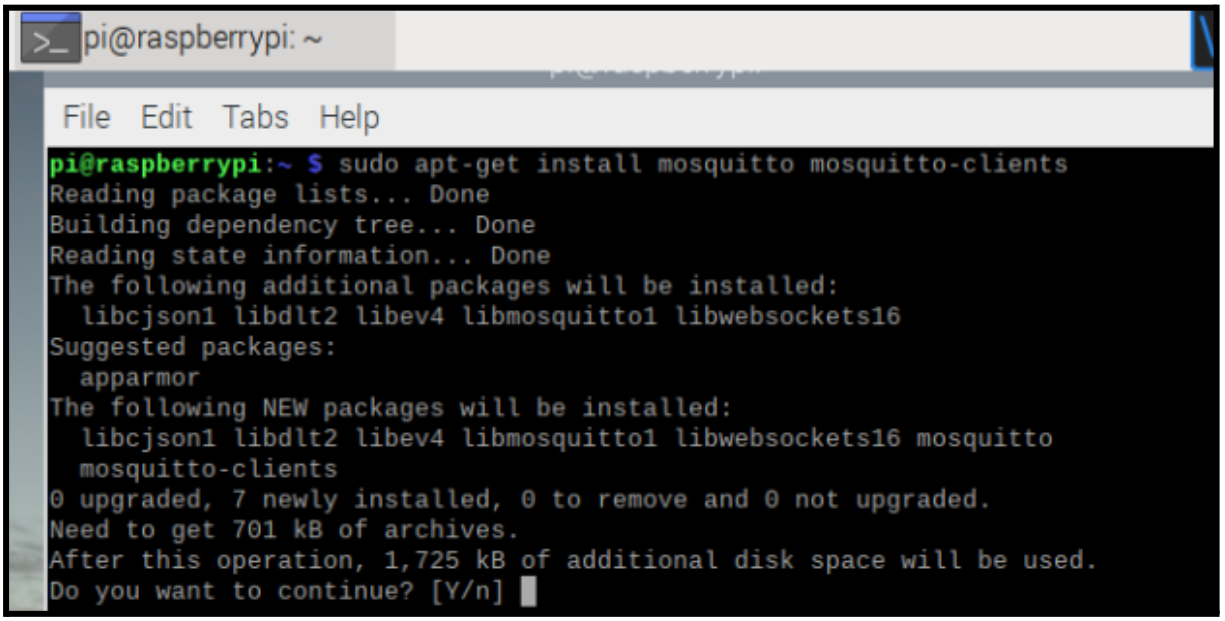
Podemos realizar la instalación de MQTT, llamado mosquito, desde la terminal de comandos o desde el entorno gráfico.

Terminal de comandos

Lo primero que realizaremos es una actualización del sistema operativo ejecutando **apt-get update** y **apt-get upgrade**



Realizamos la instalación del mosquito ejecutando **apt-get install mosquitto mosquitto-clients** ((para la parte “servidor” **y** para contar con un “cliente” mqtt**)**



Para activar el servicio cada vez que arranque la Raspberry ejecutamos:
sudo systemctl enable mosquitto.service

```
pi@raspberrypi:~ $ sudo systemctl enable mosquitto.service
Synchronizing state of mosquitto.service with SysV service script with /lib/syst
emd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
```

Podemos probar que el sistema funciona, abriendo un par de ventanas y usando los comandos mosquitto_sub y mosquitto_pub, que permite suscribir o publicar algo en el server MQTT.

En una ventana escribimos: **mosquitto_sub -d -h localhost -p 1883 -t "AEG/CO2/temp"**

```
pi@raspberrypi:~ $ mosquitto_sub -d -h localhost -p 1883 -t "AEG/CO2/temp"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: AEG/CO2/temp, QoS: 0, Options: 0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
```

y en la otra: **mosquitto_pub -d -h localhost -p 1883 -t "AEG/CO2/temp" -m "99.99"**

```
pi@raspberrypi:~ $ mosquitto_pub -d -h localhost -p 1883 -t "AEG/CO2/temp" -m "99.99"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, 'AEG/CO2/temp', ... (5 bytes))
Client (null) sending DISCONNECT
pi@raspberrypi:~ $
```

Verás que en la ventana en la que suscribiste el tópico «AEG/CO2/temp»-m » aparece el valor que publica la otra venta de 99.99

```
Client (null) received PUBLISH (d0, q0, r0, m0, 'AEG/CO2/temp', ... (5 bytes))
99.99
Client (null) sending PINGREQ
Client (null) received PINGRESP
```

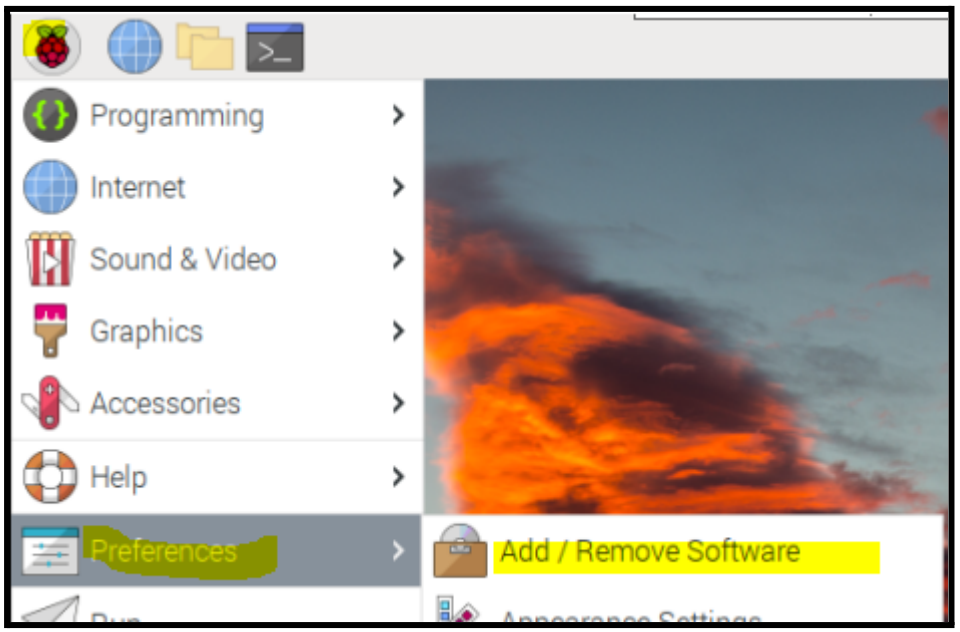
pi@raspberrypi: ~

File Edit Tabs Help

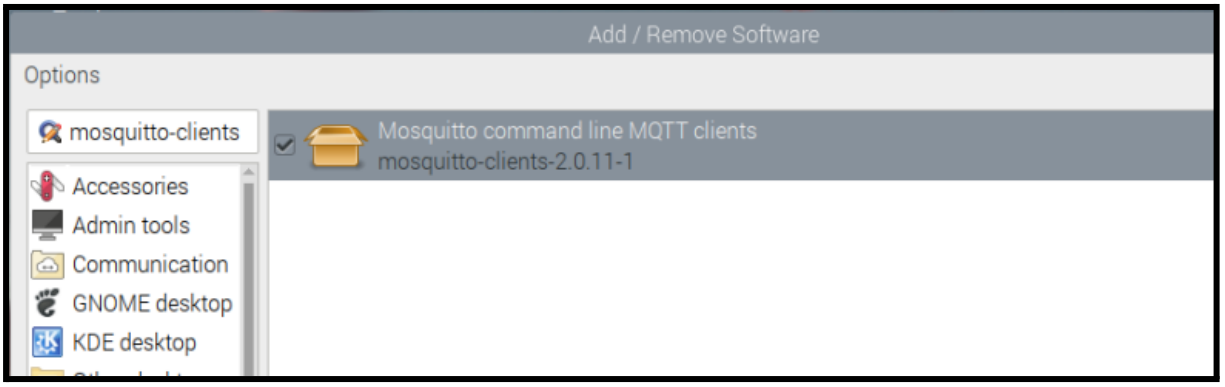
```
pi@raspberrypi:~ $ mosquitto_pub -d -h localhost -p 1883 -t "AEG/CO2/temp" -m "99.99"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q0, r0, m1, 'AEG/CO2/temp', ... (5 bytes))
Client (null) sending DISCONNECT
pi@raspberrypi:~ $
```

Menú principal

Desde el menú principal accedemos a preferencias y añadir o quitar programas.



Buscaremos mosquitto-clients, marcamos e insertamos.



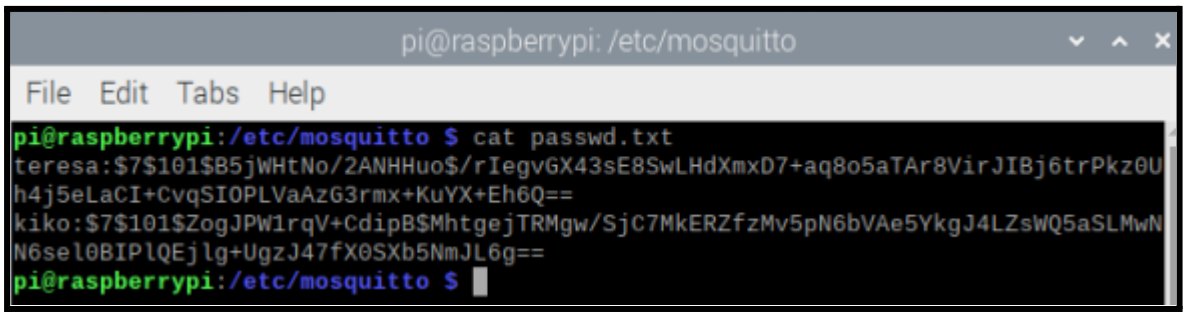
Creación de usuarios

Creamos un fichero de texto con los nombres de usuario y contraseña.
Ejecutamos **sudo nano /etc/mosquitto/passwd.txt** para crear el fichero y le añadimos los usuarios y contraseñas.

Guardamos el fichero y ahora lo encriptamos ejecutando:

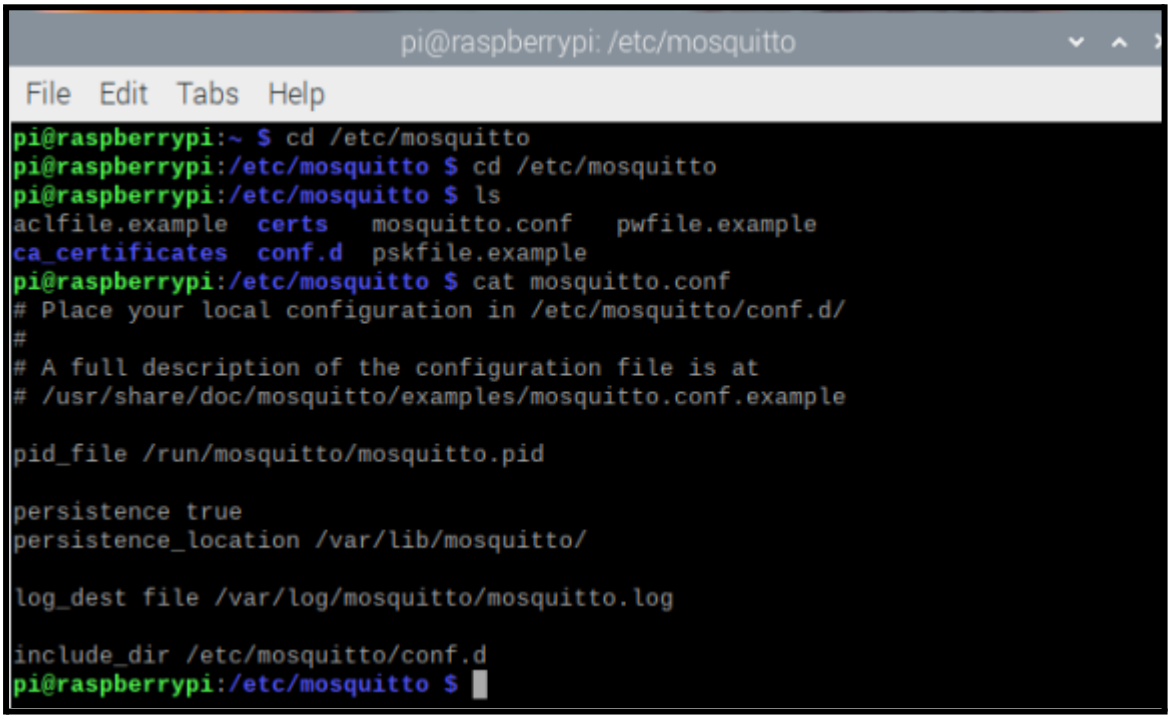
sudo mosquitto_passwd -U passwd.txt

Ahora en el fichero ha modificado las contraseñas



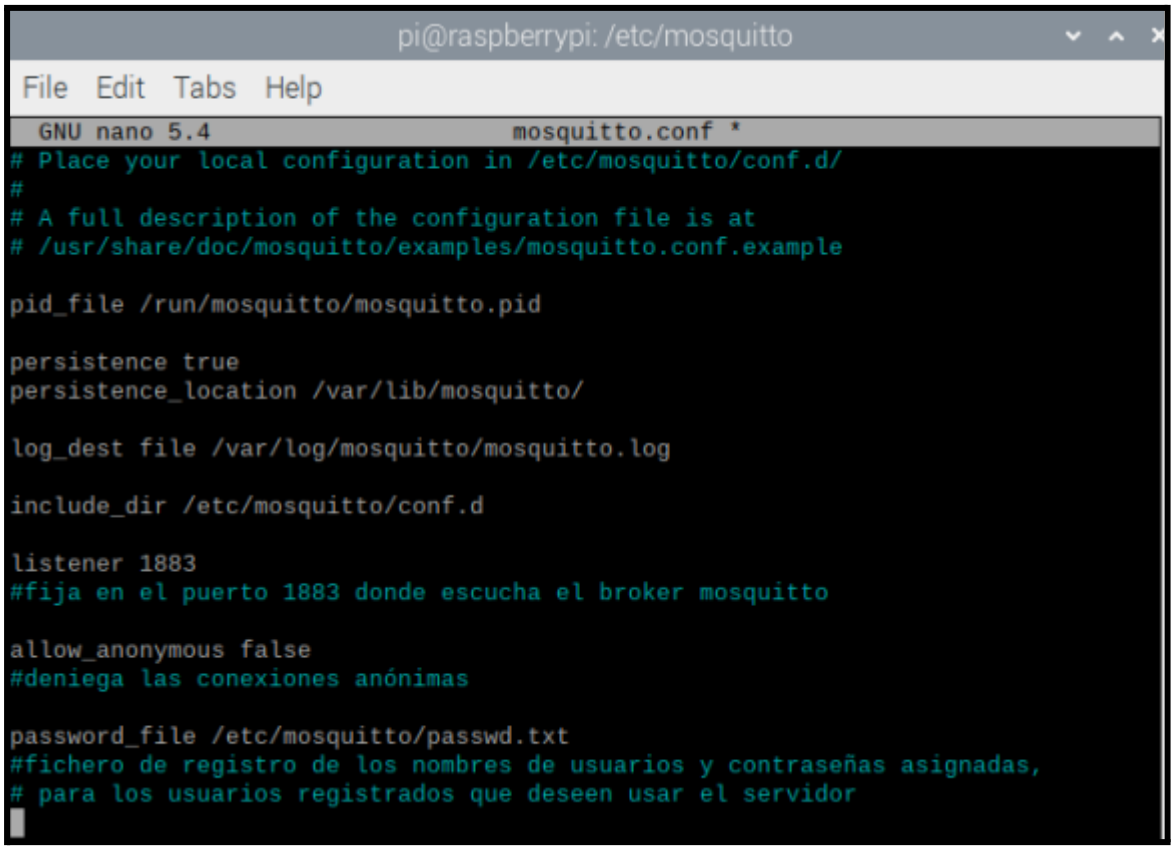
Configuración

Todo el comportamiento del Broker Mosquitto, se controla desde el fichero **mosquitto.conf**, que está situado en el mismo directorio de instalación de Mosquitto **/etc/mosquitto.**



Editamos el fichero, añadimos las siguientes líneas:

listener 1883
allow_anonymous false
password_file /etc/mosquitto/passwd.txt



Como hemos modificado el archivo de configuración se necesita rearrancar el servicio, para ello ejecutaremos:

sudo systemctl status mosquitto
sudo systemctl stop mosquitto
sudo systemctl start mosquitto

Si queremos añadir otro usuario debemos escribir:

sudo mosquito_passwd -b passwd.txt *usuario contraseña*

Si queremos eliminar un usuario debemos escribir:

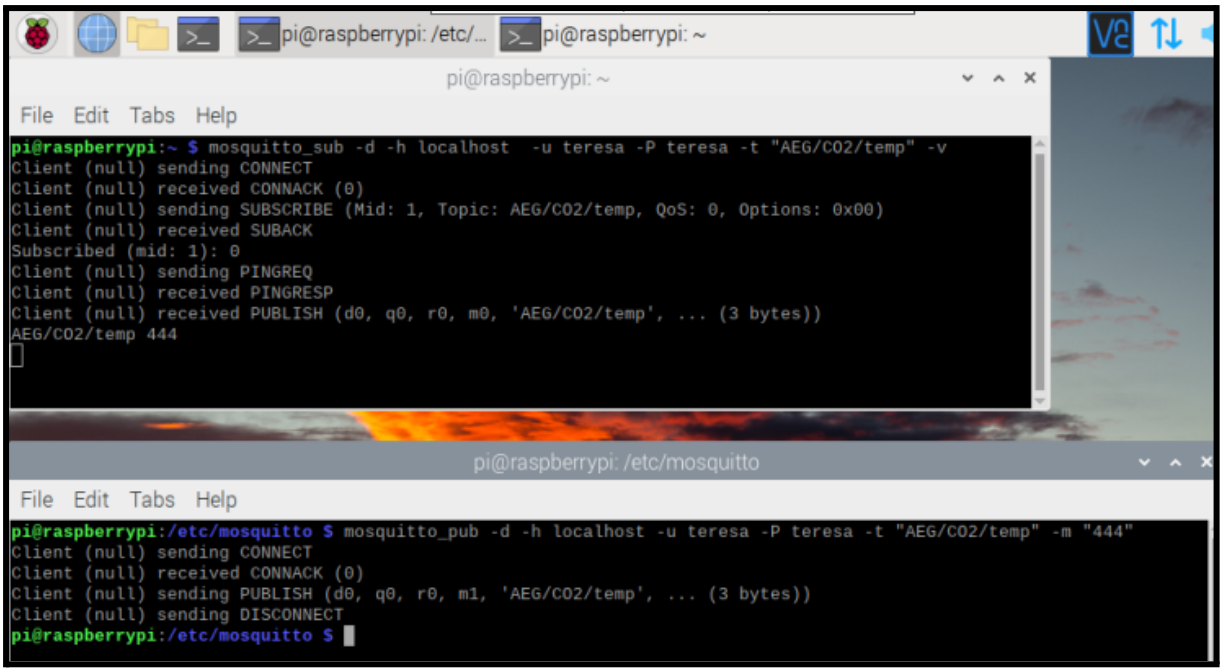
sudo mosquito_passwd -D passwd.txt *usuario*

Ahora probaremos a publicar y suscribir con la implementación de seguridad por usuario y contraseña.

Abrimos dos terminales de comandos y en una nos suscribimos y en otra publicamos utilizando la seguridad.

- Publicar
  mosquitto_sub -d -h localhost -u teresa -P teresa -t "AEG/CO2/temp" -v

- Suscribir
  mosquitto_pub -d -h localhost -u teresa -P teresa -t "AEG/CO2/temp" -m "444"

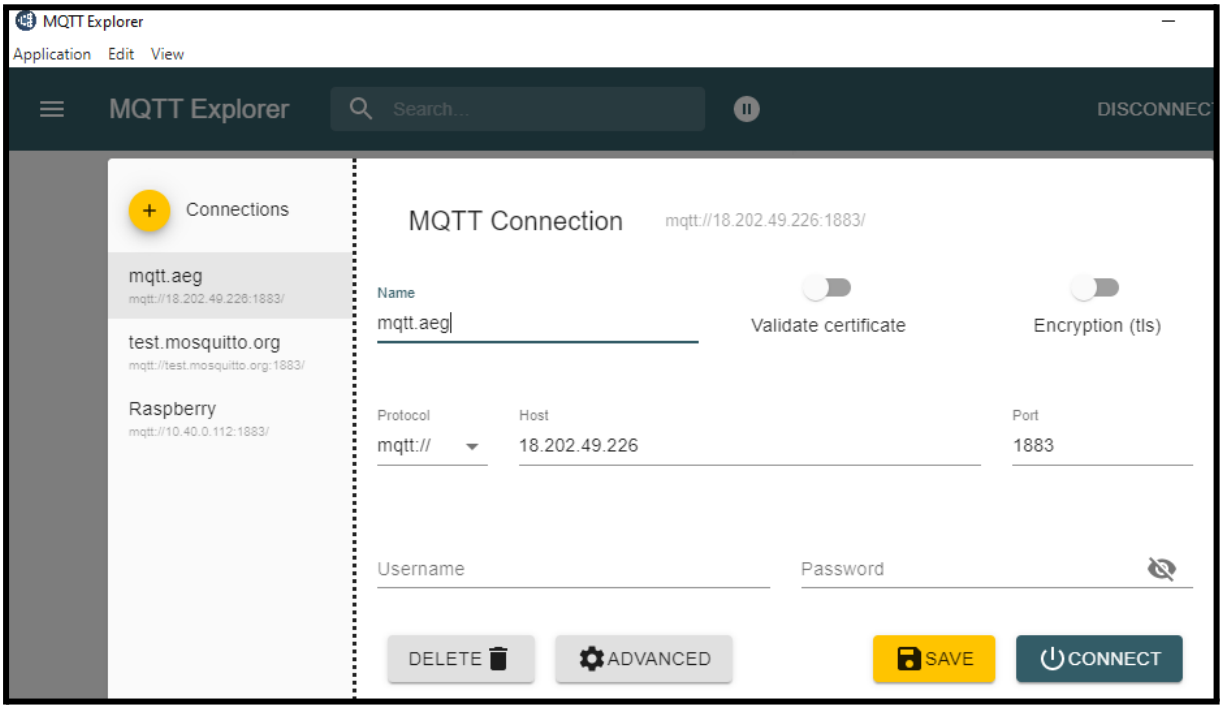


MQTT Explorer

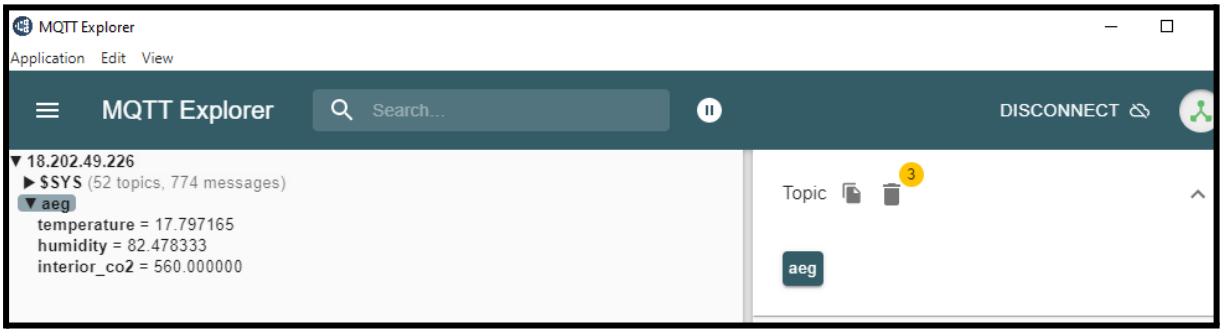
Existe una herramienta gratuita, MQTT Explorer que podemos utilizar para visualizar tanto topics como mensajes, buscar, filtrar, publicar mensajes, incluso realizar gráficas de los valores recibidos.

Podemos descargarla para nuestro sistema desde el siguiente enlace
http://mqtt-explorer.com/

Al ejecutar nos aparece esta pantalla para seleccionar conexión o crear una nueva. Debemos saber el host y añadimos usuario y contraseña (si los tiene creados), seguido pulsaremos **Conectar**

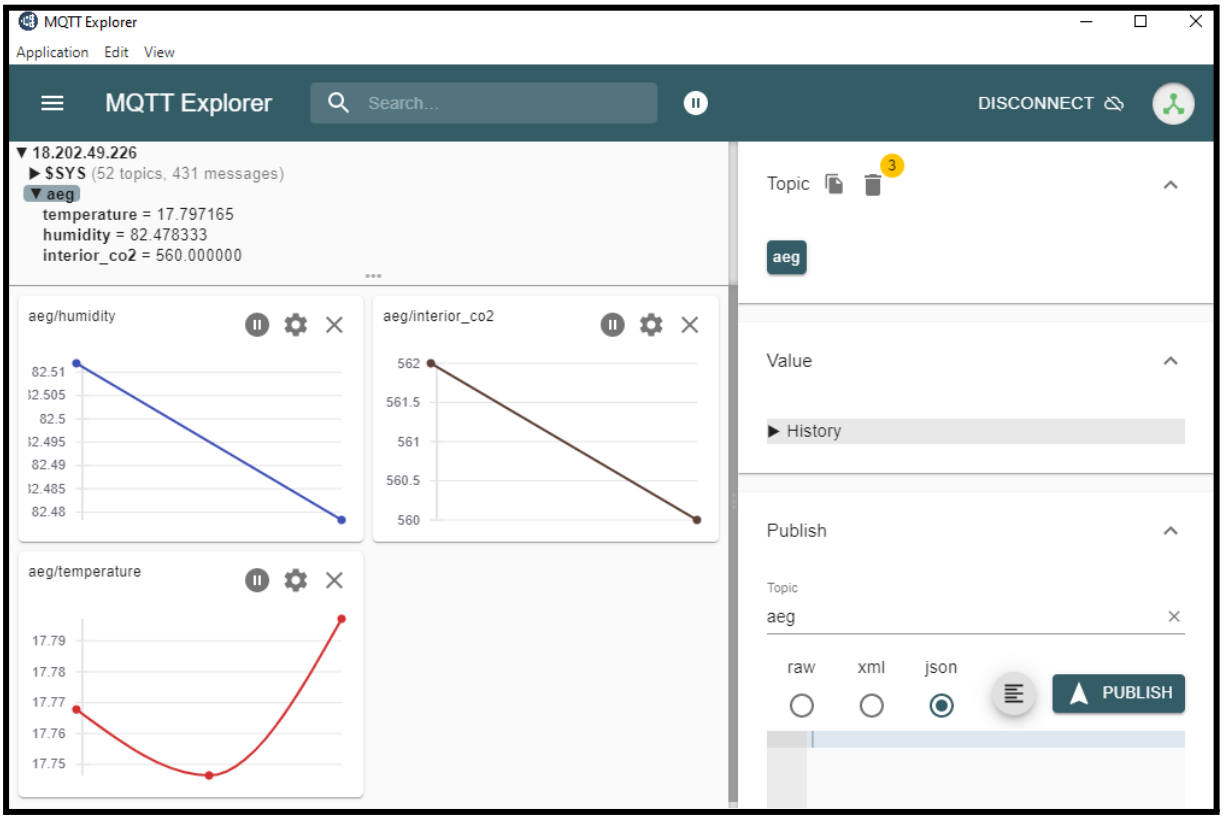


Al conectarme al broker me aparece la siguiente pantalla:

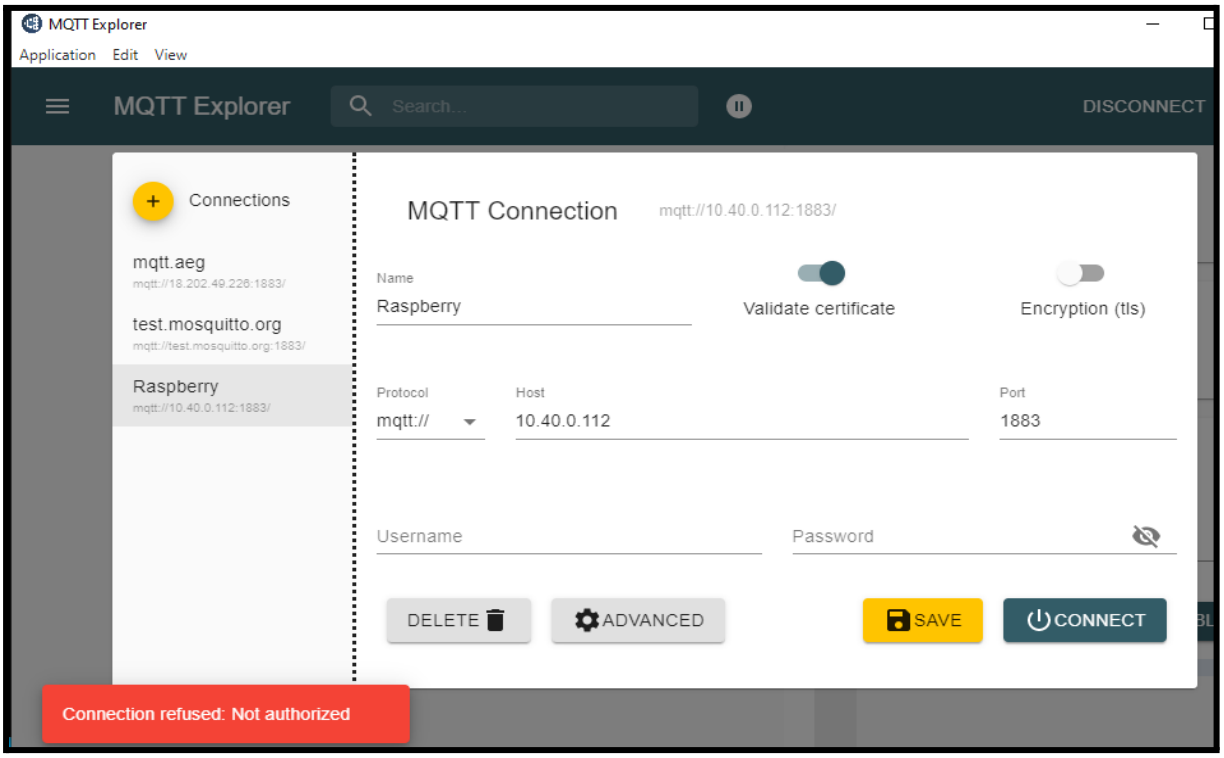


Debajo del servidor aparece el tópico SYS de sistema y la conexión a la que hemos accedido aeg, con una flecha para desplegarla y al pulsar, veremos los topic publicados, en este caso temperature, humidity e interior_co2.

Tiene diferentes funciones para visualizar los datos, aquí desde el desplegable value hemos creado unos gráficos con los datos.



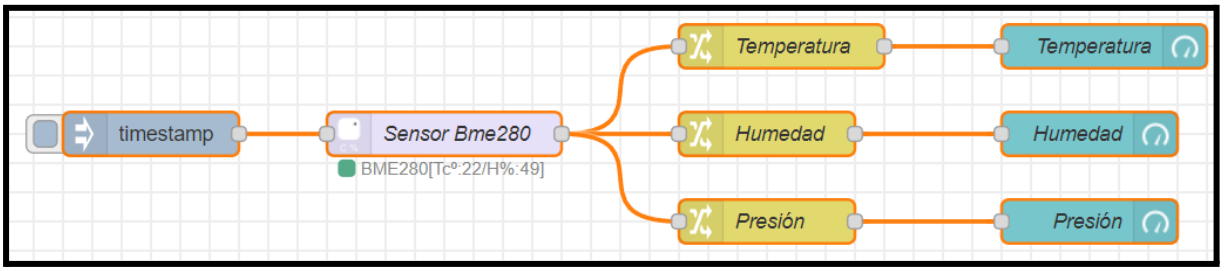
Creamos una conexión a Raspberry e intentamos suscribirnos al topic que creamos.



Al no introducir usuario y contraseña no se autoriza la conexión.

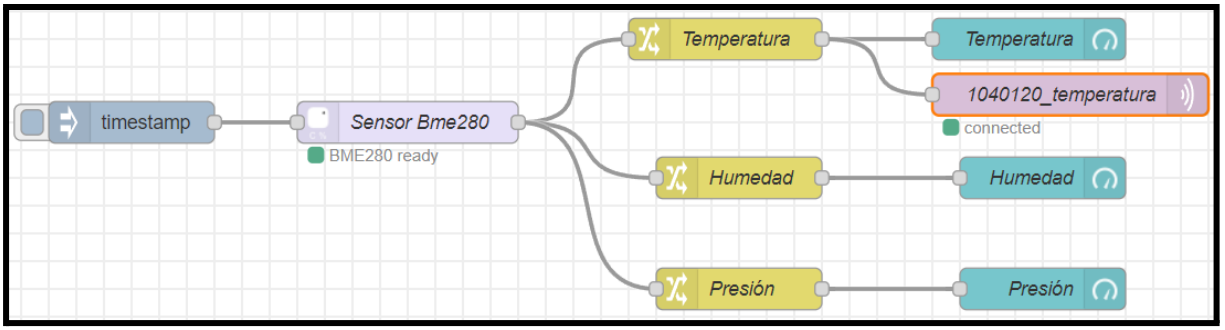
Práctica, publicar datos en un broker

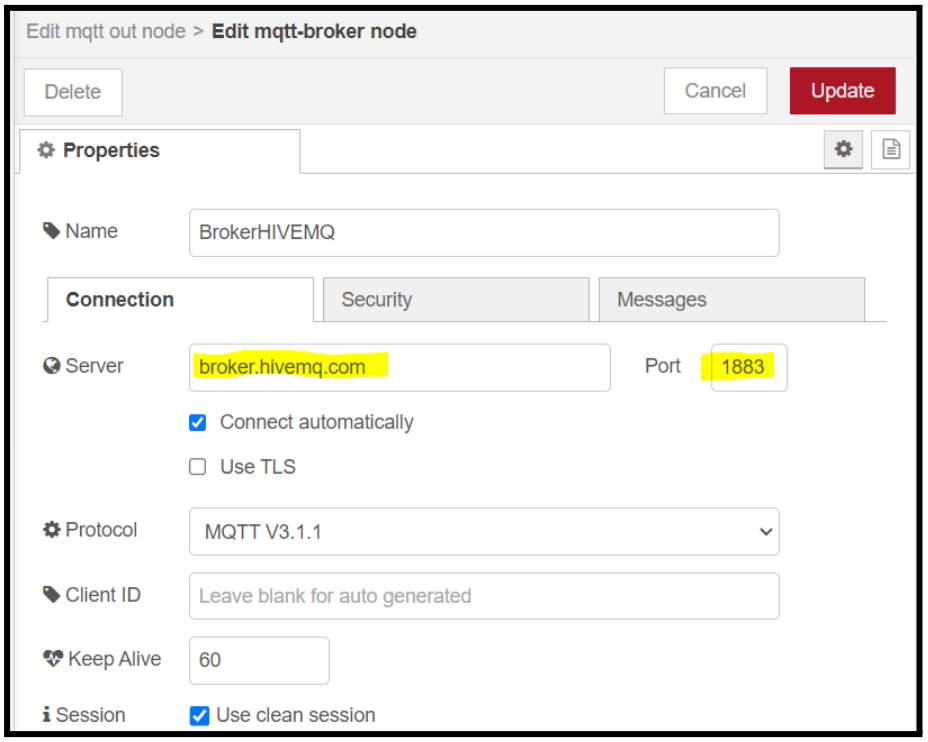
Vamos a continuar con nuestra práctica anterior y subir los datos de nuestro sensor a la nube.
Tenemos los datos del sensor, temperatura, humedad y presión.



Vamos a publicar los datos en un broker público en la nube llamado **HIVEMQ**.
Seleccionamos un nodo mqtt de salida y lo arrastramos a nuestro flujo.

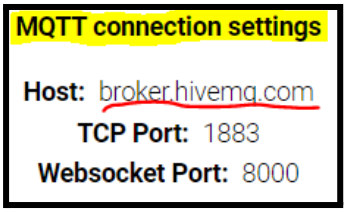
Pulsamos doble clic en el nodo e introducimos las propiedades.
En Server, desplegamos o damos al boli, seleccionamos el servidor.



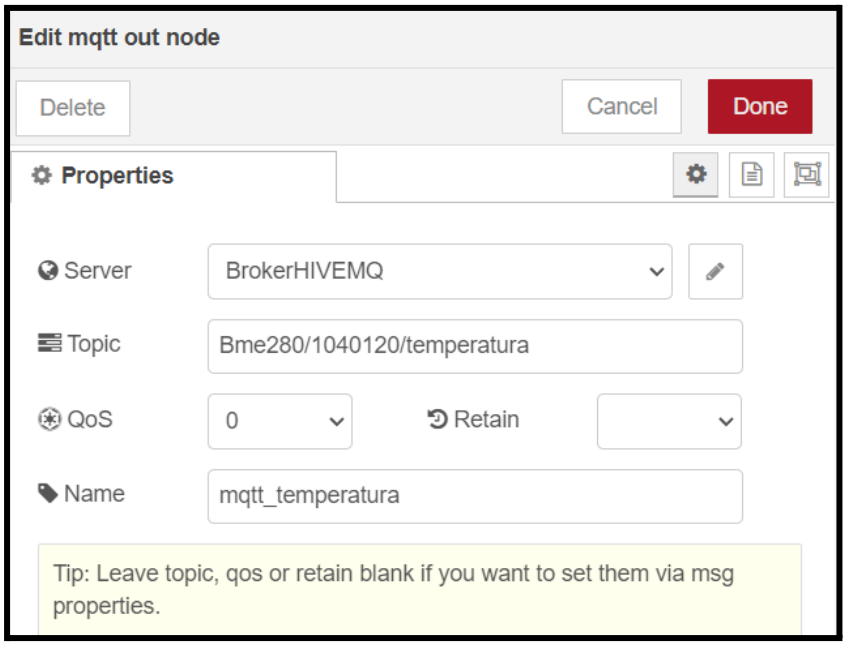


En la propiedades de la pantalla “Edit mqtt-broker node” en **server** ponemos la dirección del broker **hivemq (***broker.hivemq.com***)** y su puerto de conexión (1883) o pondríamos la IP local de una raspberry, ordenador que tuviera instalado un broker mqtt.

Si accedemos a su web http://www.mqtt-dashboard.com/ podemos ver los datos para conectar.



En **Cliente ID** se puede poner un identificador de cliente único o se puede dejar en blanco.
Las pestañas de seguridad y mensajes por ahora no utilizaremos.
Ya en la pantalla “Edit mqtt out node”

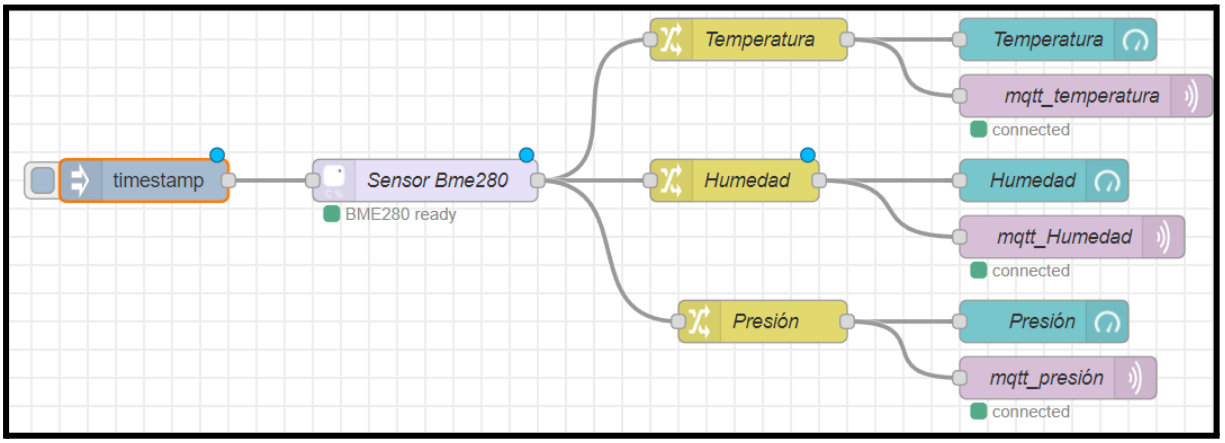


Server, servidor mqtt.
Topic, será un nombre descriptivo y único, hay que tener en cuenta que se va a publicar en un servidor gratuito en la nube. En este caso he utilizado el nombre del sensor/parte de la dirección ip de mi raspberry/nombre del dato.
QoS, calidad del servicio, predeterminado 0.

- 0, entrega y olvida
- 1, al menos una vez
- 2, una y solo una vez

Retain (retener), si el broker retiene el mensaje o no retiene. Si retiene el mensaje, cada vez que me conecte al broker me mostrará el último mensaje.
Name, nombre descriptivo.

Realizamos los mismos para los datos de humedad y presión.

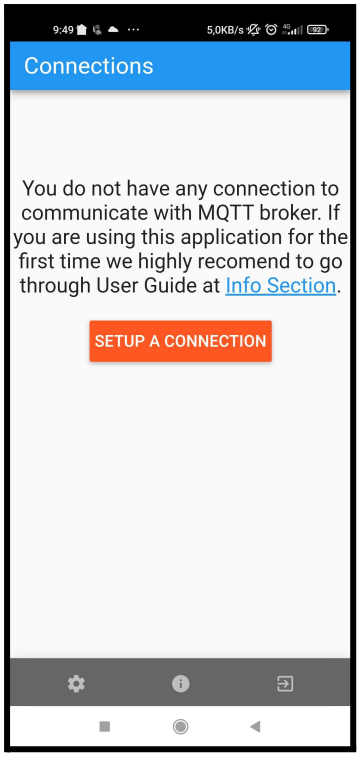


Ya tenemos los datos del sensor publicados en el broker.mqtt en la nube y ahora vamos a realizar la lectura desde un teléfono móvil.

Práctica, visualización de datos publicados en broker

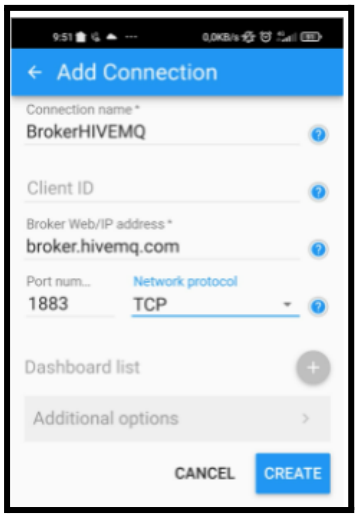
Existen diferentes app como cliente mqtt y poder visualizar los datos que están publicados en el broker hivemq.

Desde playstore instalamos la app **IoT MQTT Panel**  , esta aplicación le permite administrar y visualizar el proyecto IoT, basado en el protocolo MQTT.

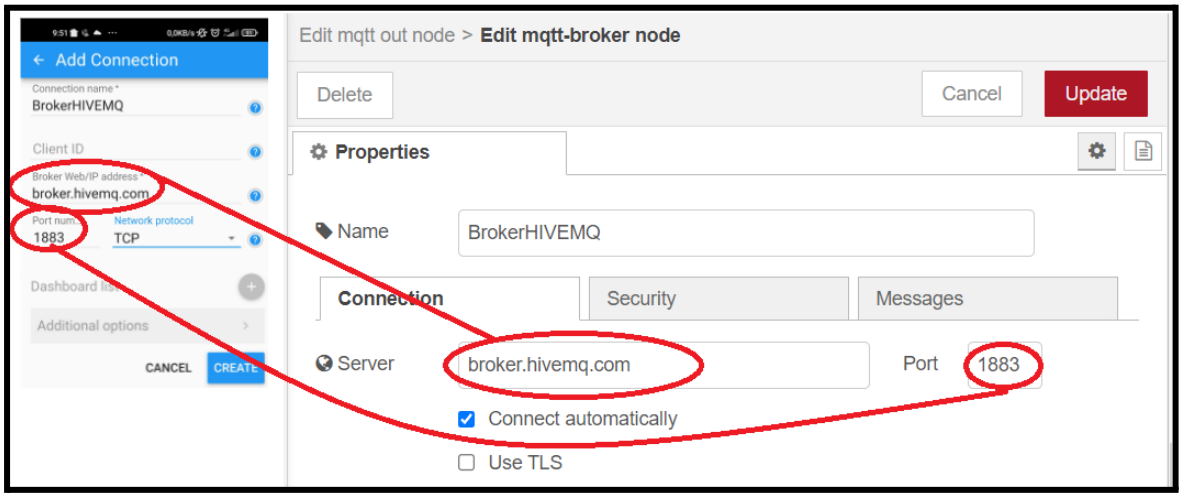


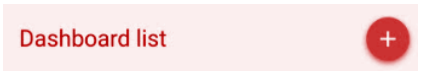
La primera vez que entramos aparece la siguiente pantalla para crear una conexión a un broker MQTT.

En primer lugar ponemos un nombre a la conexión, en cliente ID dejamos en blanco y se asignará uno aleatoriamente.
Broker web/ip address, ponemos el DNS o IP de la dirección del MQTT broker.
Port, 1883 (TCP),8883 TCP SSL),8080 (Websocket) o 8081(Websocket SSL).
Network procotol, por defecto es TCP pero puede cambiar a TCP (SSL), Websocket o Websocket (SSL).

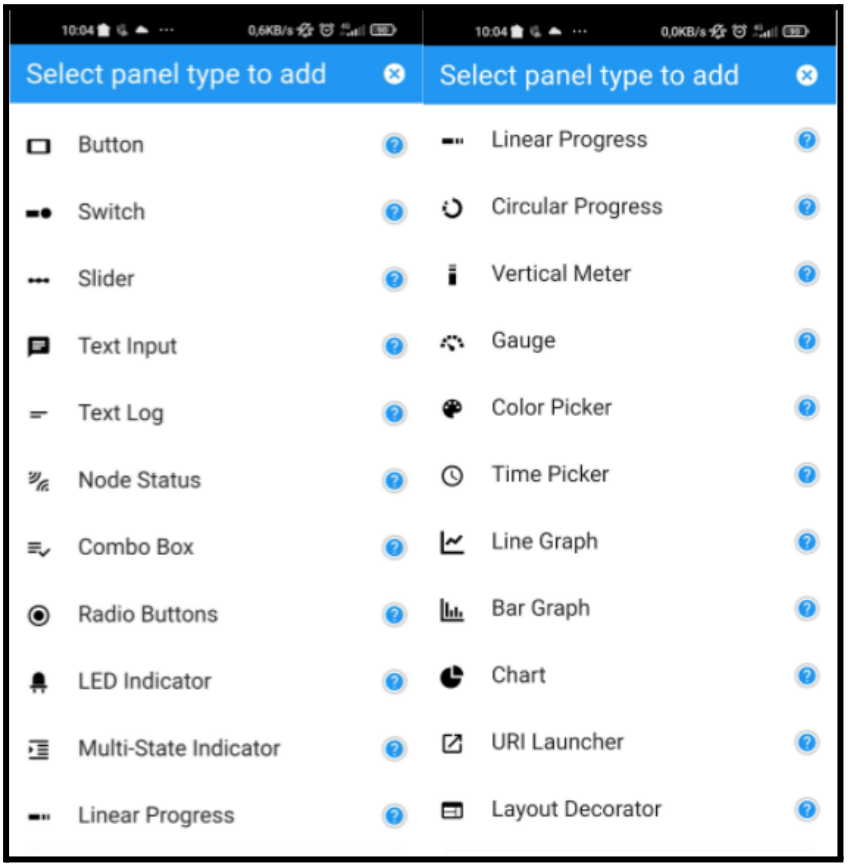


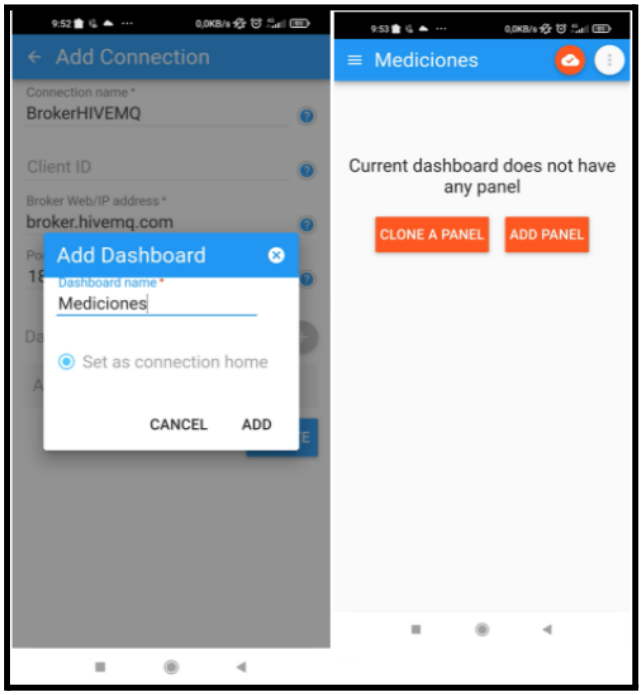
Estos datos de conexión tienen que ser iguales a los definidos en Node-RED en el nodo mqtt-broker.



Marcamos sobre 

para crear un tablero y añadimos paneles.

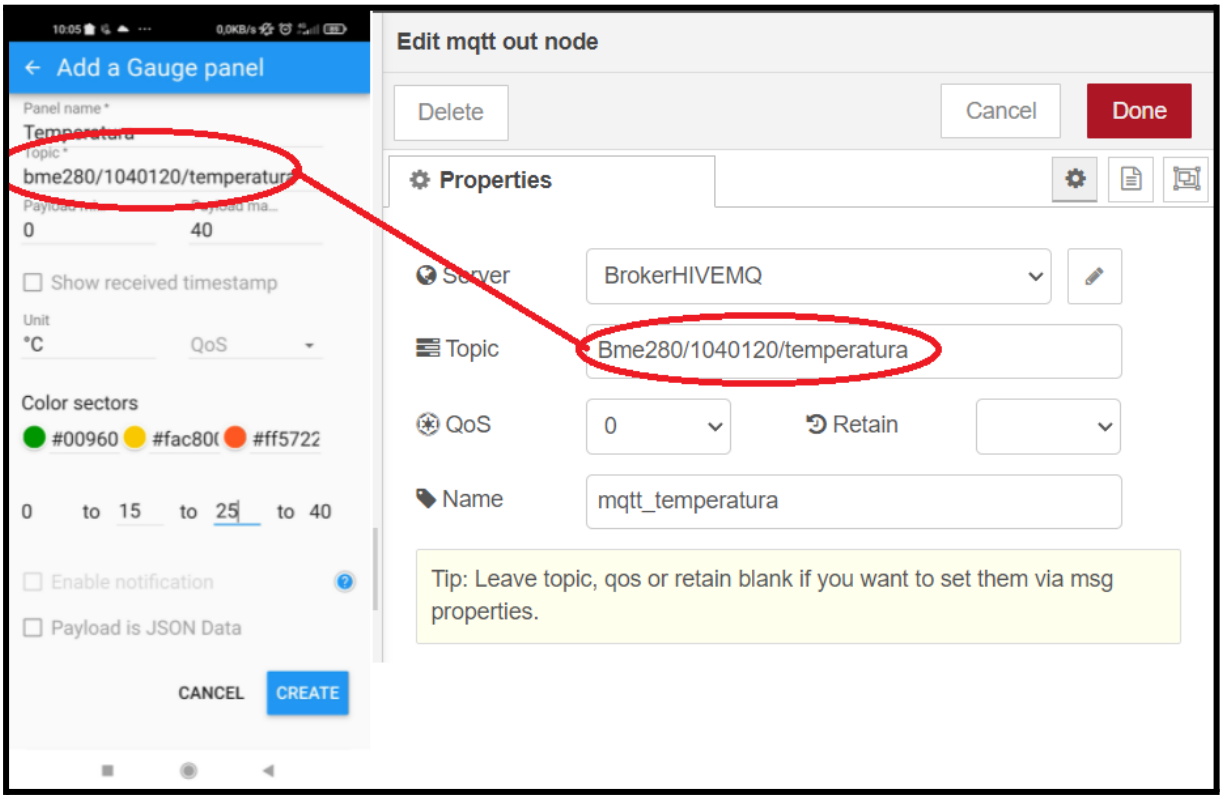




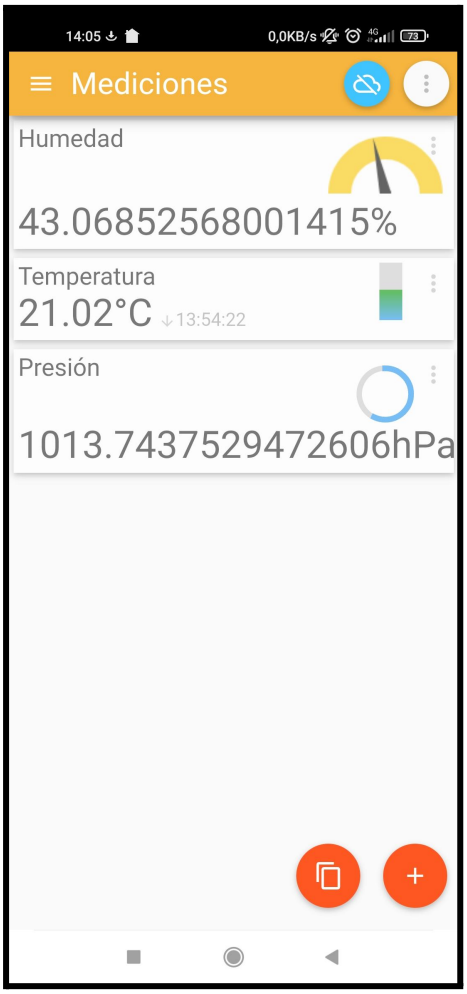
Seleccionamos el tipo de panel que deseamos mostrar.

Vamos a seleccionar Gauge.

Ponemos un nombre al panel y en **Topic** tiene que coincidir con el nombre de topic que hemos creado dentro del broker mqtt y en el que se publica la información. Podemos configurar tres colores diferentes dentro de unos rangos y establecer un valor mínimo y máximo.

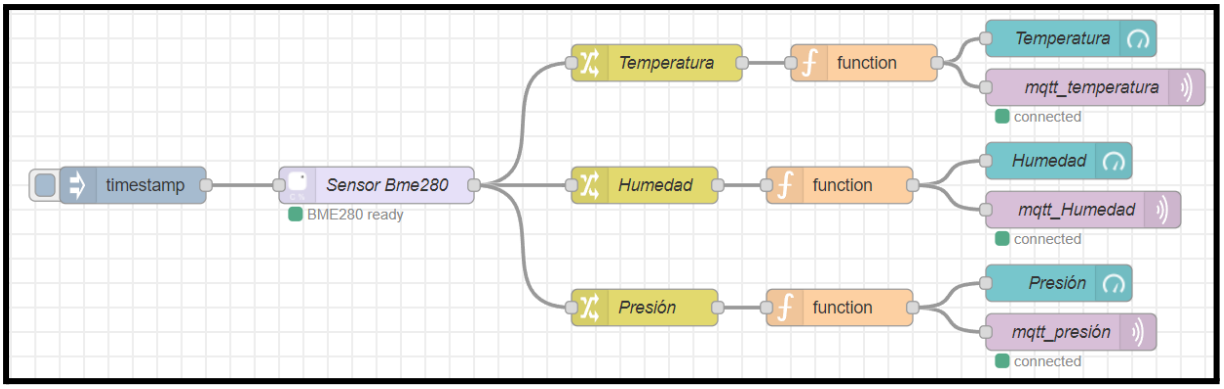


Hemos creado tres paneles y se mostrará así.

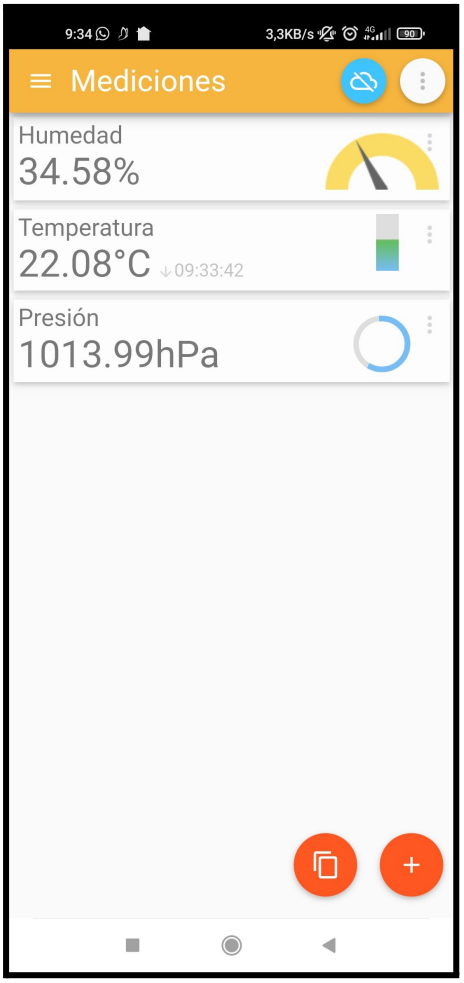


Dentro de esta app **IoT MQTT Panel** no se puede dar formato a la salida y se muestran los datos de igual forma a como se han subido a la nube.

Si deseamos dar formato a la salida modificaremos en Node-RED y añadimos un nodo tipo función para convertir los valores string a un formato numérico que hemos definido de 2 decimales.



Ahora los datos nos aparecen con el formato numérico de 2 decimales.



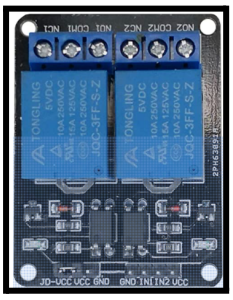
Práctica, leer datos sensor y activar actuadores

Continuando con la práctica en la que leíamos los datos del sensor **bme280**, añadiremos unos actuadores. Cuando la temperatura alcance 30ºC encenderemos un ventilador y si supera 35ºC se encenderá una bombilla roja de alarma.

Tenemos un ventilador USB

Una bombilla roja de 220 a 240V

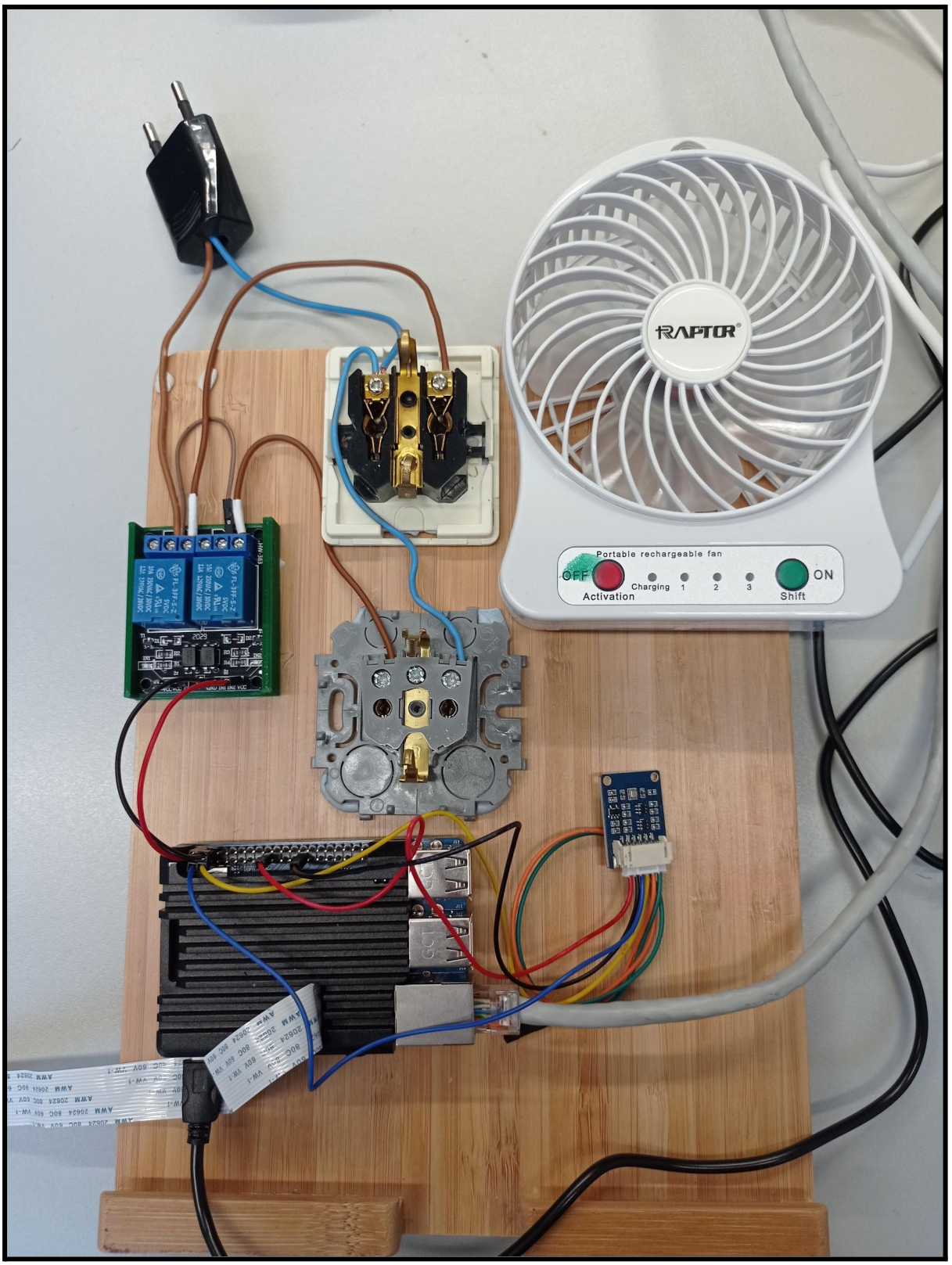
Relays 5V/In 220AC/Out



Características:

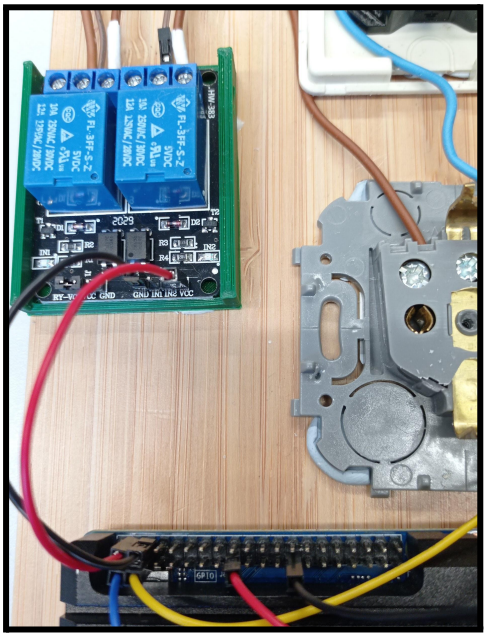
- Canales: 2 (independientes protegidos con Optoacopladores)
- Tensión de Alimentación: 5V
- Corriente de Salida: 10A
- Corriente de activación por relé: 15mA~20mA
- Aislamiento: Si
- LED indicador: Para cada canal

2 enchufes superficie

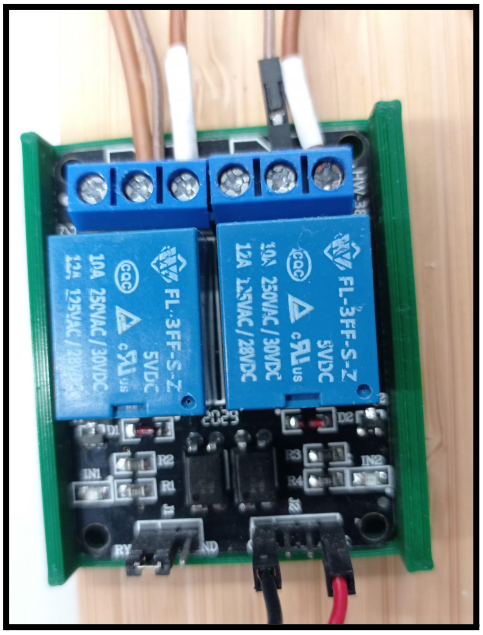


El relé es de 2 canales de 5 V y con él podemos controlar equipos o electrodomésticos de gran corriente.

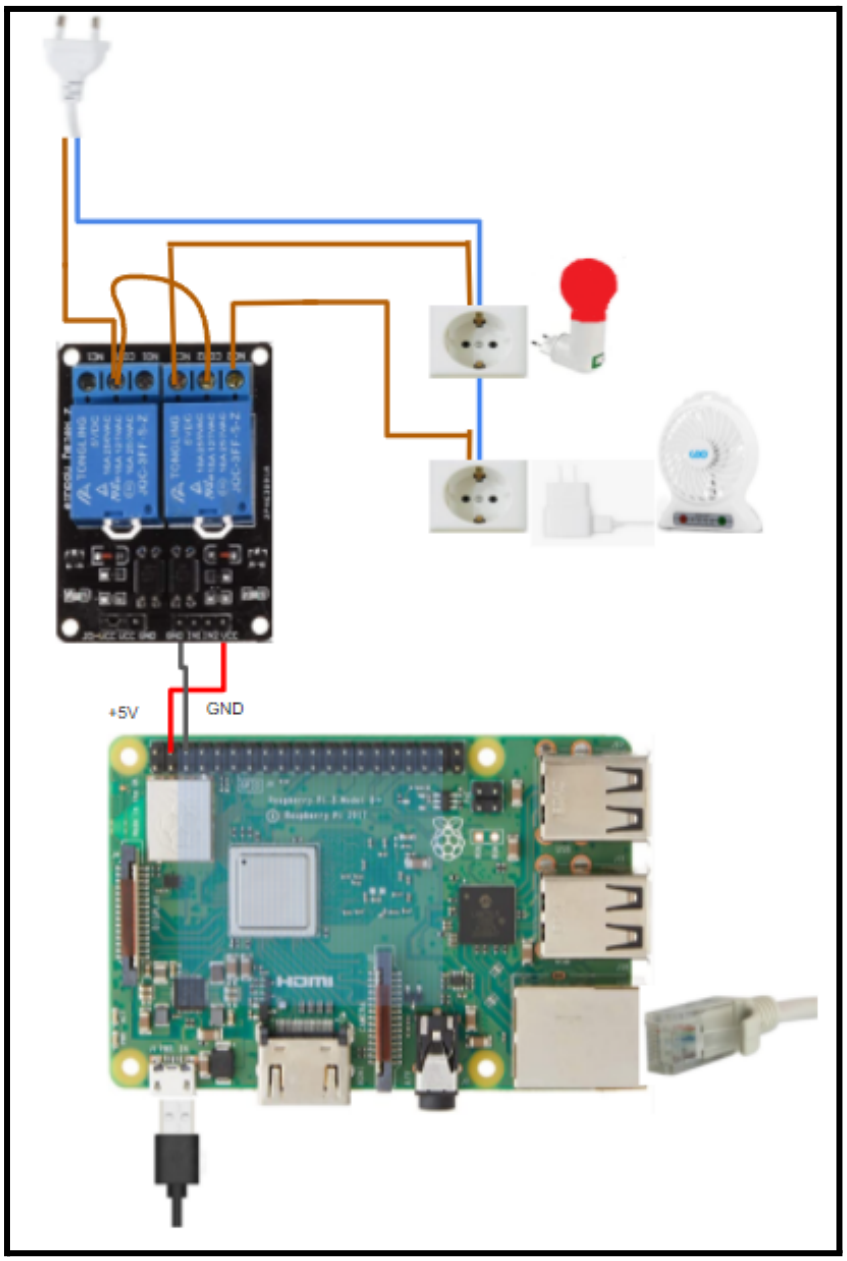
El relay que tiene una entrada de 5V me proporciona una salida de 10A que utilizaré para la alimentación de los dos enchufes.
Cableamos el relay a la Raspberry, como es de 5V utilizamos el pin 4 y el pin 6 para GND.



Unimos cada canal del relé a un enchufe, utilizamos la salida normalmente abierto y realizamos un puente entre los dos canales.



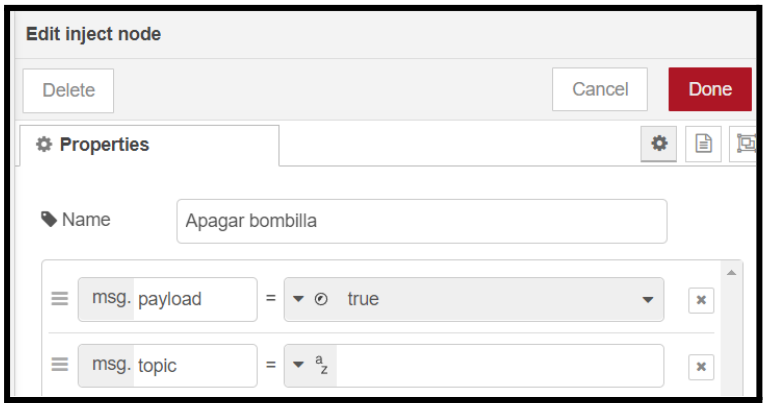
El esquema de cableado es el que se muestra en la imagen siguiente.



Con Node-RED probaremos a dar tensión a cada enchufes desde la raspberry.

Creamos un nodo **inject** que llamaremos Encender bombilla

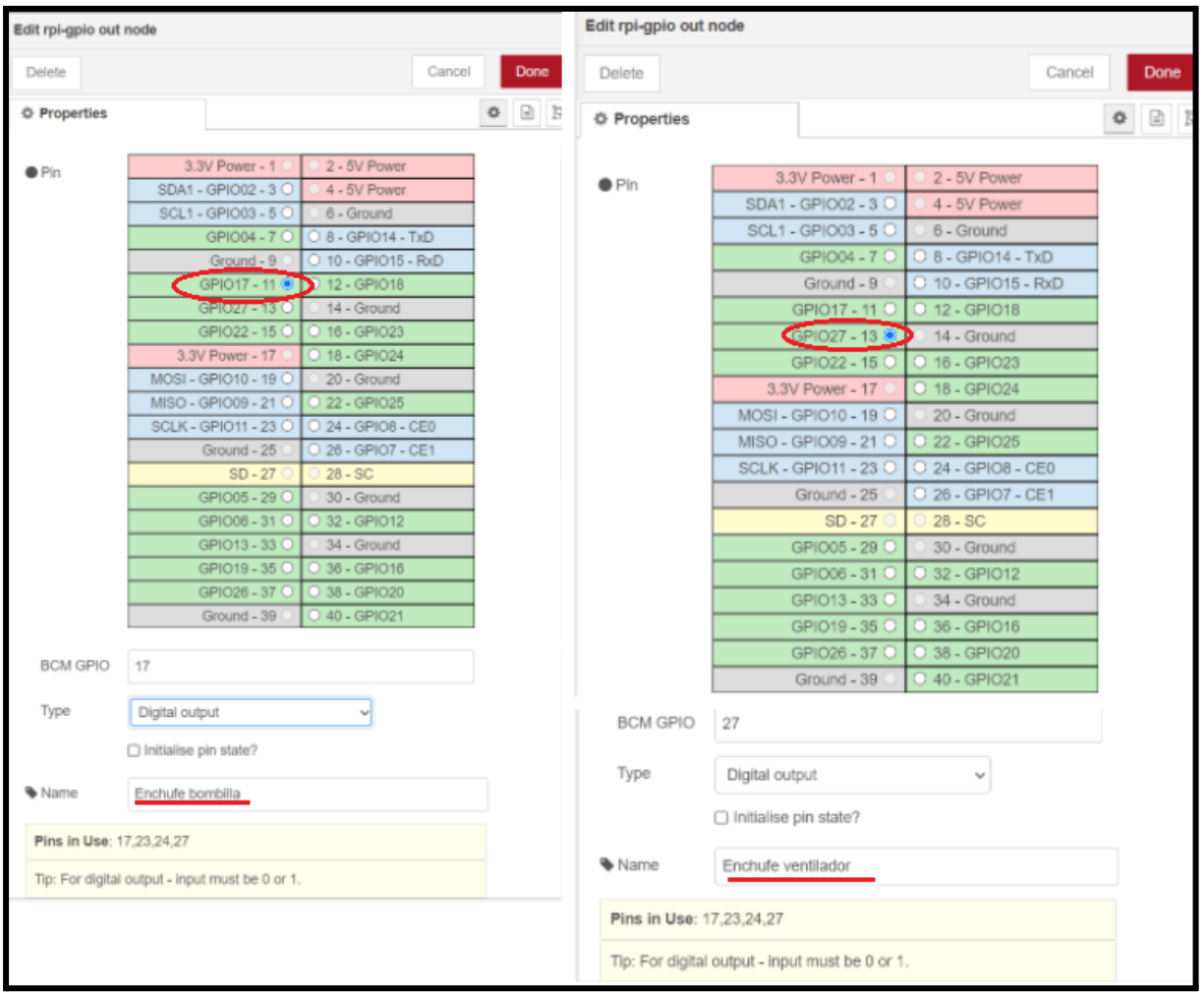
Otro nodo **inject** para apagar la bombilla.

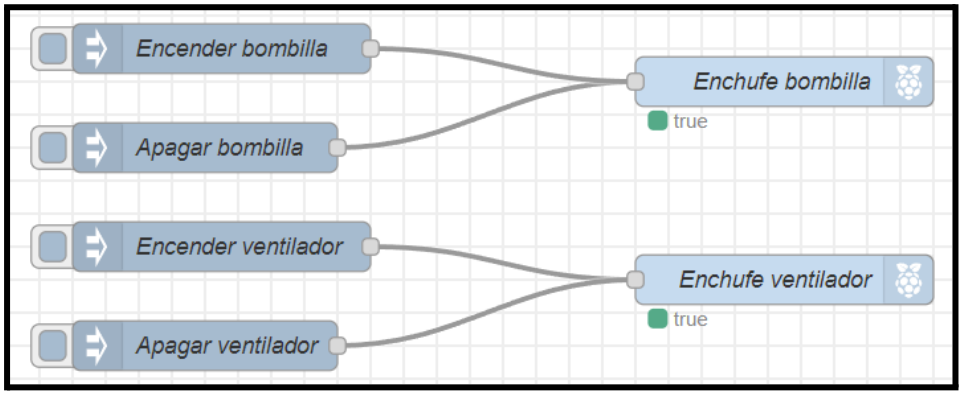


Creamos otros dos similares para encender y apagar el ventilador.

Necesitamos dos nodos **rpi- gpio out** que son la salida de las raspberry hacia el relé.

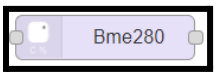
Para el enchufe de la bombilla hemos utilizado el pin 11 (GPIO17) y para el enchufe del ventilador el pin 13 (GPIO27).



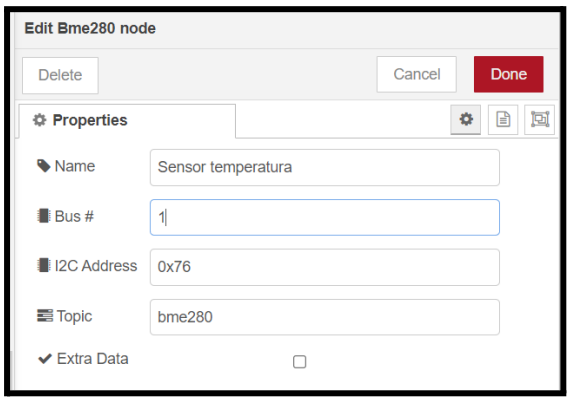


Con este diseño ya podemos encender y apagar la bombilla y el ventilador.

Ahora como tenemos un sensor de temperatura unido a la raspberry vamos a realizar el diseño en node-red para que si se pasa de 23ºC de temperatura se encienda el ventilador y por debajo de dicha temperatura se debe apagar.
También añadiremos que si la temperatura pasa de 24ºC se encienda la bombilla roja.

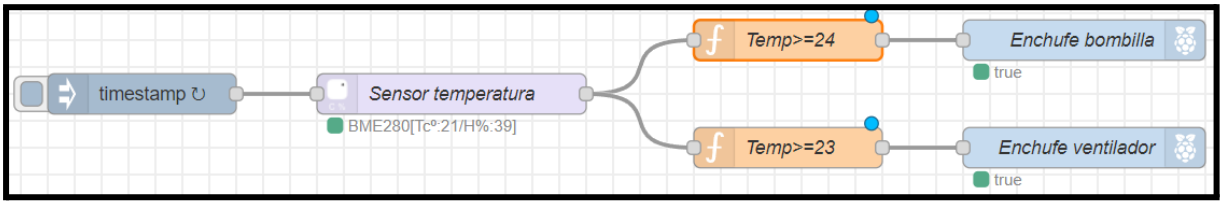
Insertamos el nodo Bme280 

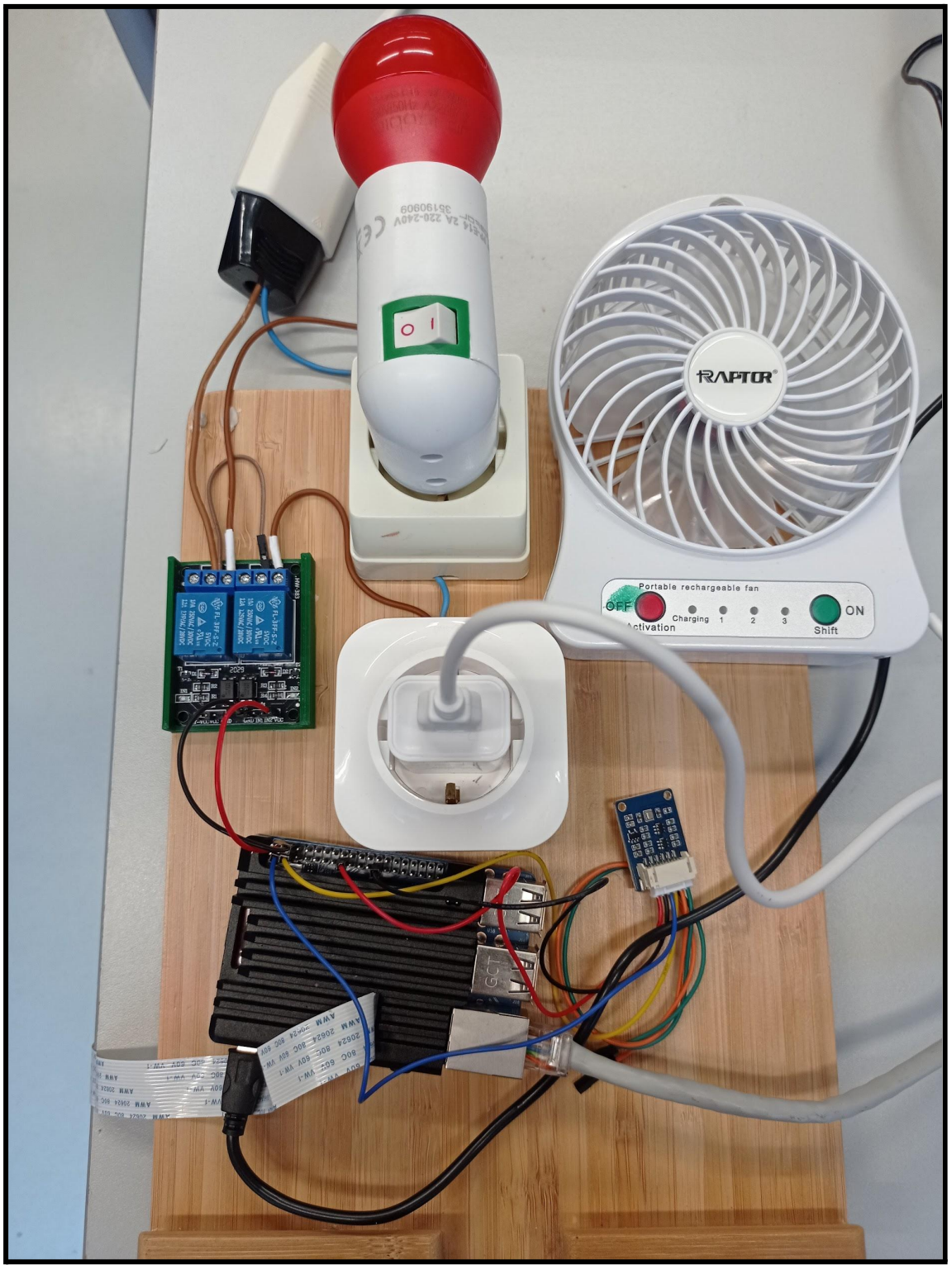
y damos doble clic para propiedades.



Ponemos un nombre y en Bus el número de bus que utilizaremos en la raspberry, aquí se utiliza el 77.

Ahora añadimos una función para activar el relé que enchufe la bombilla si la temperatura es >= 24ºC y otra función que active el relé que enchufa el ventilador si la temperatura es >= 23ºC.





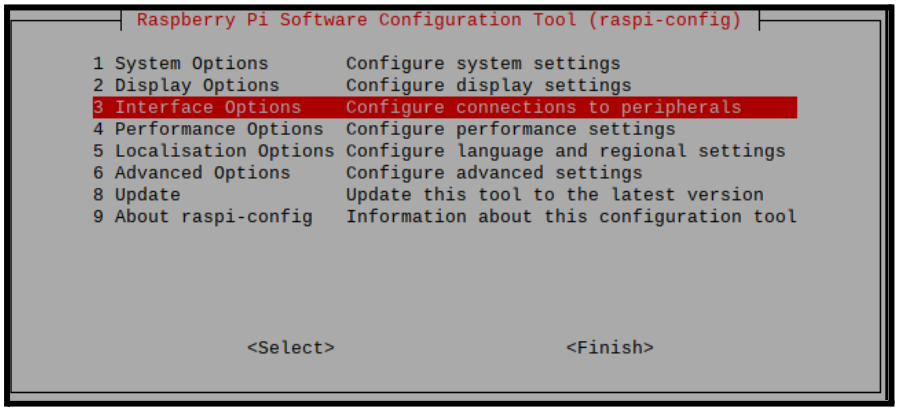
Vamos a continuar nuestra práctica añadiendo un sensor de temperatura RS485, este se comunicará con la raspberry usando el protocolo de comunicación Modbus.

Modbus sobre serie RS485

Si queremos comunicar la RaspBerry con un módulo externo necesitamos usar el puerto serie y será necesario activarlo en la RaspBerry.

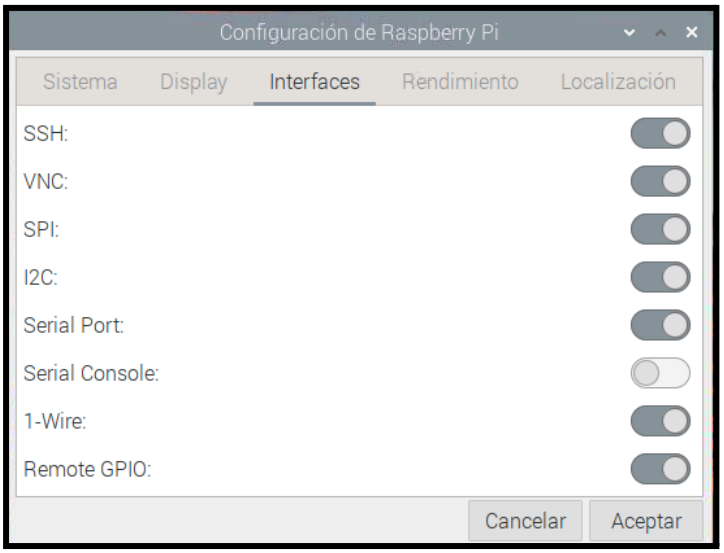
De forma predeterminada, el puerto serie de la Raspberry Pi está conectado a la consola para permitir el control de la Raspberry Pi sin una interfaz gráfica. Esto es incompatible con el uso estándar del puerto serie.

Para hacer esto, abra una terminal en su Raspberry Pi, ya sea directamente o vía SSH, luego ejecute el comando **sudo raspi-config** para acceder a la herramienta de administración de configuración de Raspberry.



Una vez iniciado **raspi-config**, elija **Interfacing options,** después **Serial**. Responder **No** a la parte que pregunta si desea activar un shell a través de la conexión en serie, luego **Yes** a la parte que pregunta si desea activar el puerto de hardware. Validar, hacer Finish, luego responda que sí a la pregunta que le pregunta si desea reiniciar la Raspberry Pi.

O se puede acceder desde la interfaz gráfica. En el menú principal seleccionamos preferencias y configuración y pulsamos la pestaña **Interfaces**. Activamos **Serial Port** y desactivamos **Serial Console**.



En los modelos de RaspBerry 3 y posteriores debemos hacer una configuración adicional, hay que desactivar el bluetooth.

La lectura interna de datos en serie utiliza lo que se denomina UART. Todos los modelos de Raspberry tienen al menos 2 UART, el número 0, que es un UART normal y el número 1, que es un mini UART con capacidades mucho más limitadas y muy restrictivas.

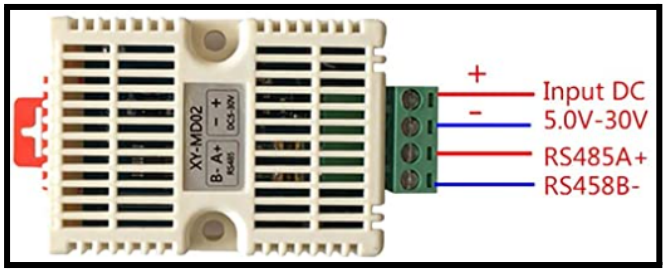
UART 0 se usaba de forma predeterminada, pero desde el cambio a la versión 3, ahora se usa para administrar el chip bluetooth. Si está utilizando una Raspberry Pi 3 o un modelo posterior, deberá modificar este comportamiento y desactivar el bluetooth.

Para hacer esto abra una terminal de comandos y ejecute lo siguiente:

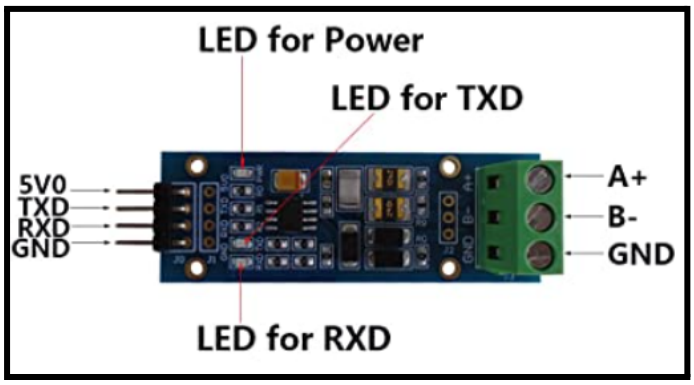
```
echo "dtoverlay=disable-bt" | sudo tee -a /boot/config.txt
sudo systemctl disable hciuart
sudo reboot
```

El puerto serie Raspberry tiene dos GPIO, uno para recibir datos, RXD, y otro para enviar TXD.

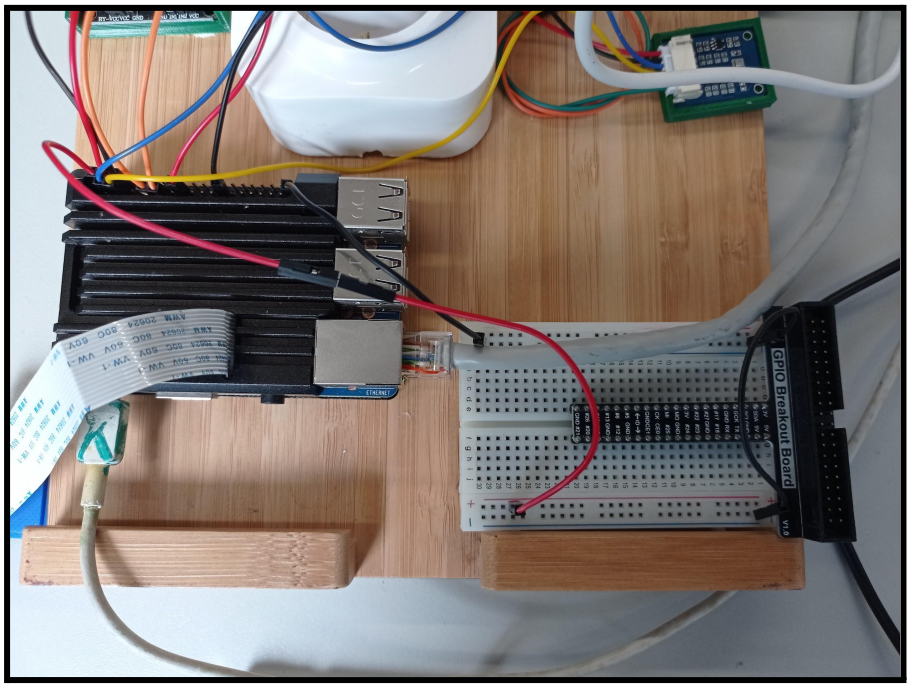
Tenemos un sensor de temperatura que utiliza el estándar de comunicación RS485.



Un convertidor de señal RS485 a señal serie TTL (lógica transistor-transistor).



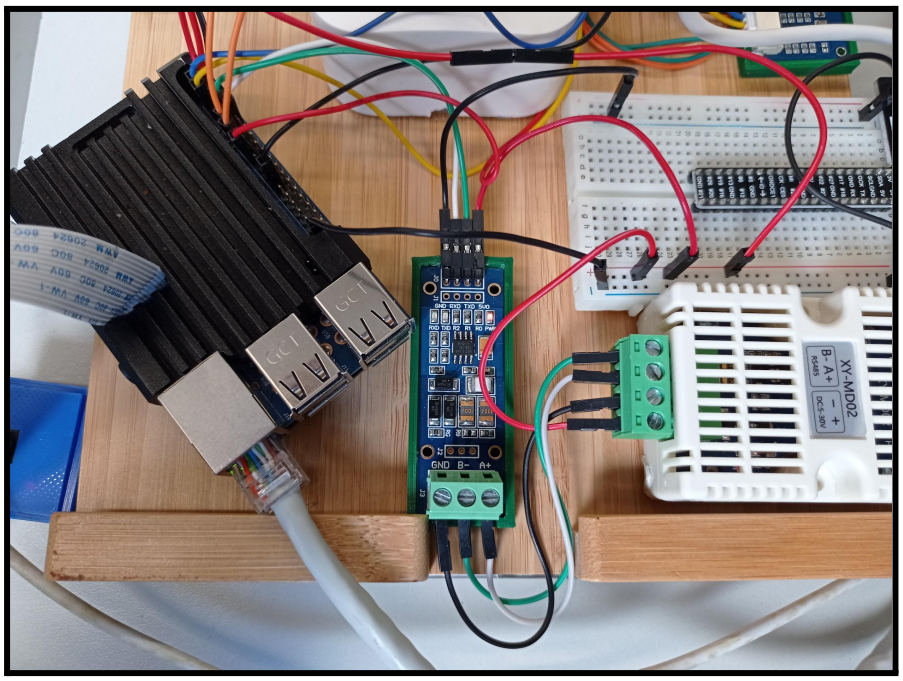
Añadimos una breadboard para poder ampliar las salidas de voltaje de la raspberry.



Conectamos el convertidor RS485 a TTL a la raspberry utilizando el pin 8 (GPIO14-TXD) es el cable blanco para entrada datos y el pin 10 (GPIO15-RXD) es el cable verde para la salida de datos.

En cable de salida de la raspberry conectado al pin 8 (GPIO14-TXD) es la entrada en el conversor RS485 y lo conectaremos al marcado como RXD y el cable de entrada de la raspberry conectado al pin 10 (GPIO15-RXD) lo conectaremos al marcado como TXD.

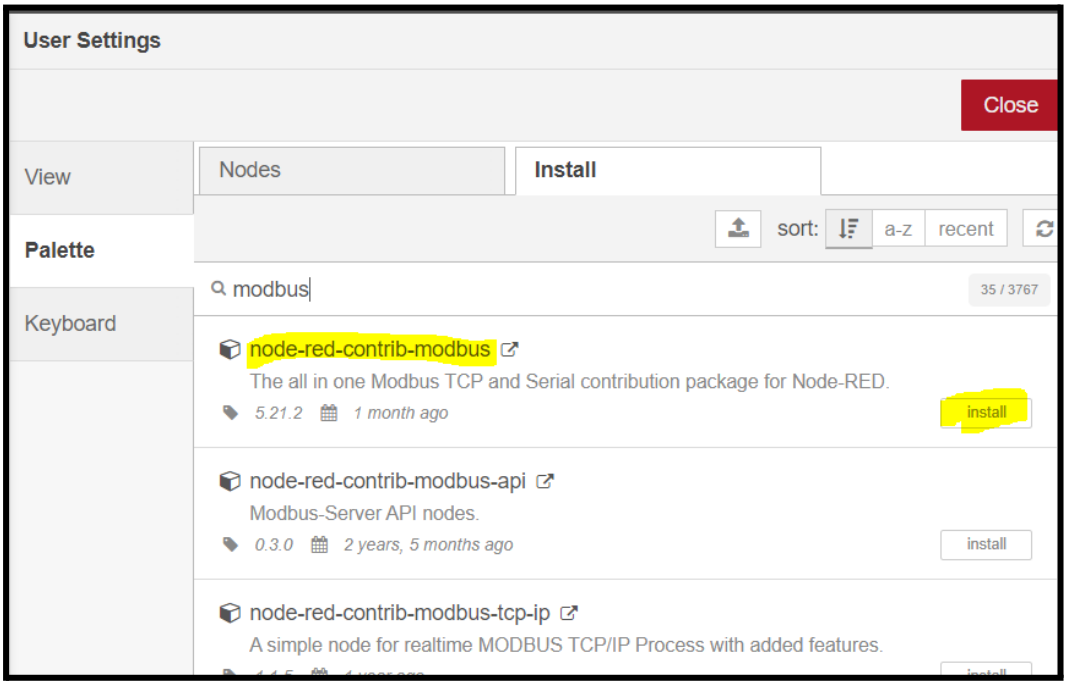
GND es el cable negro y cable rojo es corriente a 5V.



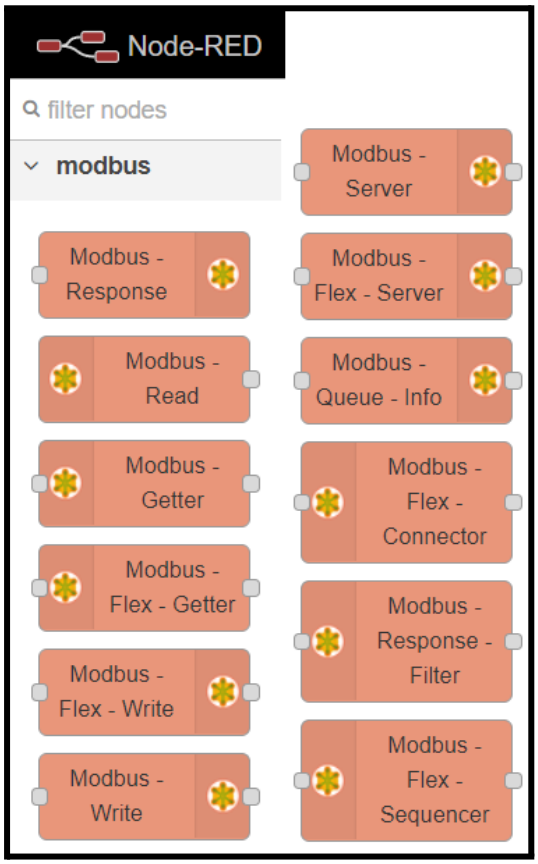
Ahora conectamos el sensor RS485 al convertidor de señal a través de las salida A+ y B-. La salida A+ (cable blanco) se conecta a la entrada A+ del convertidor y el B- (cable verde).

Ahora que todo está cableado, probaremos su funcionamiento con Node-RED.

Necesitamos añadir el paquete modbus o desde la consola de comando ejecutando **npm install node-red-contrib-modbus** o desde dentro de Node-Red accediendo al menú general, Manage Palette y en la pestaña install buscamos modbus.



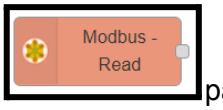
Seleccionamos node-red-contrib-modbus y pulsamos install.
Se han añadido los siguientes nodos:



Deberemos acceder a la documentación del sensor para saber que funciones soporta.
http://www.sah.rs/media/sah/techdocs/xy-md02-manual.pdf

Modbus Protocol	
Function Code	
Command Register	Funciotn
0x03	Read keep register
0x04	Read input register
0x06	Write a single keep register
0x10	Write more keep registers

Register Type	Register Address	Register Contents	Bytes
Input Register	0x0001	Temperature	2
	0x0002	Humidity	2
Keep Register	0x0101	Device Address	2
	0x0102	Baud Rate: 0:9600 1:14400 2:19200	2
	0x0103	Temperature Correction -10℃~10℃	2
	0x0104	Humidity Correction -10%RH~10%RH	2

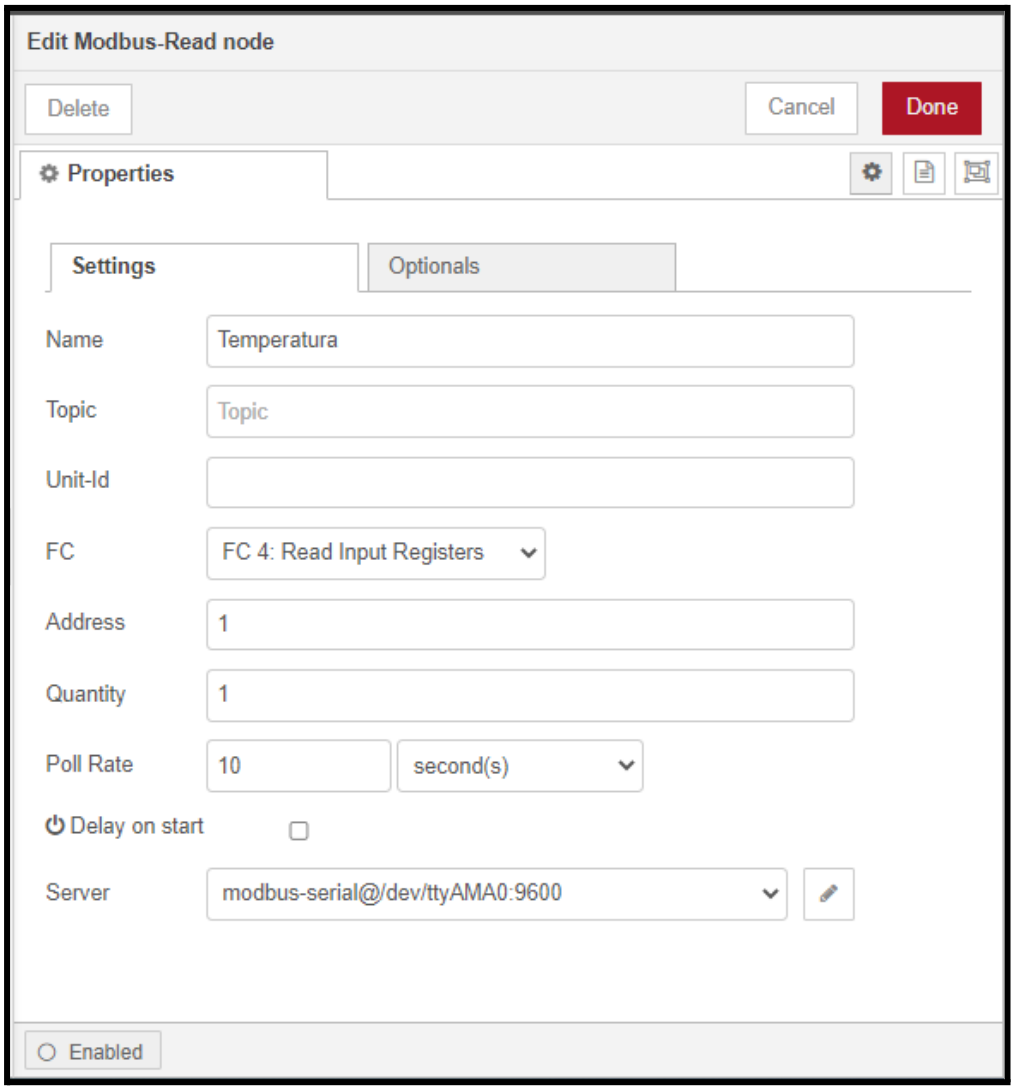
Insertamos un nodo **Modbus - Read** 

para poder leer los registros del sensor de temperatura XY-MD02 que utiliza conexión RS485.

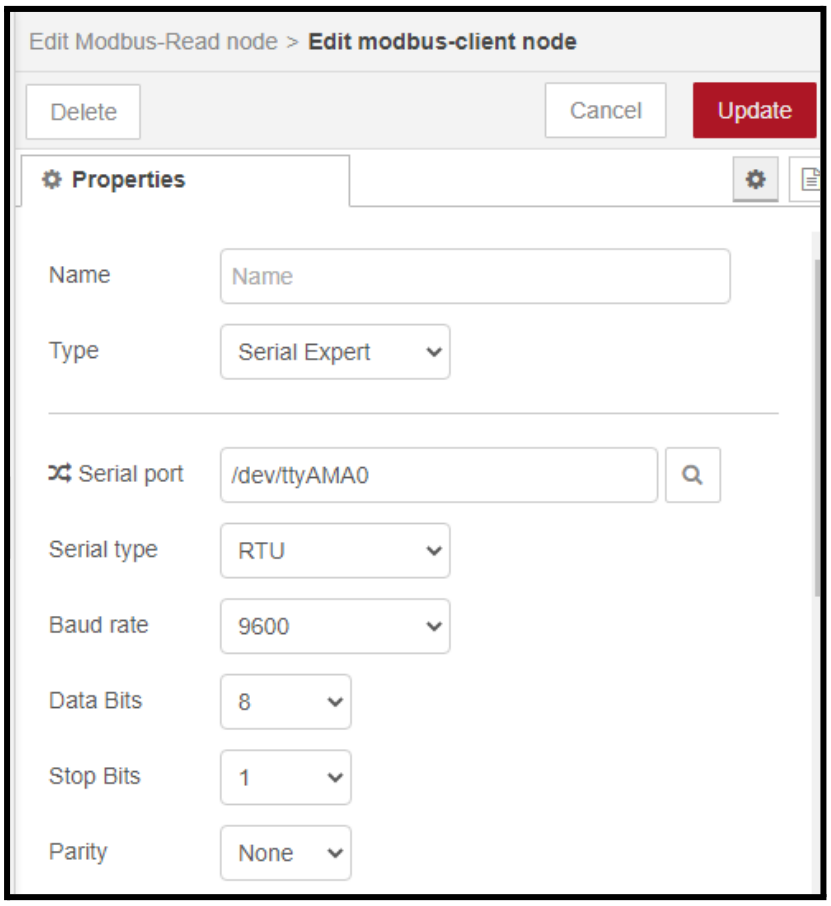
El nodo Modbus - Read soporta 4 funciones:

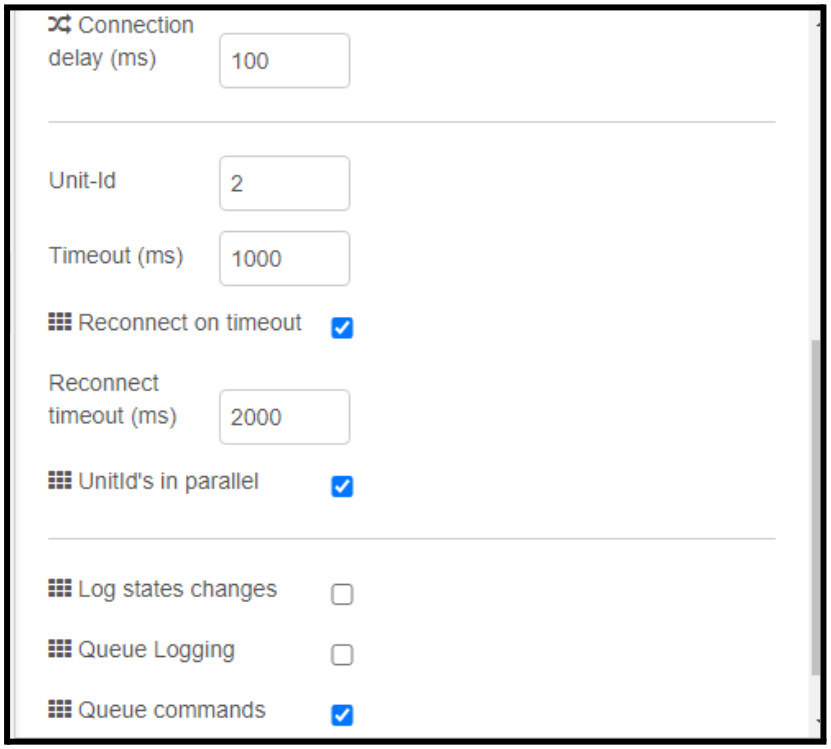
- FC 1: Read Coil Status (Leer estado sensor)
- FC 2: Read Input Status (Leer estado entrada)
- FC 3: Read Holding Registers (Leer registros de sensor)
- FC 4: Read Input Registers (Leer registros de entrada)

Unit-Id (0..255 tcp | 0..247 serial) - déjelo vacío, de lo contrario anula el Unit-ID predeterminado de la configuración del cliente.
FC: Elija el código de función (FC 4) del menú desplegable.
Address: Seleccione la dirección de inicio de bobina/entrada/registro (0:65535)
Quantity: La cantidad de bobinas/entradas/registros que se leerán desde la dirección de inicio.
Poll Rate: Configure una tasa de sondeo (mayor que cero) y la unidad de tiempo.



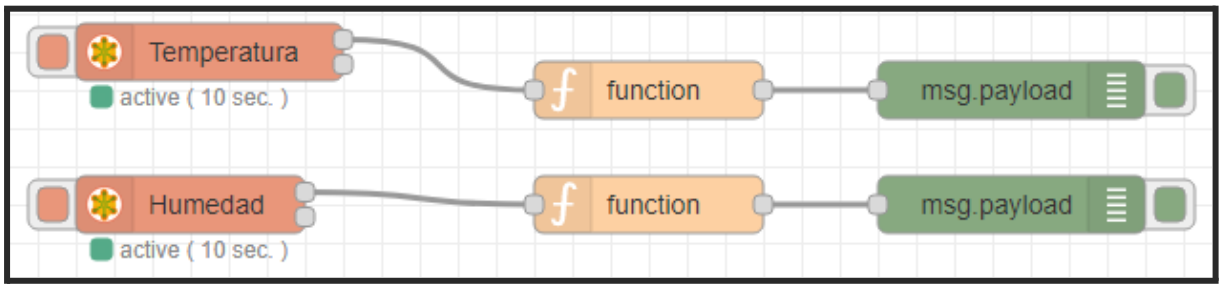
Server. Elija o edite la configuración de la conexión Modbus.


Edit Modbus-Read node > Edit modbus-client node
Delete
Cancel
Update
Properties
Name
Name
Type
Serial Expert
Serial port
/dev/ttyAMA0
Serial type
RTU
Baud rate
9600
Data Bits
8
Stop Bits
1
Parity
None

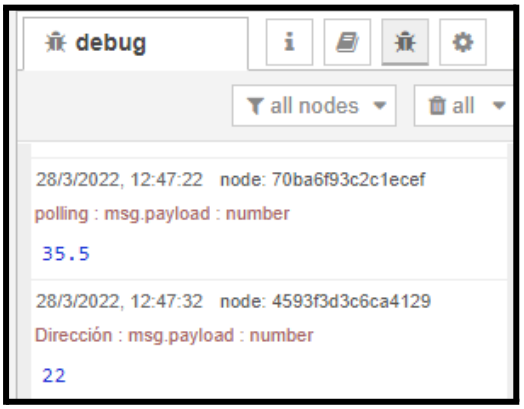

Connection delay (ms)
100
Unit-Id
2
Timeout (ms)
1000
Reconnect on timeout
Reconnect timeout (ms)
2000
UnitId's in parallel
Log states changes
Queue Logging
Queue commands

Serial Port: /dev/ttyAMA0, es para Bluetooth pero, recordamos, que hemos modificado la configuración de la Raspberry para poder utilizarlo como puerto serie.
Type: seleccionamos Serial Expert o Serial.
Baud Rate: 9600, este dato nos viene en la documentación del sensor.
Unit-Id: predeterminado 1 [serial] o 0 [tcp], para establecer una ID de unidad para todos los nodos sin ID de unidad.
Es el número de esclavo.
Aquí el valor es 2 porque previamente ha sido modificado el valor por defecto de 1.
El resto de valores los dejamos por defecto.

Hemos creado otro nodo para leer los datos de la humedad.

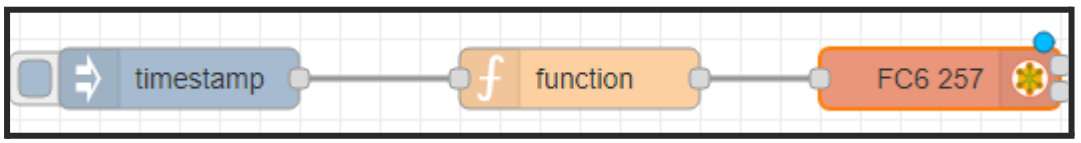


En el debug podemos ver la lectura de los datos del sensor.



Escribir dato en esclavo

En la siguiente imagen se muestra como se escribe un dato en el esclavo, aquí vamos a cambiar el número de esclavo.
Utilizamos un nodo inject para su activación, un nodo función donde ponemos msg.payload=”2” y se une a un nodo modbus-write.



En las propiedades del nodo modbus-write debemos seleccionar:

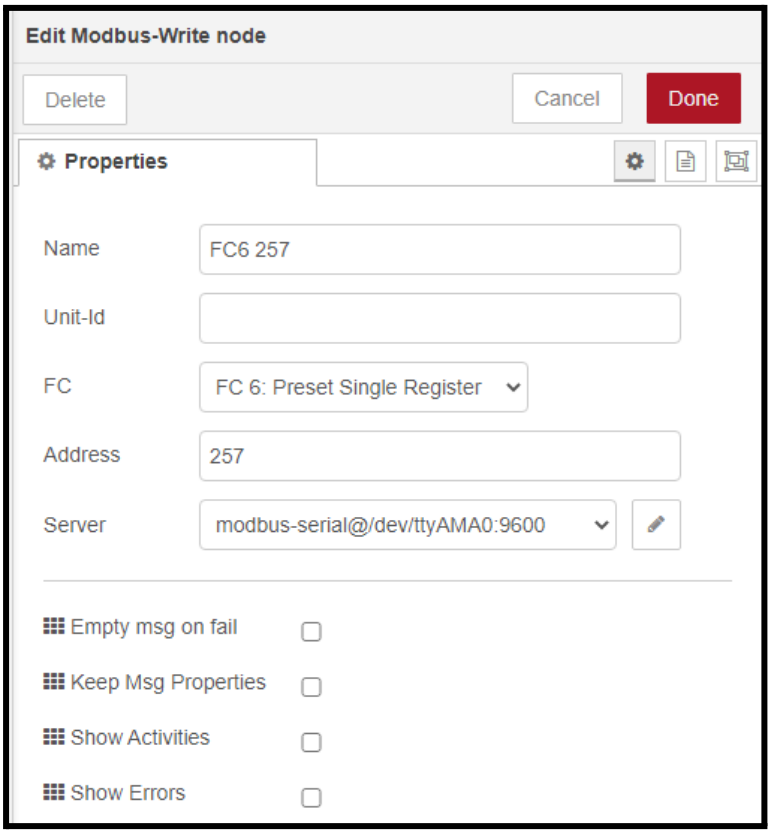
FC: Los códigos de función admitidos actualmente incluyen:

- FC 5:msg.payload debe tener un valor de 1 o 0 o verdadero o falso.
- FC 6: msg.payload debe ser un valor único entre 0:65535.
- FC 15:msg.payload debe ser una cadena de valores separados por comas verdadero o falso cada uno.
- FC 16: msg.payload debe ser una cadena de valores separados por comas entre 0:65535 cada uno.

Unid-id: lo dejamos en blanco para que reciba el valor predeterminado del servidor.
Address: seleccione la dirección, en las descripción del sensor nos dice que la dirección es 257 (0101 hexadecimal=257 decimal).
Vemos que en la dirección 0103 se podría poner una corrección a la temperatura y en la 0104 una corrección de la humedad.

Keep Register	0x0101	Device Address	2
	0x0102	Baud Rate: 0:9600 1:14400 2:19200	2
	0x0103	Temperature Correction -10℃~10℃	2
	0x0104	Humidity Correction -10%RH~10%RH	2

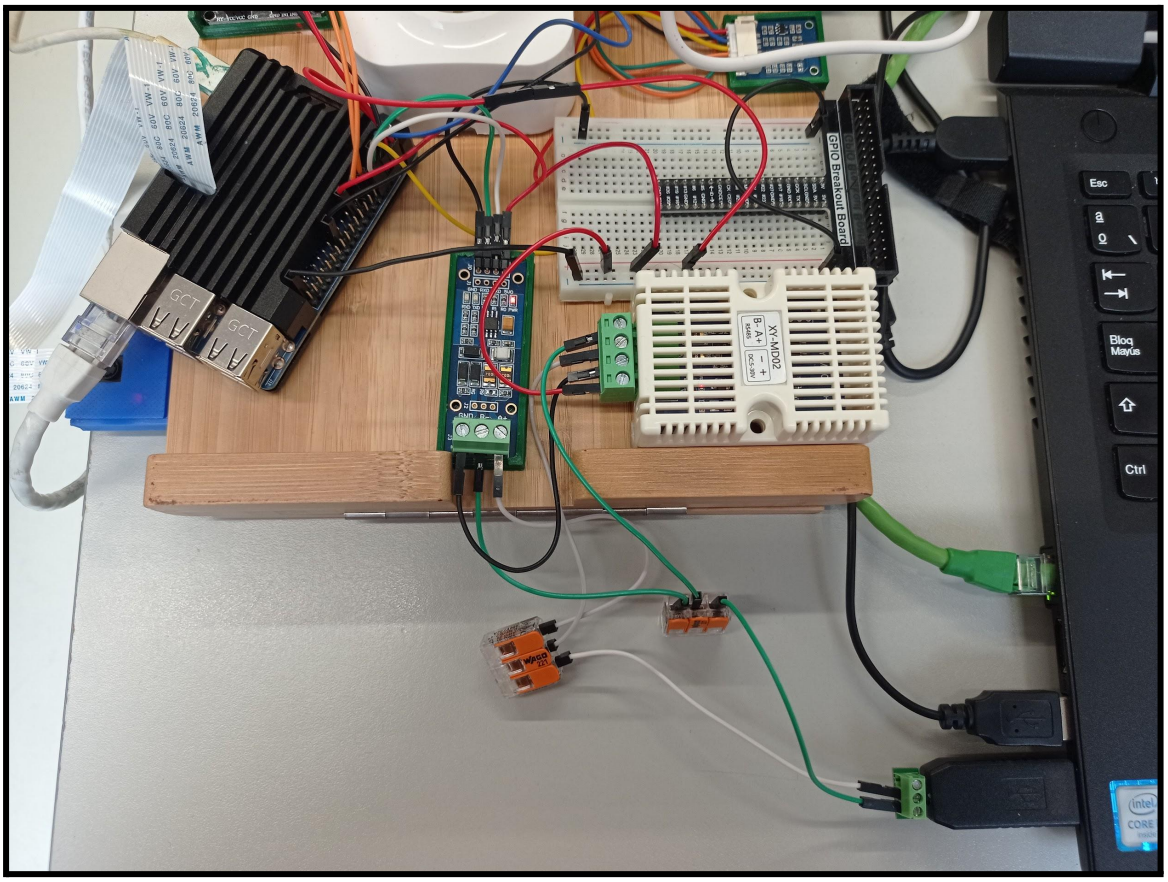
Server: Elija o edite la configuración del servidor serie/TCP Modbus para especificar el servidor al que conectarse.



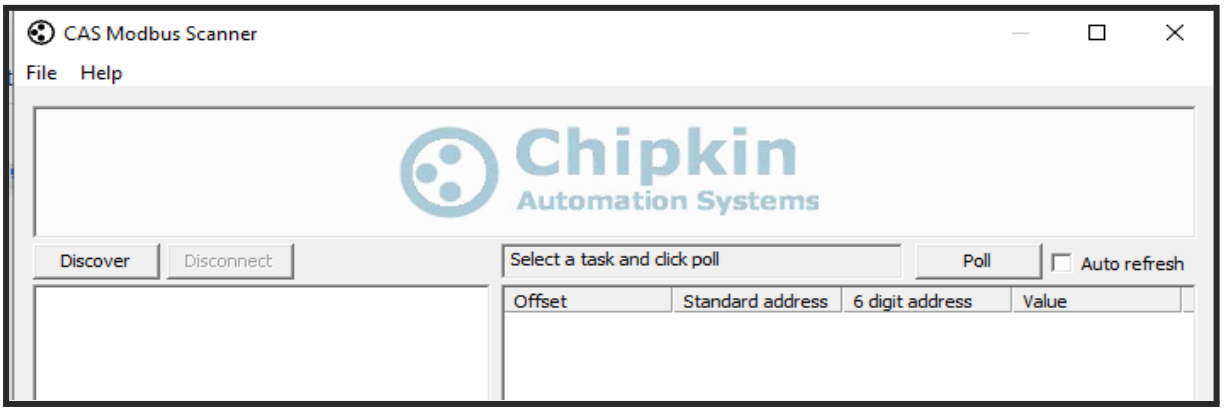
Modbus, conversor USB a RS485

En esta práctica vamos a tener acceso al sensor RS485 desde nuestro ordenador a través de un puerto usb.

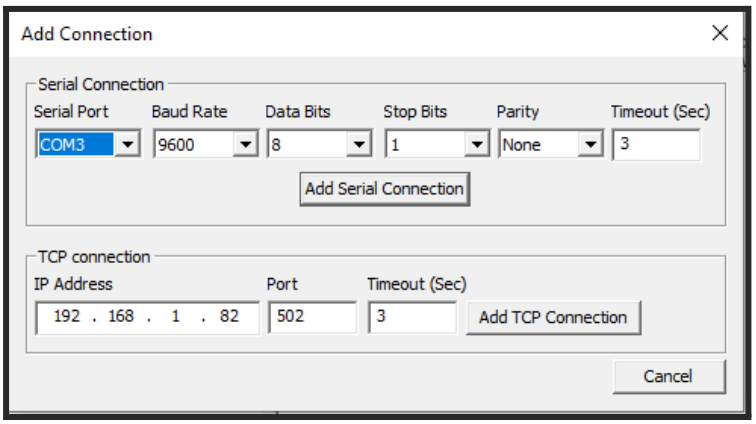
Utilizamos un conversor USB a RS485 y uniremos al sensor puenteando el cableado que va al conversor RS485 a TTL y que se une a la Raspberry.



Una vez conectado, desde el ordenador podemos utilizar un escáner modbus como CAS para descubrir los esclavos conectados en serie a la Raspberry.

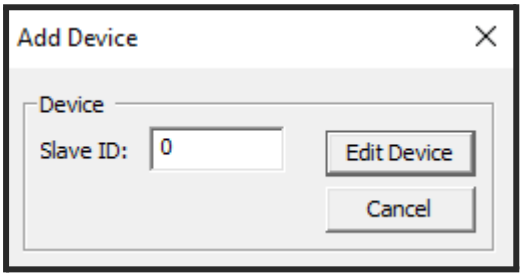


Pulsaremos el botón derecho para añadir una conexión.

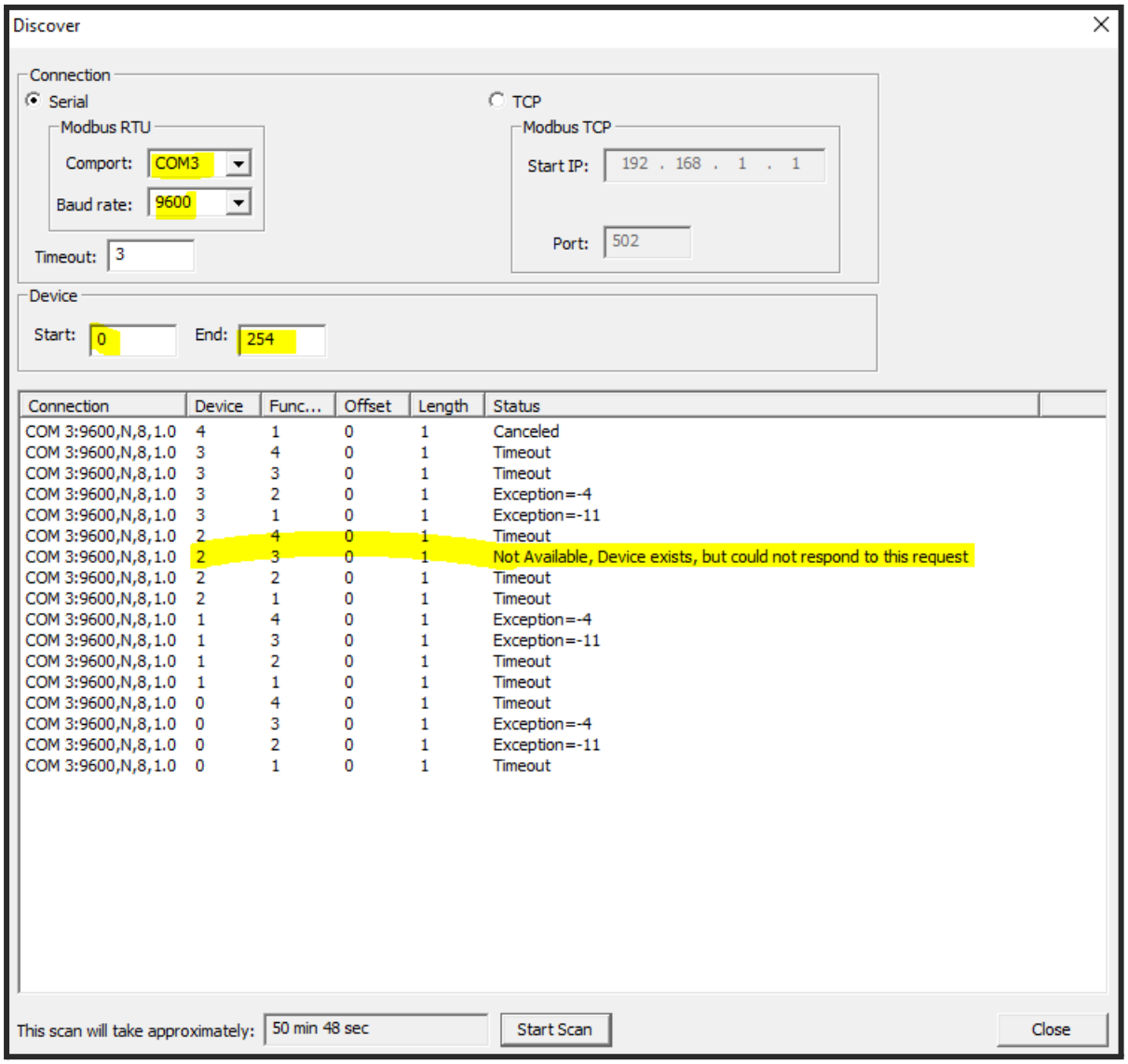


Desde el administrador de dispositivos veremos que puerto está utilizando el usb. Definimos el resto de datos de la conexión serie y pulsamos Add Serial Connection.

Nos aparece la conexión y ahora estando marcada la conexión pulsamos botón derecho del ratón para añadir un esclavo. Tiene que existir por lo menos un esclavo para poder realizar el escaner. Añadimos el esclavo 0.

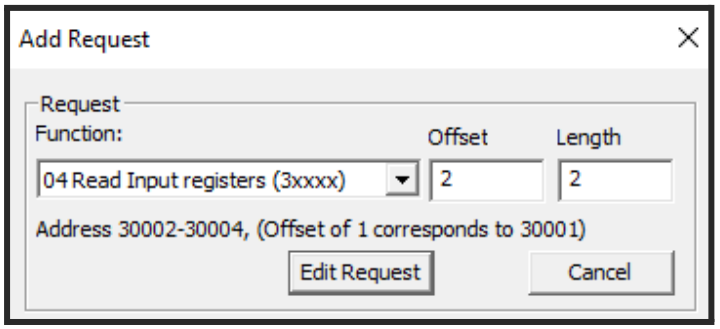


Ahora pulsamos el botón Discover para realizar un descubrimiento de que esclavos tenemos escuchando.

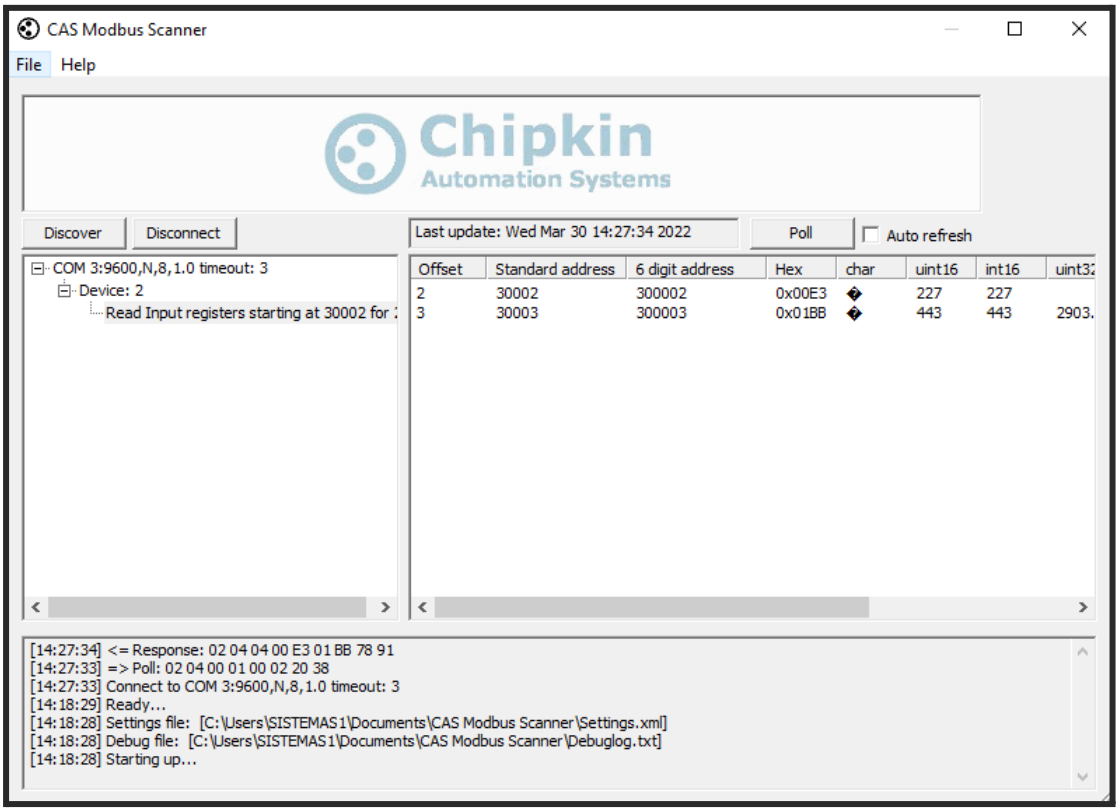


Seleccionamos para descubrir con los mismos datos que hemos creado la conexión.
Pulsamos Start Scan.
Nos realiza un escaneo de dispositivos y vemos que el dispositivo 2 existe.

Ahora debemos añadir el dispositivo 2 y una función 4 para ver los datos del sensor. Offset es la dirección.



Si pulsamos el botón Poll podemos ver la lectura del sensor, temperatura y humedad.



Hemos visto como utilizando CAS descubrimos a los esclavos.

Accediendo a la Raspberry por SSH y utilizando **mbpoll** en la línea de comando podemos comunicarnos con el esclavo ModBus (RTU o TCP).

Para su instalación en una Raspberry debemos realizar lo siguiente:

```
wget -O- http://www.piduino.org/piduino-key.asc | sudo apt-key add -
echo 'deb http://raspbian.piduino.org stretch piduino' |
sudo tee /etc/apt/sources.list.d/piduino.list
sudo apt update
sudo apt install mbpoll
```

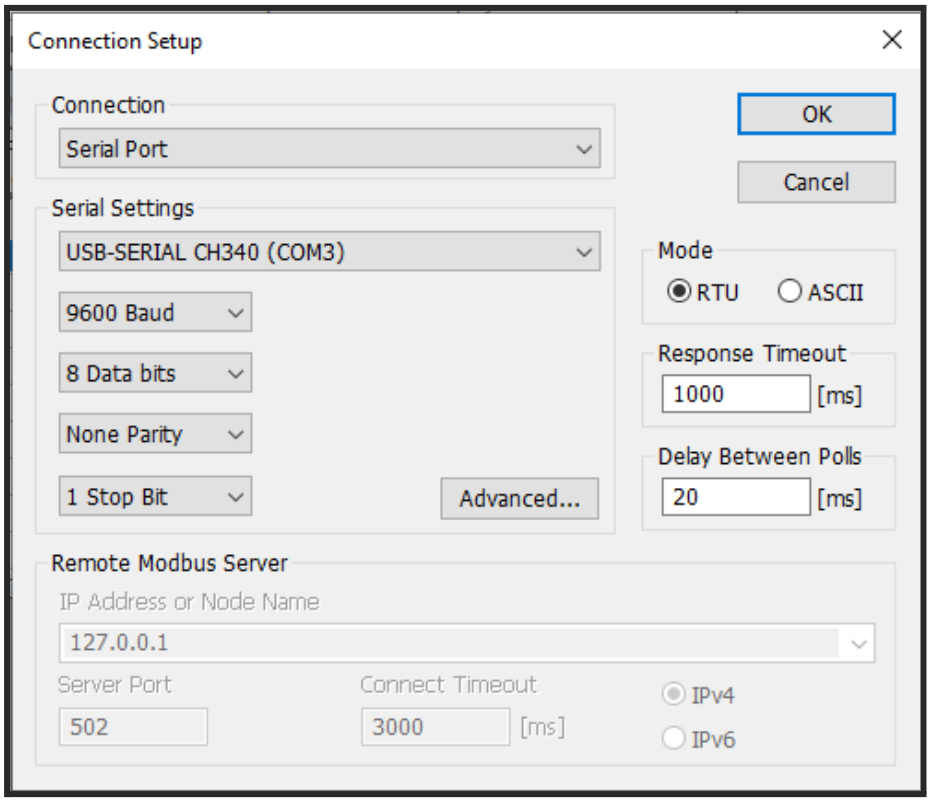
Con el siguiente comando leeremos los registros de entrada 2 y cantidad 2 del esclavo en la dirección 2 conectados a través de RTU /dev/ttyAMA0 (9600 Bd).
Los parámetros utilizados son:

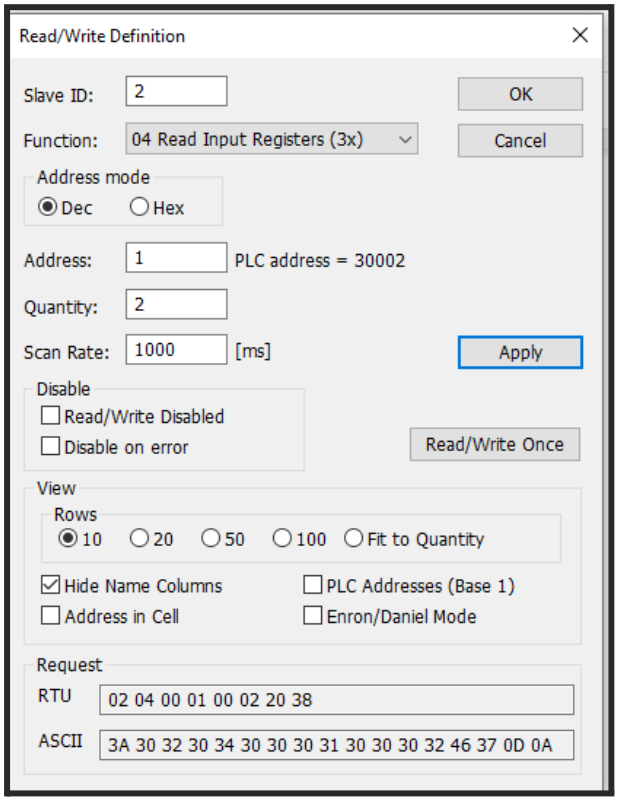
-m, modo transmisión rtu
-a, dirección esclavo
-b , velocidad
-P, paridad
-t, 3 tipo de datos de registro de entrada de 16 bits
-r, inicio
-c, valores a leer

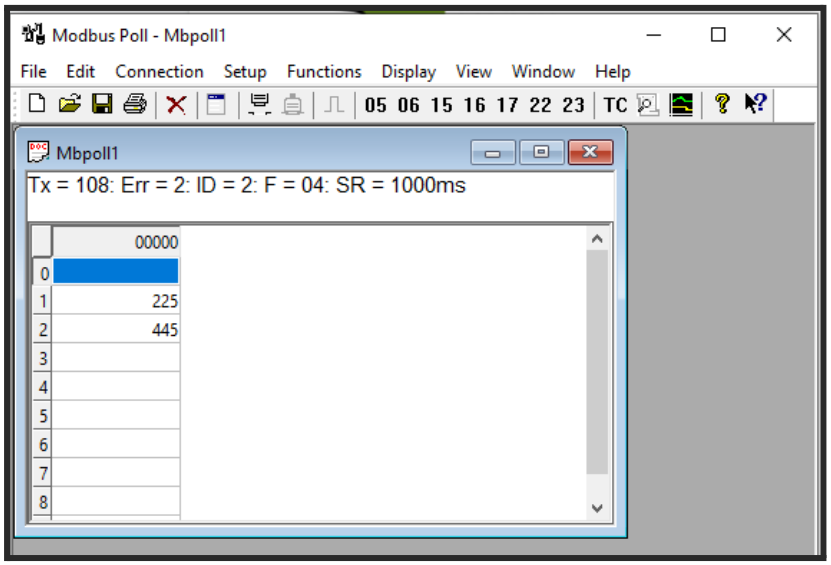
```
pi@raspberrypi:~ $ mbpoll -m rtu -a 2 -b 9600 -P none -t 3 -r 2 -c 2 -1 -v -q /dev/ttyAMA0
debug enabled
Set device=/dev/ttyAMA0
Opening /dev/ttyAMA0 at 9600 bauds (N, 8, 1)
Set response timeout to 1 sec, 0 us
-- Polling slave 2...
[02][04][00][01][00][02][20][38]
Waiting for a confirmation...
<02><04><04><00><E0><01><DD><08><BB>
[2]:    224
[3]:    477
```

Sabiendo el número de esclavo, con aplicaciones como Modbus Poll y Modbus Slave podemos simular un “ataque” que afectaría a la “**Confidencialidad**” lectura de datos del sensor, su ”**Integridad**” modificando los datos leídos (por ejemplo manipulando la corrección de temperatura/humedad) o su “**Disponibilidad**” impidiendo la conexión con el maestro, (modificando los parámetros RS485 o cambiando el ID del esclavo)

Utilizando Modbus Poll y Modbus Slave podemos modificar la lectura de los datos o el número de esclavo.

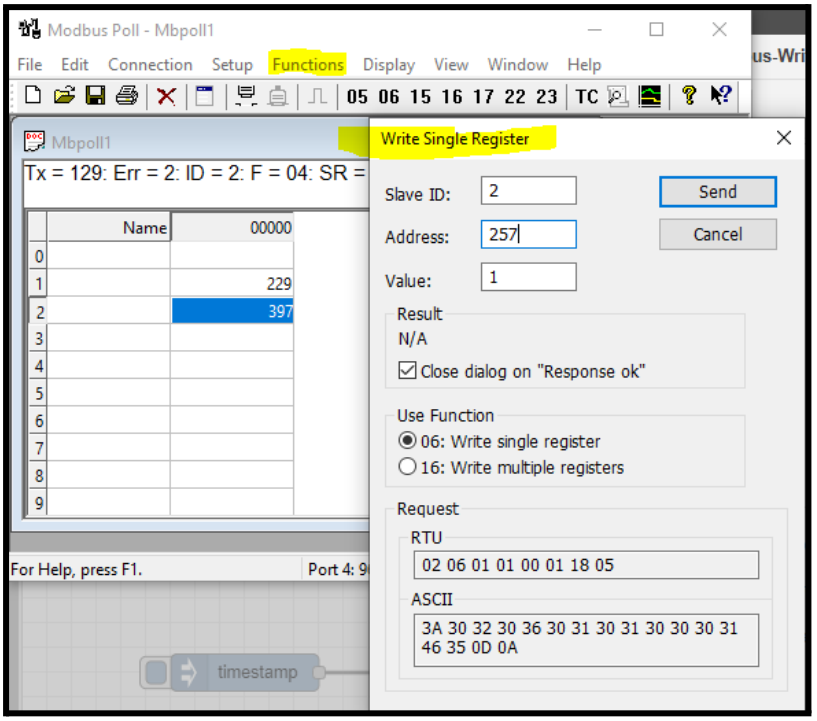

Connection Setup
Connection
Serial Port
OK
Cancel
Serial Settings
USB-SERIAL CH340 (COM3)
Mode
RTU
ASCII
9600 Baud
Response Timeout
1000
[ms]
8 Data bits
None Parity
Delay Between Polls
20
[ms]
1 Stop Bit
Advanced...
Remote Modbus Server
IP Address or Node Name
127.0.0.1
Server Port
502
Connect Timeout
3000
[ms]
IPv4
IPv6


Read/Write Definition
Slave ID:
2
OK
Function:
04 Read Input Registers (3x)
Cancel
Address mode
Dec
Hex
Address:
1
PLC address = 30002
Quantity:
2
Scan Rate:
1000
[ms]
Apply
Disable
Read/Write Disabled
Disable on error
Read/Write Once
View
Rows
10
20
50
100
Fit to Quantity
Hide Name Columns
PLC Addresses (Base 1)
Address in Cell
Enron/Daniel Mode
Request
RTU
02 04 00 01 00 02 20 38
ASCII
3A 30 32 30 34 30 30 30 31 30 30 30 32 46 37 0D 0A



Si pulsamos funciones y seleccionamos la 06, escribir en un registro, podemos modificar los datos como número de esclavo o los índices de corrección.

La siguiente imagen muestra como escribir en la dirección 257 que es la dirección del esclavo al cambiarla se perdería la disponibilidad de los datos del sensor.



Accedemos a la Raspberry por SSH y utilizando **mbpoll** en la línea de comando podemos comprobar si se produce la lectura de datos.

Práctica: Encendido/Apagado dispositivos controlado por sensor temperatura modbus

El objetivo de esta práctica es el diseño de una plataforma de automatización, la cual está constituida por un hardware de bajo coste. Controlaremos el encendido de un ventilador y una bombilla desde la raspberry con las mediciones que reciba de un sensor de temperatura con conexión modbus.

Utilizaremos un convertidor TTL a RS485 y un sensor de temperatura RS485 comunicación modbus.

Convertidor TTL a RS485

Este módulo conecta a la raspberry y permite convertir las señales seriales TTL a RS-485.

Características:

- Modelo: MAX-485
- Fuente de alimentación de amplio voltaje (3,0 V ~ 30V), voltaje límite 33V, se recomienda utilizar dentro de 30V, el voltaje mínimo debe ser superior a 2,8 V.
- Compatible con señales de 3,3 V y 5,0 V, ambos pueden coincidir con el módulo.
- Corriente de operación: 120uA~500uA
- Transferencia máxima de hasta 2.5mbps
- Potencia de operación: 830mW
- Tipo de Comunicación: Half-Duplex
- Velocidad máxima de 10 Mbit/s (a 12 metros)
- Longitud máxima de alcance de 1200 metros (a 100 kbit/s)

Podemos establecer comunicación simplex, half-duplex y full-duplex. Sin embargo, para la comunicación full-duplex tendremos que establecer dos canales distintos, y disponer de un receptor y emisor en cada uno de los terminales.

Sensor de temperatura RS485

Transmisor de temperatura y humedad RS485 Modbus-RTU

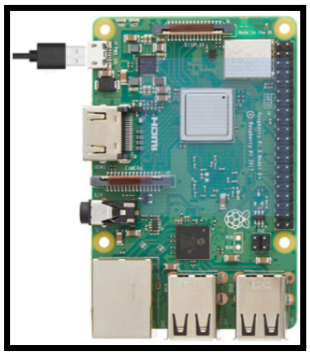
Relays 5V/In 220AC/Out

Características:

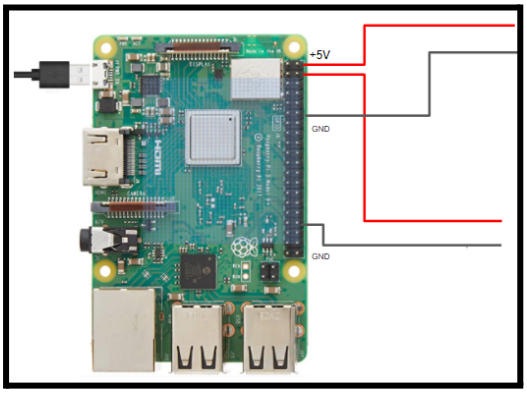
- Canales: 2 (independientes protegidos con Optoacopladores)
- Tensión de Alimentación: 5V
- Corriente de Salida: 10A
- Corriente de activación por relé: 15mA~20mA
- Aislamiento: Si
- LED indicador: Para cada canal

Conexiones

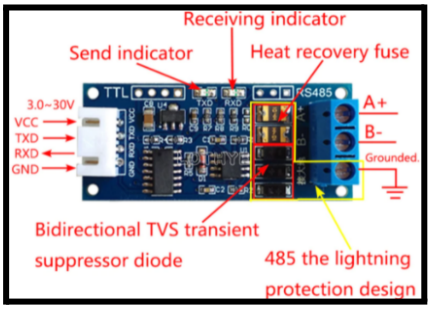
Al igual que en las prácticas anteriores empezaremos por alimentar a la Raspberry Pi 3 a través de cable usb de 5v.



Conectaremos a la raspberry.
Utilizamos los pines 2 y 4 que son alimentación a 5 V y los pines 14 y 34 que son toma de tierra.

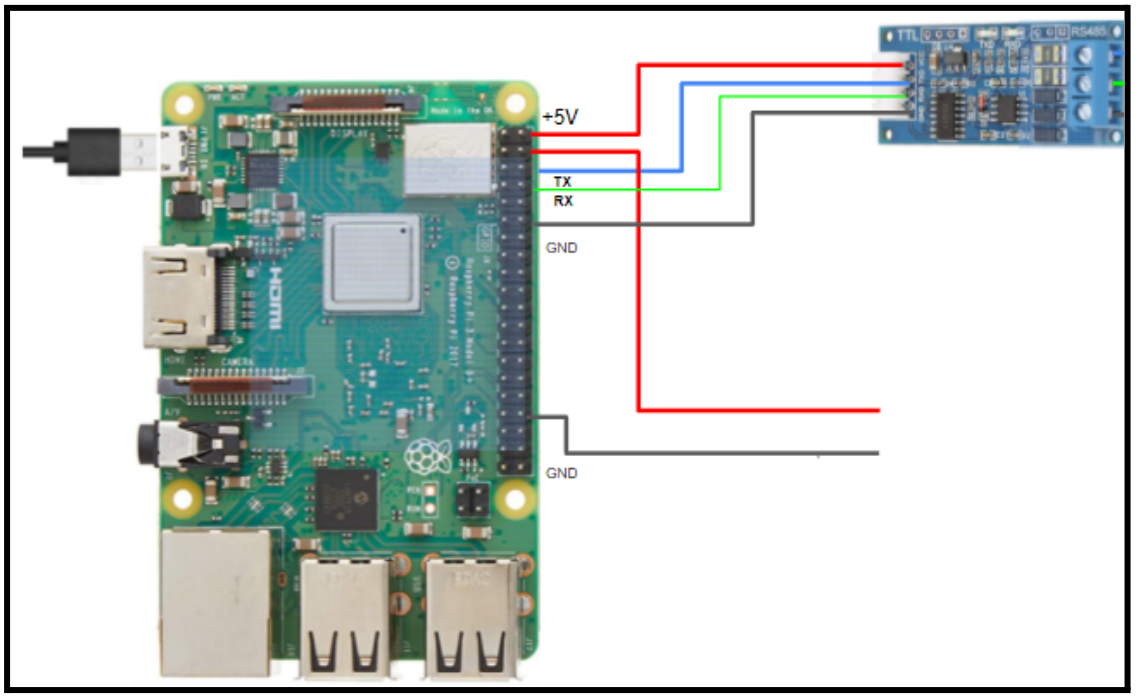


Conectamos el convertidor de señal TTL a RS485

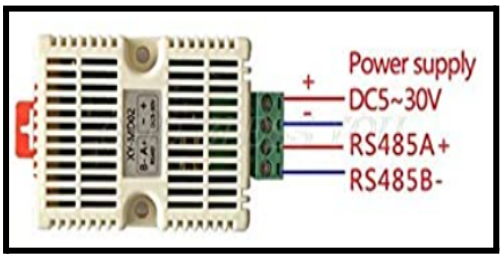


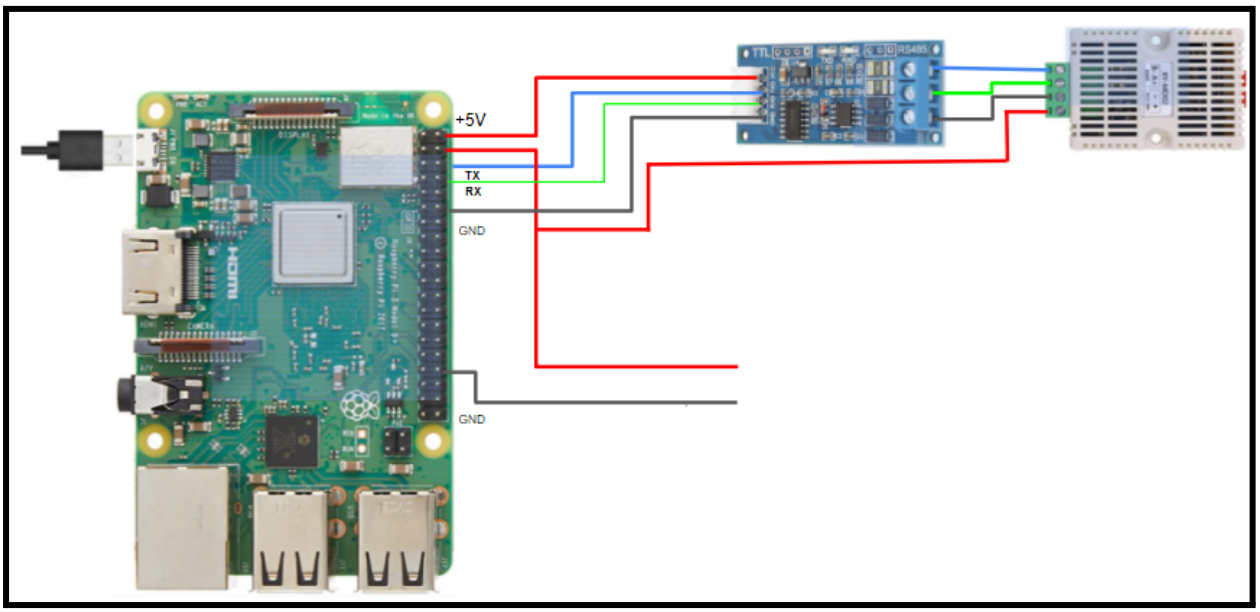
Alimentamos el módulo conectando Vcc a 5V y Gnd que me llega de la raspberry. Cables rojo y negro.

El pin 10 RX para la recepción de la raspberry se conecta al pin TXD, cable verde, y el pin 8 TX para transmisión se conecta al RXD del conversor, cable azul.



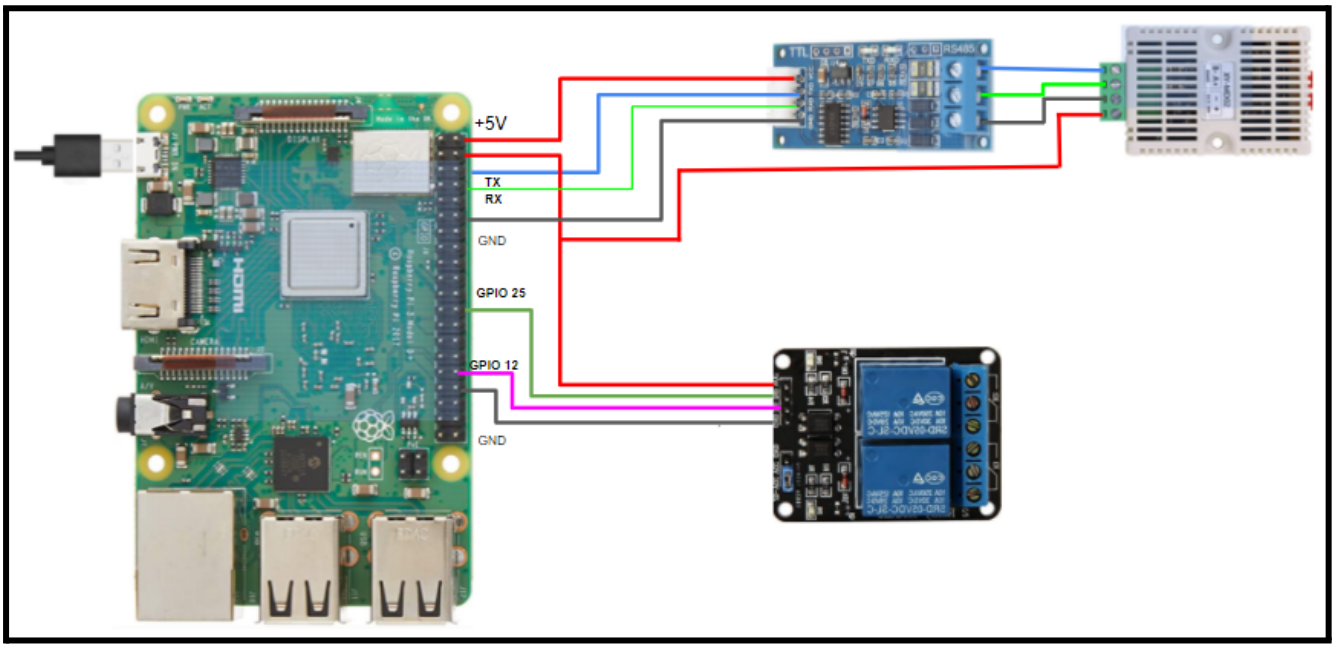
Por otro lado, tendremos que conectar los conductores A y B del par trenzado que constituyen el propio bus RS485, y al que estarán conectados todos los dispositivos que pertenezcan al mismo bus, en este ejemplo es el sensor de temperatura.





Ahora conectaremos el Relay a la Raspberry Pi 3
El relé tiene 4 pines que vamos a conectar a la raspberry:

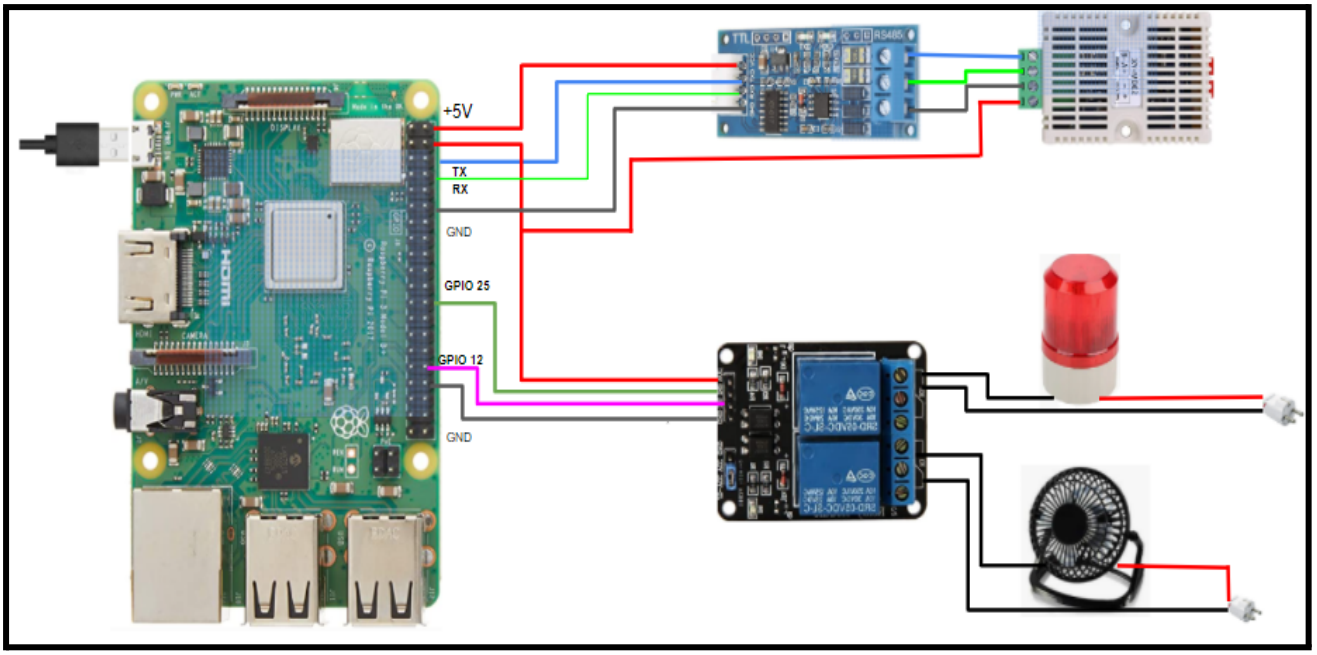
GND - Cable tierra al pin 14 de la raspberry
IN1 - Salida relé 1 al pin 32 (Gpio 12), cable morado
IN2 - Salida relè 2 al pin 22 (Gpio 25), cable verde
VCC - Conexión a 5 V al pin 1



El último paso será conectar la lámpara y el ventilador al relé.

Cada canal de relé tiene 3 conexiones. En el central va el cable neutro de alimentación y el cable que va al relé puede ir en el primero o tercero, según si deseas que el circuito esté abierto o cerrado cuando el relé está apagado o no está recibiendo energía (estado “por defecto”). Eso lo puedes diferenciar según las líneas que están unidas en el pin 2-3 y para el pin 1 no lo está.

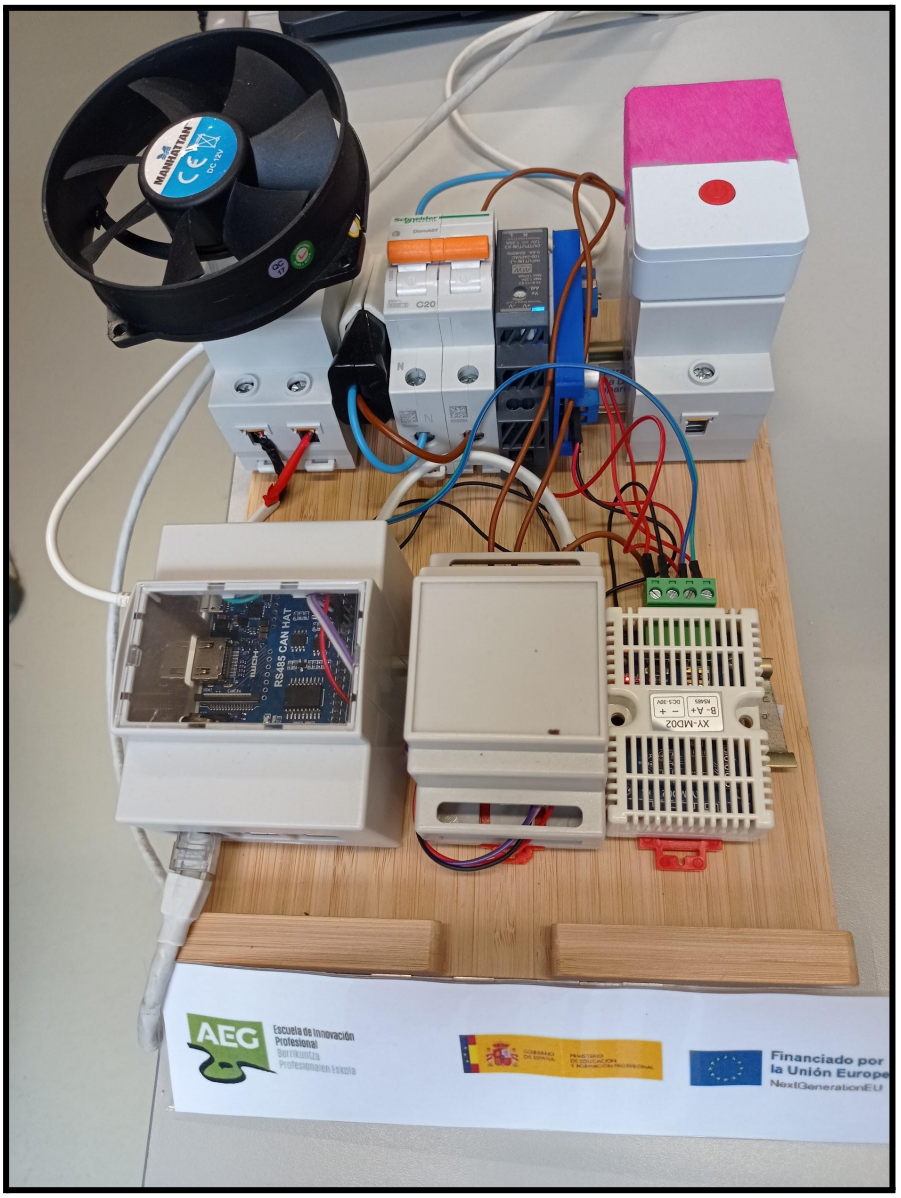
En este caso se queda conectado el 1-2, para que quede abierto el circuito por defecto.



Práctica: Prototipo comunicaciones industriales

El objetivo de esta práctica es el diseño de una plataforma de automatización en una entorno industrial utilizando hardware de bajo coste. Controlaremos el encendido de un ventilador y una bombilla con las mediciones que reciba de un sensor de temperatura.

El elemento principal elegido para esta plataforma de automatización de bajo coste es una Raspberry PI 3. Todas las comunicaciones entre dispositivos se realizan a través del protocolo de comunicaciones Modbus RTU.



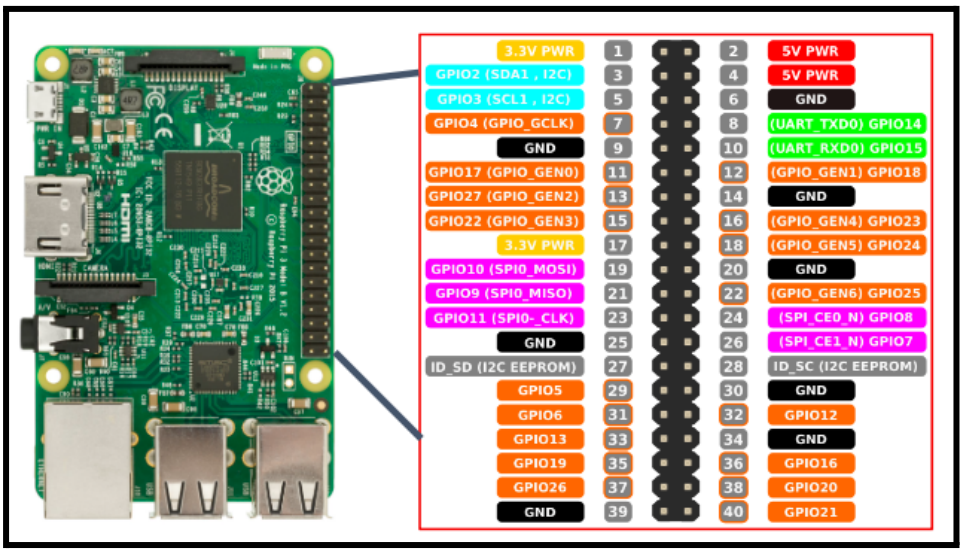
Componentes

A continuación, se describen las principales características de los elementos hardware utilizados para la elaboración de este trabajo.

Raspberry Pi3

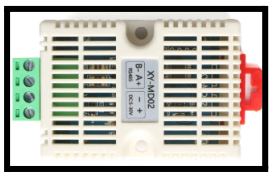
Es una placa computadora de bajo coste. En la práctica usaremos el modelo B que tiene las siguientes características:

- Chip integrado Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + Puerto USB)
- CPU 1.2 GHz 64-bit Quad-core ARMv8
- GPU Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 h.264/MPEG-4 AVC
- Memoria SDRAM 1 GB (compartidos con la GPU)
- CONECTIVIDAD INALÁMBRICA
  - 2.4GHz / 5GHz
  - IEEE 802.11.b/g/n/ac
  - Bluetooth 4.2, BLE
- CONECTIVIDAD DE RED - Gigabit Ethernet over USB 2.0 (300 Mbps de máximo teórico)
- PUERTOS
  - GPIO 40 pines
  - HDMI
  - 4 x USB 2.0
  - CSI (cámara Raspberry Pi)
  - DSI (pantalla táctil)
  - Toma auriculares / vídeo compuesto
  - Micro SD
  - Micro USB (alimentación 5V)
  - Power-over-Ethernet (PoE)
- Consumo energético 800 mA (4.0 W)

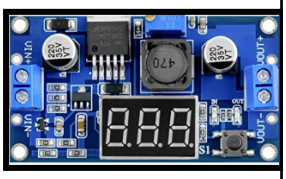


HAT de comunicación RS485 	Características: • Voltaje de funcionamiento: 3,3 V. • Controlador CAN: MCP2515 • Transmisor CAN: SN65HVD230 • Transmisor 485: SP3485

Este HAT habilita a cualquier Raspberry Pi para soportar comunicaciones a larga distancia con otros dispositivos de forma estable, vía protocolo RS485 y funciones CAN.

Sensor de temperatura RS485 	Características: • Funcionamiento: CC 5 V-30 V. • Señal de salida: señal RS485. • Dirección de comunicación: 1 ~ 247 (predeterminado 1). • Rango de temperatura: -40?~60? • Precisión de temperatura: +/-0.5? • Resolución de temperatura: 0,1

Relays 5V/In 220AC/Out 	Características: • Canales: 2 (independientes protegidos con Optoacopladores) • Tensión de Alimentación: 5V • Corriente de Salida: 10A • Corriente de activación por relé: 15mA~20mA • Aislamiento: Si • LED indicador: Para cada canal

Módulo convertidor de voltaje de 12v a 5v	Lámpara AC 220v	Ventilador DC 12v	Interruptor magnetotérmico
			

Enchufe carril DIM	Convertidor 220v a 12v	Conectores de mini cables	Carril DIN
			

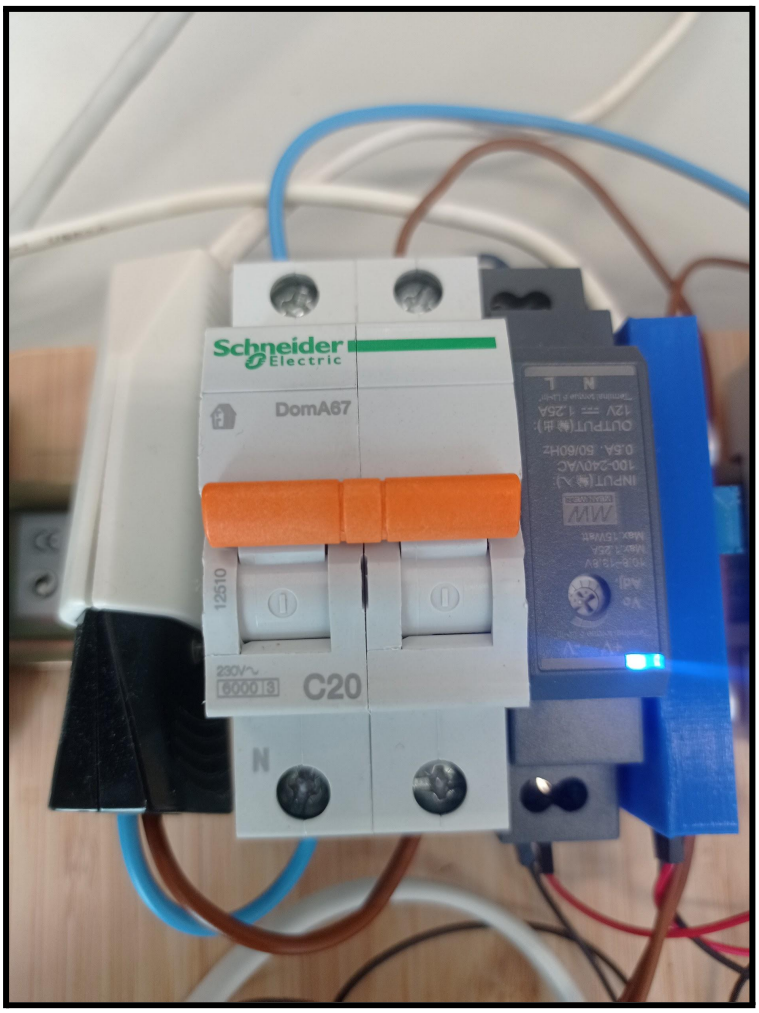
Conexiones

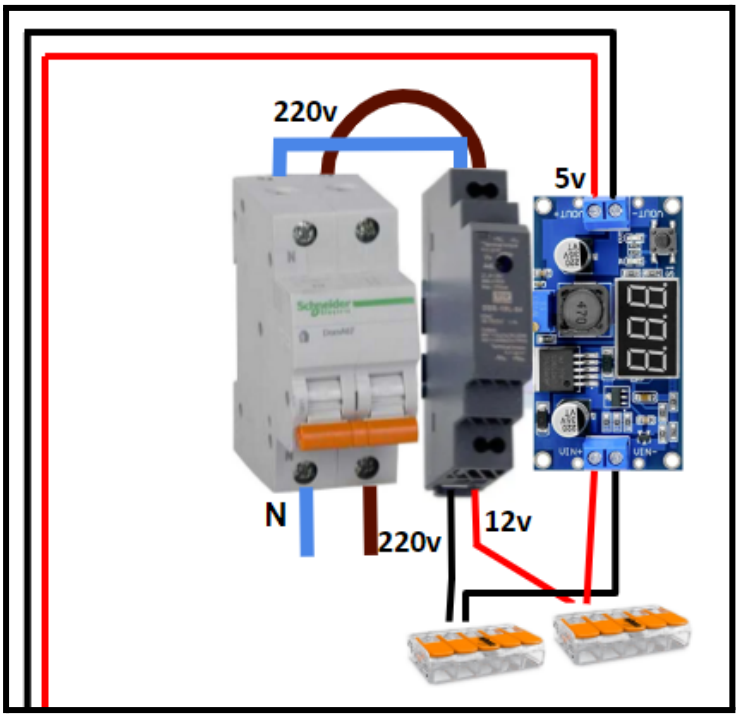
Realizaremos primero toda la conexión y conversión de voltajes.

Montamos el interruptor en el carril DIM.
Cableamos el interruptor magnetotérmico cableando la entrada y salida, entrando y saliendo 220v.

Unimos al carril DIM el convertidor de 220v a 12v y lo cableamos desde el interruptor. Así tendremos una salida de 12v para alimentar a diferentes componentes.

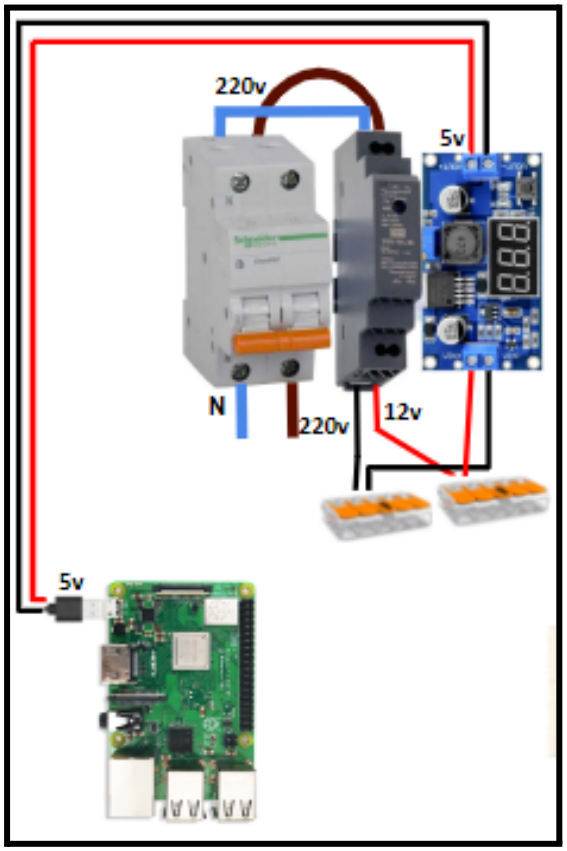
Esta salida de 12v la llevamos a un conector múltiple para alimentar a diferentes componentes y al convertidor de 12v a 5v que necesitamos para alimentar a la Raspberry.


Schneider Electric
DomA67
C20
N

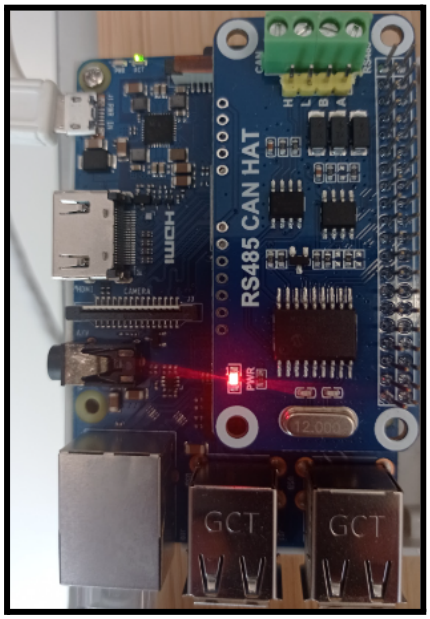


Ahora se dispone de alimentación a diferentes tensiones 220v, 12v, 5v.

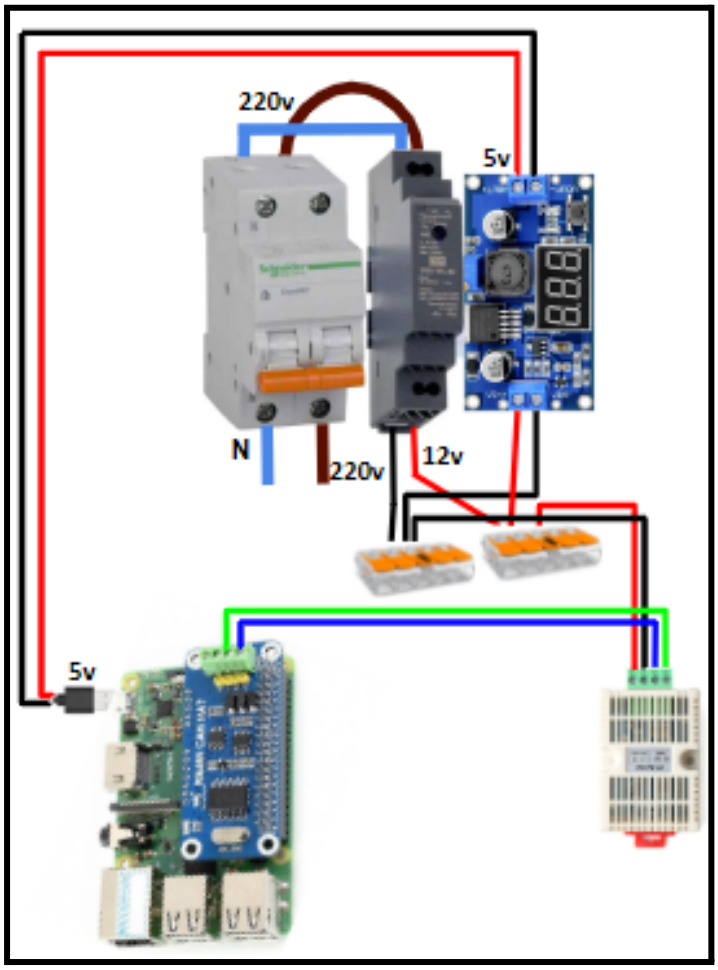
Empezaremos por alimentar a la Raspberry Pi 3 a través de cable usb de 5v.



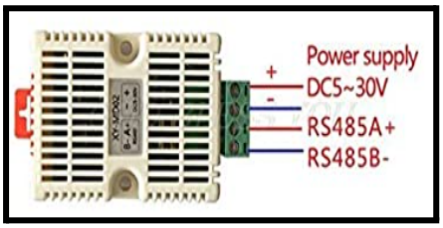
Unimos el HAT de comunicación RS485 a la Raspberry uniendolos por los pins.



Conectaremos el sensor de temperatura al módulo RS485 CAN HAT.

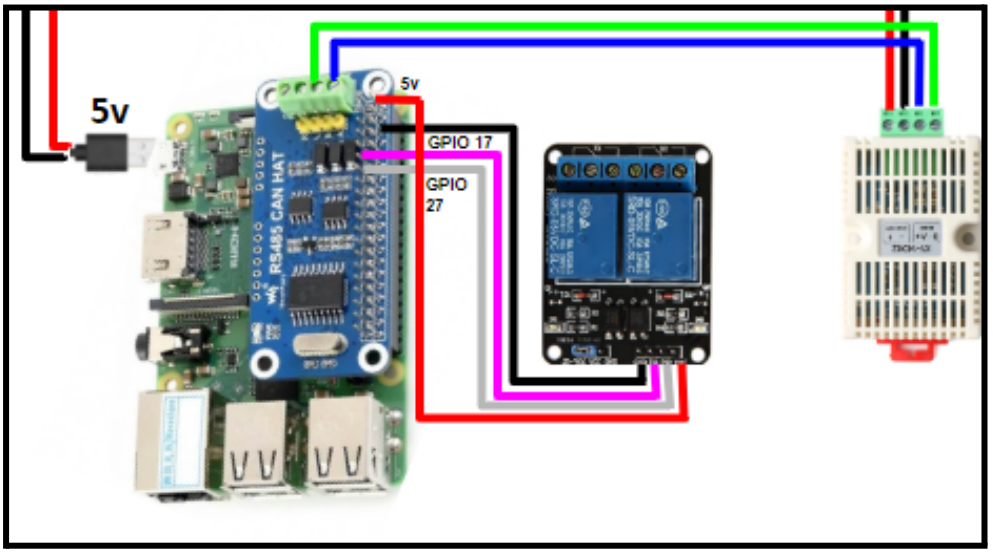


Tenemos que conectar los conductores A y B del par trenzado que constituyen el propio bus RS485 al módulo RS485 CAN HAT y estará alimentado con tensión de 12v que proviene del conector de cables.



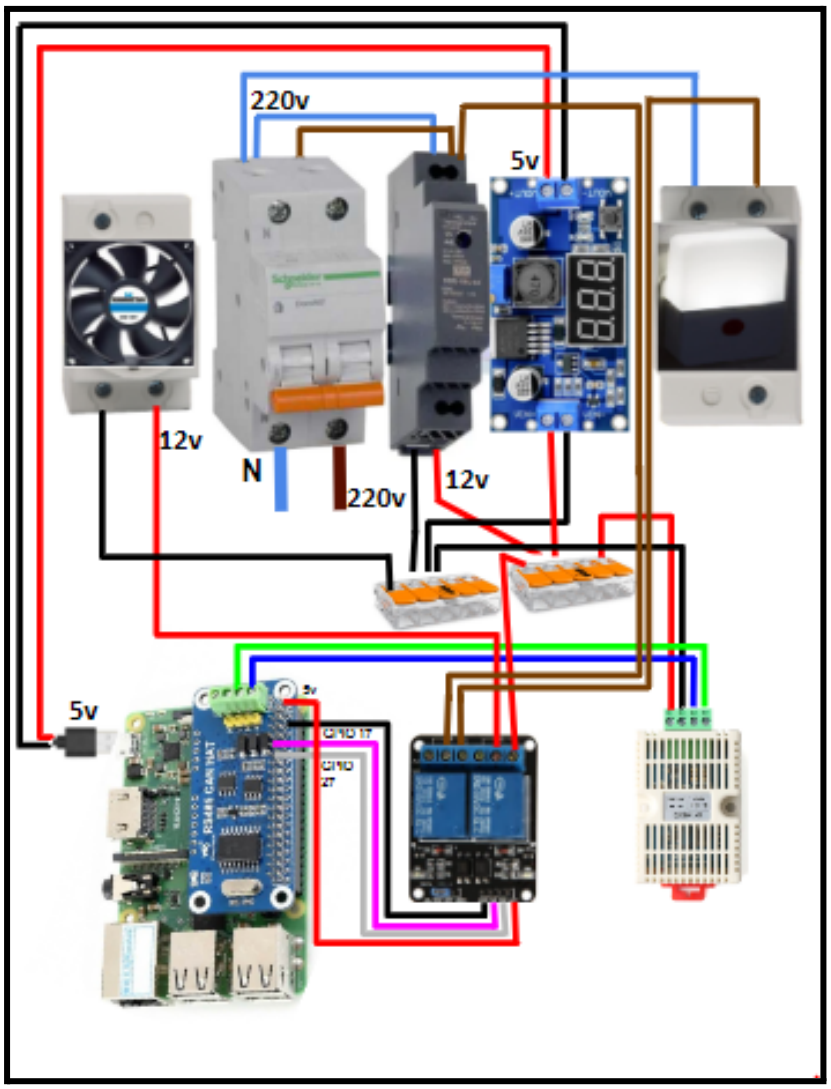
Ahora conectaremos el Relay a la Raspberry Pi 3. El relé tiene 4 pines que vamos a conectar a la raspberry:

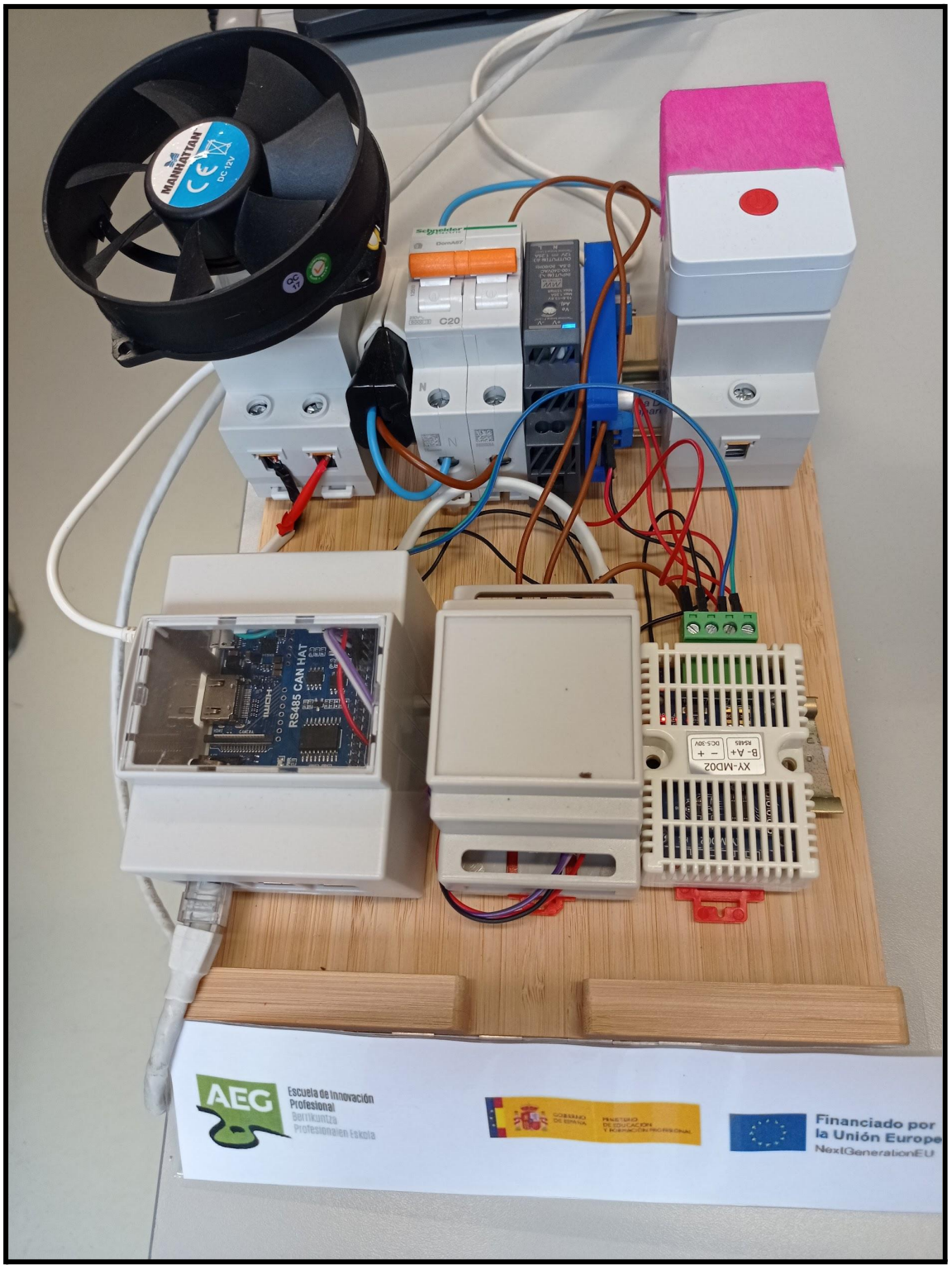
GND - Cable tierra al pin 6 de la raspberry
IN1 - Salida relé 1 al pin 11(Gpio 17), cable morado
IN2 - Salida relè 2 al pin 13 (Gpio 27), cable blanco
VCC - Conexión a 5 V al pin 2



El último paso será conectar la lámpara y el ventilador al relé.
Cada canal de relé tiene 3 conexiones. En el central va el cable neutro de alimentación y el cable que va al relé puede ir en el primero o tercero, según si deseas que el circuito esté abierto o cerrado cuando el relé está apagado o no está recibiendo energía (estado “por defecto”). Eso lo puedes diferenciar según las líneas que están unidas en el pin 2-3 y para el pin 1 no lo está.

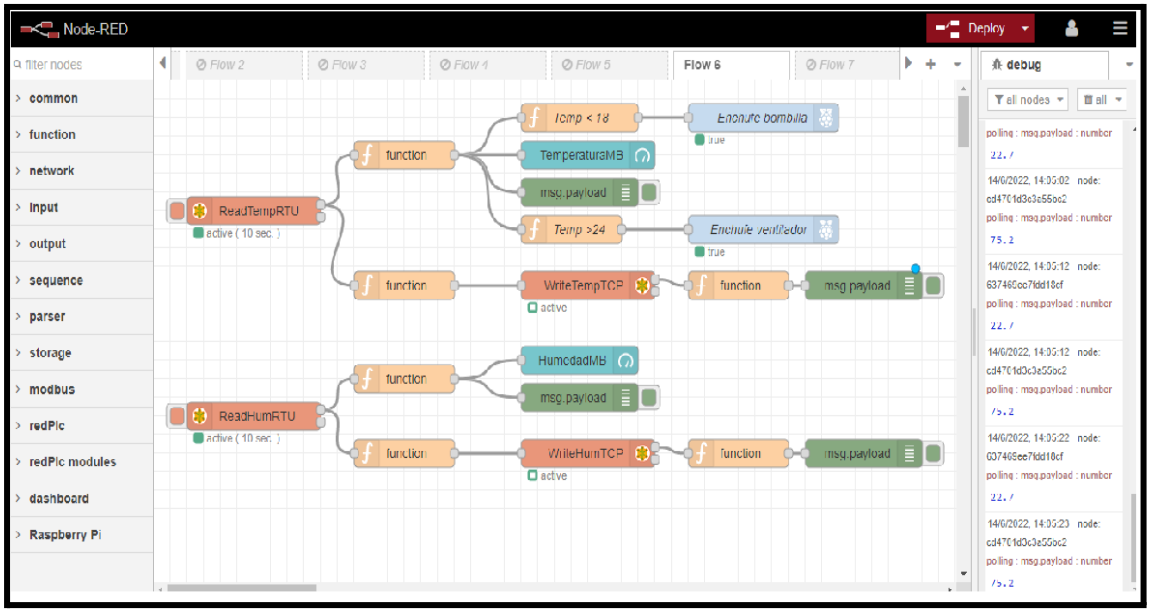
En este caso se queda conectado el 1-2, para que quede abierto el circuito por defecto.

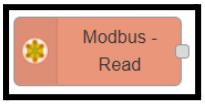

220v
5v
12v
N
220v
12v
5v


MANHATTAN
DC 12V
C20
RS485 CAN HAT
XY-MD02
B- A+
RS485
DC5-30V
AEG
Escuela de Innovación Profesional
Berrikuntza Profesionalen Eskola
Financiado por la Unión Europe
NextGenerationEU

Programación en Node-Red

Realizaremos la programación de Raspberry con la aplicación Node-Red.



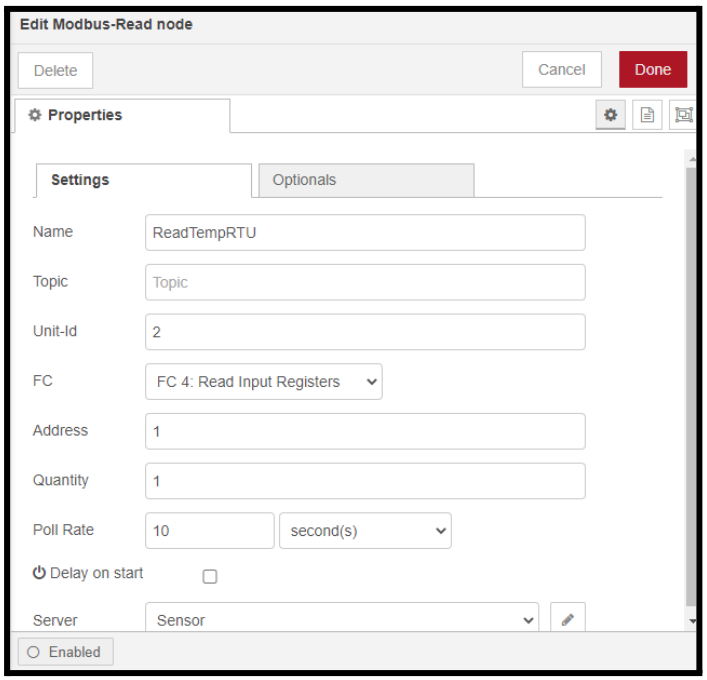
Insertamos dos nodos **Modbus - Read** 

para poder leer los registros del sensor de temperatura XY-MD02 que utiliza conexión RS485.

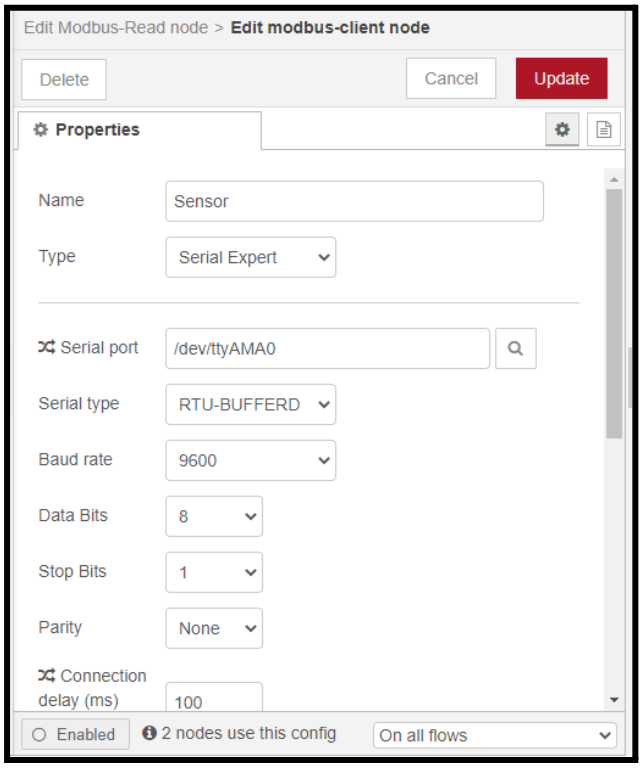
El nodo Modbus - Read soporta 4 funciones:

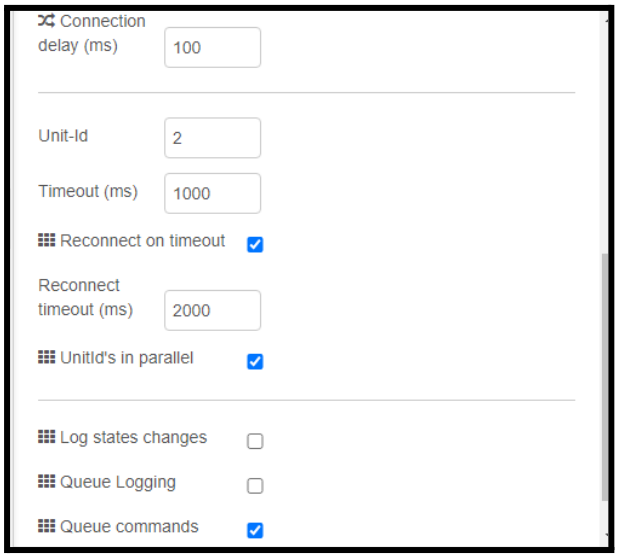
- FC 1: Read Coil Status (Leer estado sensor)
- FC 2: Read Input Status (Leer estado entrada)
- FC 3: Read Holding Registers (Leer registros de sensor)
- FC 4: Read Input Registers (Leer registros de entrada)

Unit-Id (0..255 tcp | 0..247 serial) - déjelo vacío, de lo contrario anula el Unit-ID predeterminado de la configuración del cliente.
FC: Elija el código de función (FC 4) del menú desplegable.
Address: Seleccione la dirección de inicio de bobina/entrada/registro (0:65535)
Quantity: La cantidad de bobinas/entradas/registros que se leerán desde la dirección de inicio.
Poll Rate: Configure una tasa de sondeo (mayor que cero) y la unidad de tiempo.



Server. Elija o edite la configuración de la conexión Modbus.





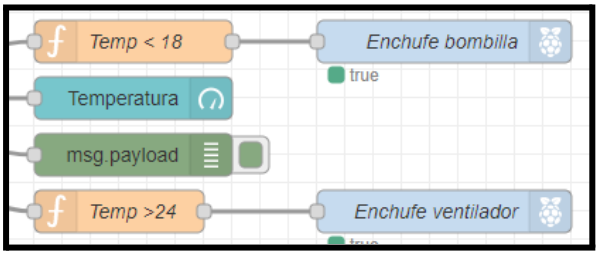
Serial Port: /dev/ttyAMA0, es para Bluetooth pero, recordamos, que hemos modificado la configuración de la Raspberry para poder utilizarlo como puerto serie.
Type: seleccionamos Serial Expert o Serial.
Baud Rate: 9600, este dato nos viene en la documentación del sensor.
Unit-Id: predeterminado 1 [serial] o 0 [tcp], para establecer una ID de unidad para todos los nodos sin ID de unidad. **Es el número de esclavo,** aquí el valor es 2 porque previamente ha sido modificado el valor por defecto de 1.
El resto de valores los dejamos por defecto.

Realizamos la configuración del nodo para los valores de la humedad.

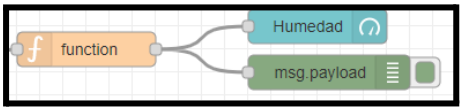
En la opción debug podemos ver los datos del sensor.

Insertamos un nodo función que activaran la bombilla cuando la temperatura esté por debajo de 18 y otro nodo función que encenderá el ventilador si la temperatura supera los 24.

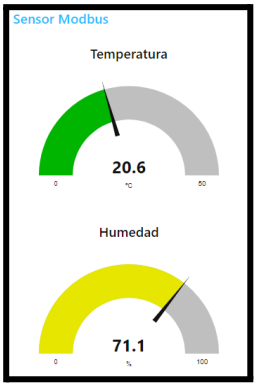
También añadimos un nodo gauge para ver representada la temperatura en una dashboard.



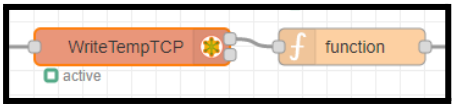
De la misma forma añadimos un nodo gauge para ver los datos de la humedad en dashboard.



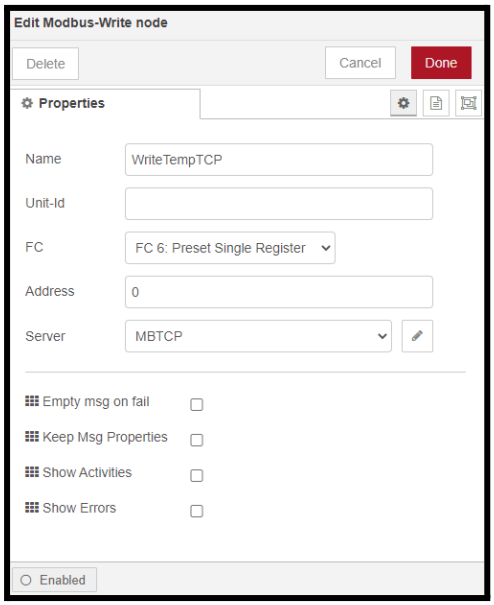
Al acceder a la dashboard vemos los datos de temperatura y humedad del sensor representados.

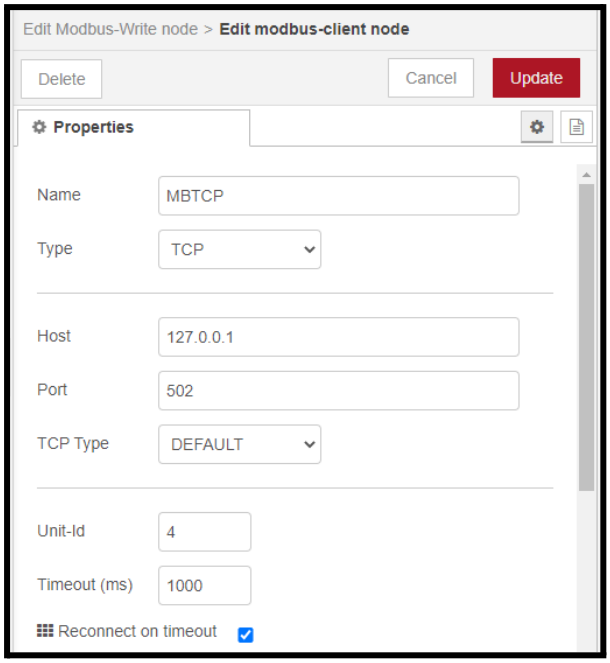


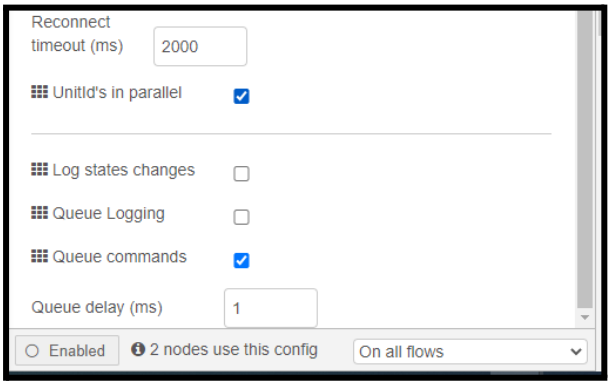
También utilizaremos un nodo Modbus-Write para escribir el dato de la temperatura en la raspberry que funcionará como servidor Modbus TCP en el puerto 502.



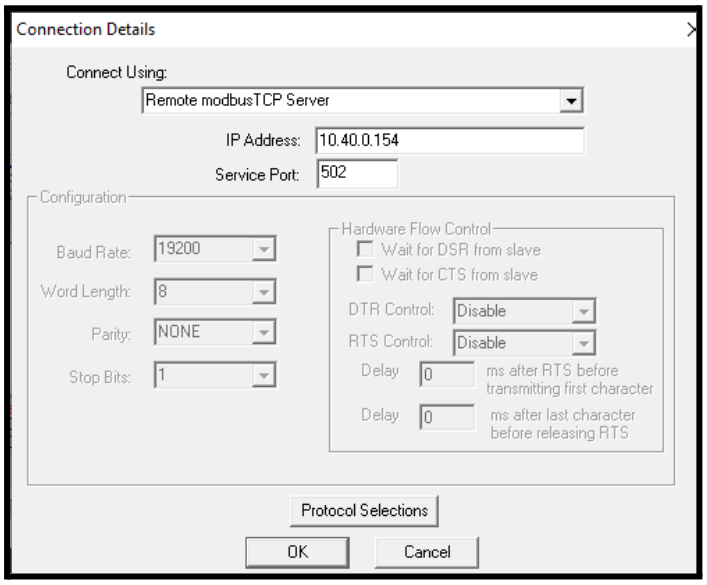
Aplicaremos el código de función 6 para grabar un registro único y definiremos el servidor, la dirección de inicio del registro y la cantidad a escribir

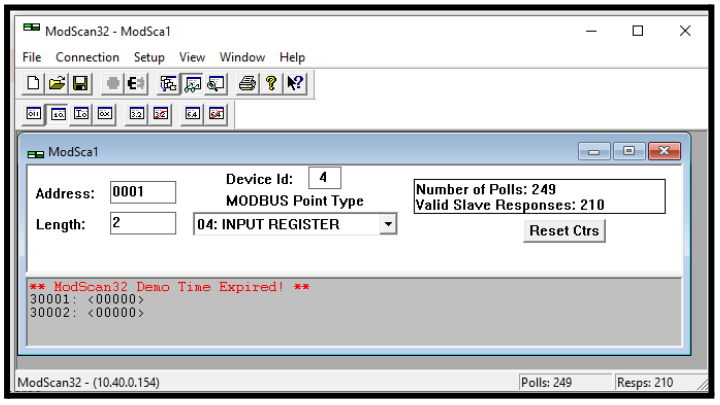

Edit Modbus-Write node
Delete
Cancel
Done
Properties
Name
WriteTempTCP
Unit-Id
FC
FC 6: Preset Single Register
Address
0
Server
MBTCP
Empty msg on fail
Keep Msg Properties
Show Activities
Show Errors
Enabled


Edit Modbus-Write node > Edit modbus-client node
Delete
Cancel
Update
Properties
Name
MBTCP
Type
TCP
Host
127.0.0.1
Port
502
TCP Type
DEFAULT
Unit-Id
4
Timeout (ms)
1000
Reconnect on timeout


Reconnect timeout (ms)
2000
UnitId's in parallel
Log states changes
Queue Logging
Queue commands
Queue delay (ms)
1
Enabled
2 nodes use this config
On all flows

Para visualizar los datos grabados podemos utilizar el programa ModScan y conectarnos por TCP al servidor modbus, nuestra raspberry en el puerto 502.





Guía didáctica creada en el marco del Proyecto Industria 4.0

INDUSTRIA 4.0

Socios del proyecto:

Centro de estudios AEG ARROKA S.L

Muévete Gestión Integral S.L.

CES S. Ramón y Cajal

Titanium Industrial Security

HORINTEG Soluciones Tecnológicas, S.L.

“Financiado por el Ministerio de Educación y Formación Profesional – U.E. – Next Generation”