

4 - Posibilidades avanzadas

El objetivo de este capítulo es compartir con vosotros/as alguna cuestión de mayor dificultad a las ya planteadas en capítulos anteriores. Concretamente vamos a ver como montar un alojamiento en la nube similar a lo que vendría siendo Google Drive. También vamos a usar nuestra Raspberry Pi para gestionar la domótica de nuestra vivienda y, para terminar, haremos uso de los distintos pines de nuestra Raspberry Pi

- [4.1 NextCloud. Tu nube personal](#)
- [4.2 Haciendo uso del pinout](#)
- [4.3 Domótica](#)

4.1 NextCloud. Tu nube personal



Imagen obtenida de https://es.m.wikipedia.org/wiki/Archivo:Nextcloud_Logo.svg

Esta herramienta sirve para...

Una vez mas vamos a recurrir a la wikipedia:

Nextcloud es una serie de programas cliente-servidor que permiten la creación de servicios de alojamiento de archivos. Su funcionalidad es similar al software Dropbox, aunque Nextcloud es en su totalidad software de código abierto. Nextcloud permite a los usuarios crear servidores privados. Su modelo de desarrollo abierto permite añadir y/o modificar la funcionalidad del software del servidor en forma de aplicaciones.

<https://es.wikipedia.org/wiki/Nextcloud>

Y según lo anterior me podréis decir que en el apartado [3.9 File Browser. Explorador de ficheros en remoto](#) ya nos hemos dotado de una muy buena herramienta web que nos permite gestionar nuestros ficheros en remoto. En esta ocasión vamos a ver una herramienta que mejora las posibilidades (y aumenta la complejidad) de la anterior ya que cuenta entre otras con las siguientes características:

- Los archivos Nextcloud son almacenados en estructuras de directorio convencionales y se pueden acceder a través del protocolo WebDAV si es necesario.
- Los archivos son encriptados en la transmisión y opcionalmente durante el almacenamiento.
- Los usuarios pueden manejar calendarios (CalDAV), contactos (CardDAV), tareas programadas y reproducir contenido multimedia (Ampache).
- Permite la administración de usuarios y grupos de usuarios (vía OpenID o LDAP) y definir permisos de acceso.
- Posibilidad de añadir aplicaciones (de un solo clic) y conexiones con Dropbox, Google Drive y Amazon S3.
- Disponibilidad de acceso a diferentes bases de datos mediante SQLite, MariaDB, MySQL, Oracle Database, y PostgreSQL.
- Disponibilidad de un software llamado Nextcloud box basado en Raspberry Pi que funciona en Ubuntu Core.
- Posibilidad de integrar los editores en línea ONLYOFFICE mediante la aplicación oficial.

<https://es.wikipedia.org/wiki/Nextcloud>

que permiten dar un paso adelante con respecto a la solución que con anterioridad habíamos visto.

Web de proyecto y otros enlaces de interés

Página web oficial del proyecto: <https://nextcloud.com/>

Repositorio de código: <https://github.com/nextcloud/server>

Instrucciones de instalación en un servidor propio: <https://nextcloud.com/install/#instructions-server>

Despliegue

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir nextcloud
cd nextcloud
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido (adaptado de <https://github.com/nextcloud/all-in-one/blob/main/docker-compose.yml>):

```
version: "3.7"

volumes:
  nextcloud_aio_mastercontainer:
    name: nextcloud_aio_mastercontainer # This line is not allowed to be changed

services:
  nextcloud:
    image: nextcloud/all-in-one:latest
    restart: always
    container_name: nextcloud-aio-mastercontainer # This line is not allowed to be changed
    volumes:
      - nextcloud_aio_mastercontainer:/mnt/docker-aio-config # This line is not allowed to be changed
      - /var/run/docker.sock:/var/run/docker.sock:ro # May be changed on macOS, Windows or docker rootless.
    See the applicable documentation. If adjusting, don't forget to also set 'DOCKER_SOCKET_PATH'!
    ports:
      - 80:80 # Can be removed when running behind a web server or reverse proxy (like Apache, Nginx and else).
    See https://github.com/nextcloud/all-in-one/blob/main/reverse-proxy.md
      - 8080:8080
      - 8443:8443 # Can be removed when running behind a web server or reverse proxy (like Apache, Nginx and
    else). See https://github.com/nextcloud/all-in-one/blob/main/reverse-proxy.md
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos.

Si algún otro servicio está utilizando los puertos que en este servicio vamos a utilizar se generará un conflicto y puede que ninguno de los servicios funcione o, mas probable, el último que pongamos en marcha.

Para usar un puerto diferente puedes cambiar el valor que aparece ANTES de los : por un valor que no esté en uso. También puedes acceder al directorio dónde se encuentra el otro

servicio y ejecutar `docker-compose down`.

Posteriormente ponemos en marcha los contenedores con `docker-compose up -d` (le costará un buen rato).

```
pi@mediacenter:~$ cd nextcloud/
pi@mediacenter:~/nextcloud$ nano docker-compose.yml
pi@mediacenter:~/nextcloud$ docker-compose up -d
Creating network "nextcloud_default" with the default driver
Creating volume "nextcloud_aino_mastercontainer" with default driver
Pulling nextcloud (nextcloud/all-in-one:latest)...
latest: Pulling from nextcloud/all-in-one
af0ea76a39c: Already exists
65da1d1bbae1: Pull complete
53b894e9854e: Pull complete
52c363c99e0a: Pull complete
5e2784d359b4: Pull complete
2fe294a1d431: Pull complete
a1b5d448da44: Pull complete
d6fa85b4242a: Pull complete
7e8cead75c08: Pull complete
0991d6d48ba0: Pull complete
80fe1ffc4db7: Pull complete
ecd333471262: Pull complete
4f4fb700ef54: Pull complete
857a401066d5: Pull complete
cd5d0c30d9d9: Pull complete
1b23aed07d3a: Pull complete
45907596a6ce: Pull complete
154af1b1b7b2: Pull complete
085dfb9f819e: Pull complete
55d9fb1d7c1f: Pull complete
38d61e23d3ae: Pull complete
f8e1308d997f: Pull complete
b23aed78b40: Pull complete
21a2dceec33a: Pull complete
1d863f3d5896: Pull complete
bfae6fa003a0: Pull complete
eca684d1937a: Pull complete
549a505a53b7: Pull complete
0115fde3233: Pull complete
89a32a191695: Pull complete
35f9b4e0ef10: Pull complete
63992b3a3136: Pull complete
Digest: sha256:0e7e474012a4992cfe86b41ff9c81dc743bedb732291c55f4490891255bca181
Status: Downloaded newer image for nextcloud/all-in-one:latest
Creating nextcloud-aino-mastercontainer ...
Creating nextcloud-aino-mastercontainer ... error
ERROR: for nextcloud-aino-mastercontainer: Cannot start service nextcloud: driver failed programming external connectivity on endpoint nextcloud-aino-mastercontainer (8a97ae270906083253e607ed74361ebc405d52a1ce2aa62aa46b0c2a16964b2a): Bind for 0.0.0.0:8080 failed: port is already allocated
ERROR: for nextcloud: Cannot start service nextcloud: driver failed programming external connectivity on endpoint nextcloud-aino-mastercontainer (8a97ae270906083253e607ed74361ebc405d52a1ce2aa62aa46b0c2a16964b2a): Bind for 0.0.0.0:8080 failed: port is already allocated
ERROR: Encountered errors while bringing up the project.
```

Elaboración propia

En esta ocasión el despliegue nos ha fallado por lo que comentaba en la advertencia anterior. El mensaje de error nos indica que el puerto 8080 ya está en uso. Con `docker ps` voy a ver qué contenedores están utilizando qué puertos:

```
pi@mediacenter:~/nextcloud$ docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
d455db3ea0be	pihole/pihole:latest	pihole	"/sbin-init"	49 minutes ago	Up 9 minutes (healthy)	0.0.0.0:53->53/udp, 0.0.0.0:53->53/tcp, 0.0.0.0:67->67/udp, 0.0.0.0:8088->80/tcp
97289b09964a	hurlenko/filebrowser	filebrowser	"/filebrowser --root..."	13 days ago	Up 9 minutes	0.0.0.0:8080->8080/tcp
b47f1be3aadd	linuxserver/deluge	deluge	"/init"	13 days ago	Up 9 minutes	0.0.0.0:8112->8112/tcp, 0.0.0.0:58846-58850->58846-58850/tcp, 0.0.0.0:58946-58950->58946-58950/tcp
65b94d7bc33b	linuxserver/jackett	jackett	"/init"	13 days ago	Up 9 minutes	0.0.0.0:9117->9117/tcp
b8c841631d14	linuxserver/sonarr	sonarr	"/init"	13 days ago	Up 9 minutes	0.0.0.0:8989->8989/tcp
a72d97516678	linuxserver/bazarr	bazarr	"/init"	13 days ago	Up 9 minutes	0.0.0.0:6767->6767/tcp
fccdd989aalc	portainer/portainer-ce	portainer	"/portainer"	13 days ago	Up 9 minutes	8000/tcp, 9443/tcp, 0.0.0.0:9000->9000/tcp
89c42586e7be	linuxserver/radarr	radarr	"/init"	13 days ago	Up 9 minutes	0.0.0.0:7878->7878/tcp
f23ea8117a30	lscr.io/linuxserver/duckdns:latest	duckdns	"/init"	13 days ago	Up 9 minutes	
415f691fe1a7	crazymax/diun:latest	diun	"diun serve"	2 weeks ago	Up 9 minutes	
32baad3eabc2	lscr.io/linuxserver/wireguard:latest	wireguard	"/init"	6 weeks ago	Up 9 minutes	0.0.0.0:51820->51820/udp

Elaboración propia

Aquí veo que el servicio pihole es quién está usando actualmente ese puerto. Posibles soluciones:

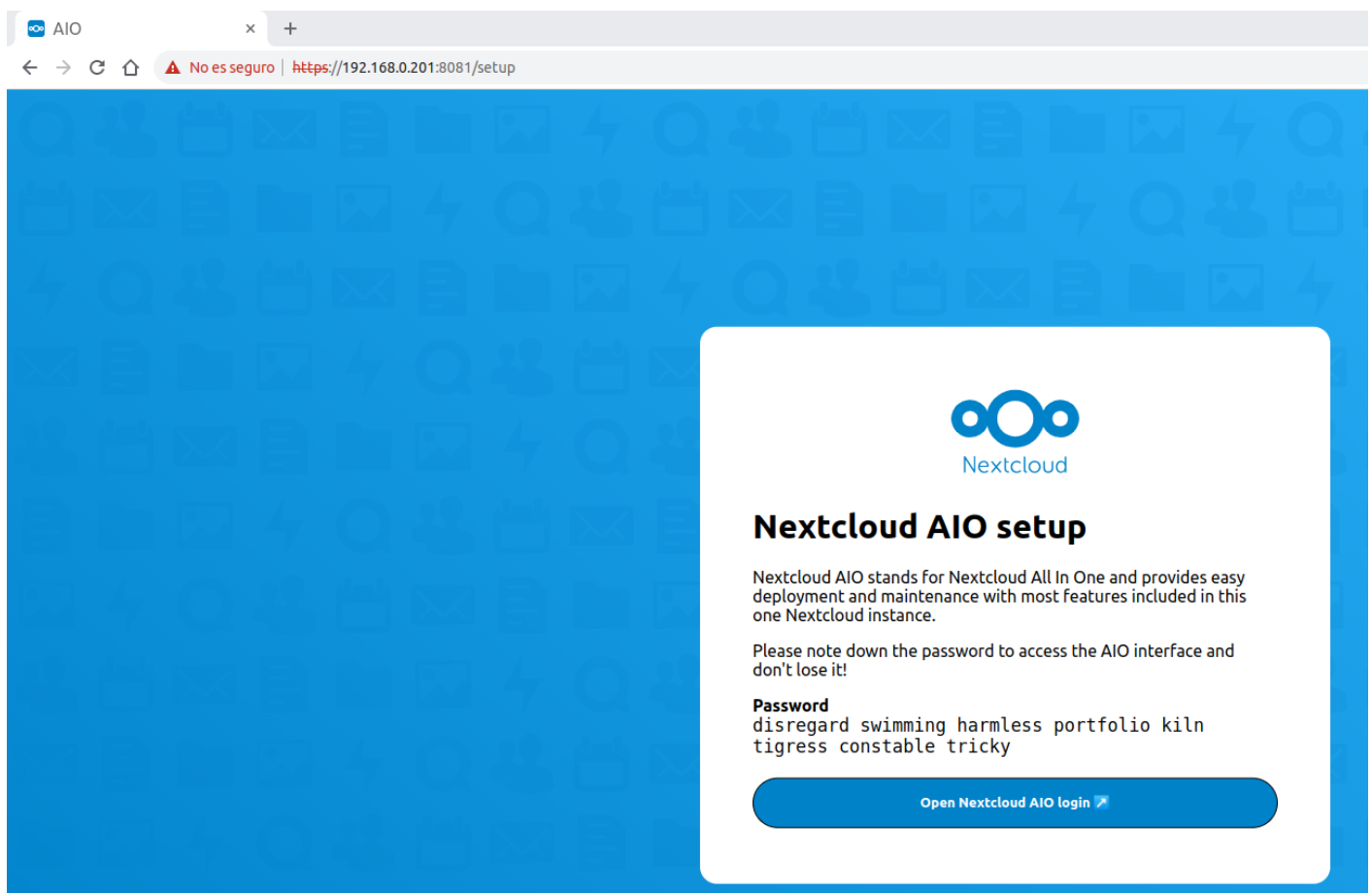
- Detener pi-hole: accedo al directorio dónde está pi-hole y ejecuto `docker-compose down`. Esto hará que deje de funcionar.
- Cambiar en el fichero docker-compose.yml de nextcloud la línea que dice 8080:8080 por, por ejemplo, 8081:8080. Nosotros optaremos por esta segunda opción

tomemos la decisión que tomemos deberemos volver a acceder al directorio de nextcloud y ejecutar `docker-compose up -d`. Si ahora va bien veremos algo como:

```
pi@mediacenter:~/nextcloud $ nano docker-compose.yml
pi@mediacenter:~/nextcloud $ docker-compose up -d
Recreating nextcloud-aio-mastercontainer ... done
```

Elaboración propia

Si accedemos, como en ocasiones anteriores, a `http://IP:PUERTO` siendo en mi caso `http://192.168.0.201:8081` veremos un mensaje de error que nos indica que utilicemos el protocolo https así pues accederemos a `https://IP:PUERTO` que en mi caso es `https://192.168.0.201:8081` es probable que nos aparezca un mensaje de certificado no válido, no hay problema, le indicamos que continúe y veremos algo como:



Elaboración propia

Funcionamiento

Ahora la contraseña que genera nextcloud. En mi imagen fíjate que aparece justo encima del botón. Tras acceder llegaremos a una pantalla como la siguiente en la cual deberemos indicar un

dominio que apunte a nuestra raspberry pi (podemos utilizar alguno que hayamos configurado con duckdns)

[Log out](#)

Nextcloud AIO v4.6.2

Nextcloud AIO stands for Nextcloud All In One and provides easy deployment and maintenance with most features included in this one Nextcloud instance.

New AIO instance

Please type in the domain that will be used for Nextcloud if you want to create a new instance:

Make sure that this server is reachable on Port 443 and you've correctly set up the DNS config for the domain that you enter.

If you have a dynamic IP-address, you can use e.g. [DDclient](#) with a compatible domain provider for DNS updates.

Hint: If the domain validation fails but you are completely sure that you've configured everything correctly, you may skip the domain validation by following [this documentation](#).

Restore former AIO instance from backup

You can alternatively restore a former AIO instance from backup.

Please enter the location of the backup archive on your host and the password of the backup archive below:

The folder path that you enter must start with **/** and must **not** end with **/**.

Elaboración propia

Podemos trampear este paso si al final de nuestro fichero docker-compose.yml añadimos:

```
environment:
  SKIP_DOMAIN_VALIDATION: 'true'
```

Para recargar los cambios en el fichero docker-compose.yml deberemos tirar el servicio con `docker-compose down` y volver a levantarlo con `docker-compose up -d`. Si volvemos a acceder a la url anterior veremos algo como:

Nextcloud AIO v4.6.2

Clicking on the button below will download all docker containers and start them. This can take a lot of time depending on your internet connection. Since the overall size is a few GB, this will take around 5-10 min or more. So be aware and patient!

Start containers

Optional addons

In this section you can enable or disable optional addons.

☐ ClamAV (Antivirus backend for Nextcloud, only supported on x64, needs ~1GB additional RAM)

☒ Collabora (Nextcloud Office)

☐ Fulltextsearch (needs ~1GB additional RAM)

☐ Imaginary (for previews of heic, heif, illustrator, pdf, svg, tiff and webp)

☒ Nextcloud Talk (needs ports 3478/TCP and 3478/UDP open in your firewall/router)

Minimal system requirements: When any optional addon is enabled, at least 2GB RAM, a dual-core CPU and 40GB system storage are required. When enabling ClamAV or Fulltextsearch, at least 3GB RAM are required. When enabling everything, at least 4GB RAM are required. Recommended are at least 1GB more RAM than the minimal requirement.

Timezone change

In order to get the correct time values for certain Nextcloud features, it makes sense to set the timezone for Nextcloud to the one that your users mainly use. Please note that this setting does not apply to the mastercontainer and any backup option.

You can configure the timezone for Nextcloud below:

Elaboración propia

Si pulsamos en `start containers` (le costará un buen rato) se pondrá en marcha el servicio y ya podremos disfrutar del mismo.

Quizás una Raspberry Pi modelo 4 de 4 GB, como es mi caso, resulta insuficiente para un servicio tan potente como el descrito. Si ejecutamos el comando `htop` veremos que está saturada a mas no poder.

4.2 Haciendo uso del pinout

Llegó el momento de *mancharnos las manos*

Todos los montajes **deben hacerse** con la Raspberry Pi apagada.

Cuestiones previas

Antes de comenzar a trabajar con la raspberry Pi necesitamos tener claras algunas cuestiones como el pinout del modelo con el que estemos trabajando. Así, el pinout del modelo 4 es el siguiente:

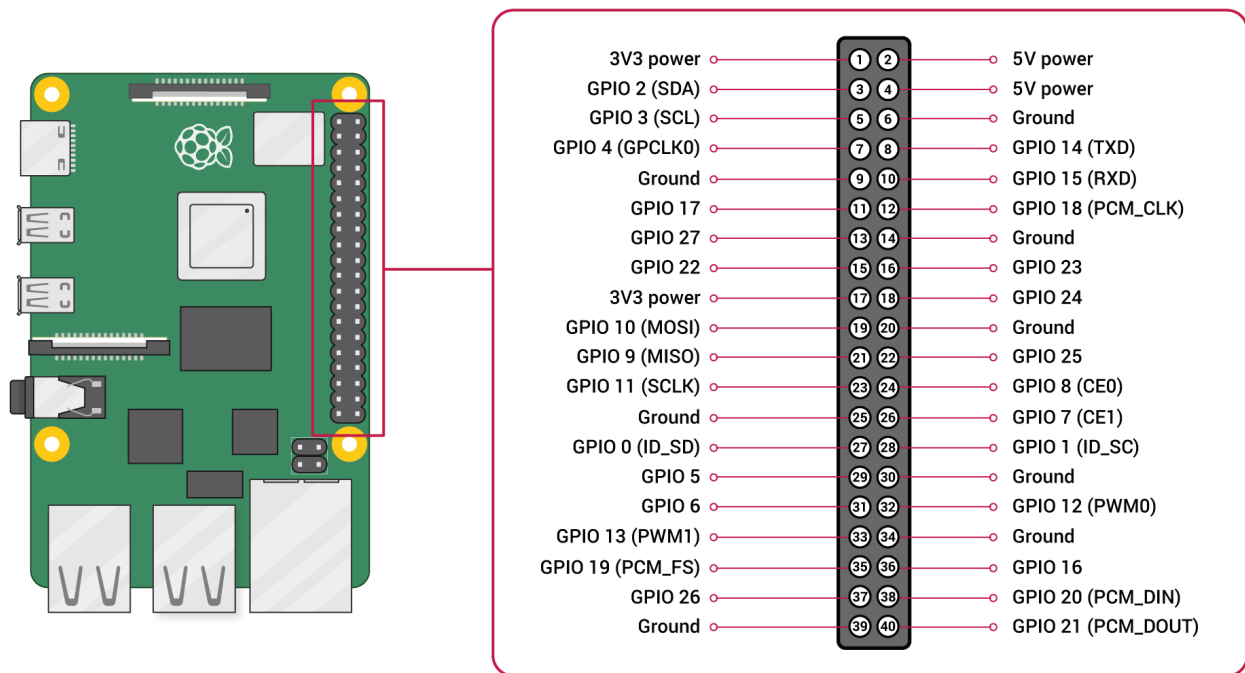


Imagen obtenida de <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>

Una cosa que han hecho muy bien desde Raspberry Pi es mantener la **retrocompatibilidad** entre modelos así el pinout del modelo 2 B es el mismo que es de este modelo.

Para las prácticas que realizaré a continuación utilizaré una Raspberry Pi modelo 2 B. Las mismas deberían ser totalmente compatibles

Vamos a hacer uso de resistencias para algunos montajes por ello te dejo a mano el código de colores de las resistencias:


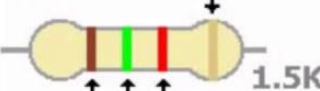
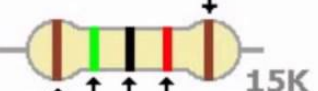
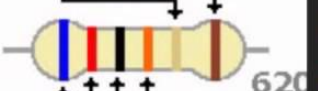
			
Código de Colores	Resistencias de 4 Bandas	Resistencias de 5 Bandas	Resistencias de 6 Bandas
			
	0×1	0×1	0×1
	1×10	1×10	1×10
	2×100	2×100	2×100
	3×1000	3×1000	3×1000
	4×10000	4×10000	4×10000
	5×100000	5×100000	5×100000
	6×1000000	6×1000000	6×1000000
	$7 \div 10$	$7 \div 10$	$7 \div 10$
	$8 \div 100$	$8 \div 100$	$8 \div 100$
	9	9	9

Imagen obtenida de <https://i.ytimg.com/vi/ox8Su0lyFXo/maxresdefault.jpg>

También necesitaremos conocer los diferentes segmentos de un display

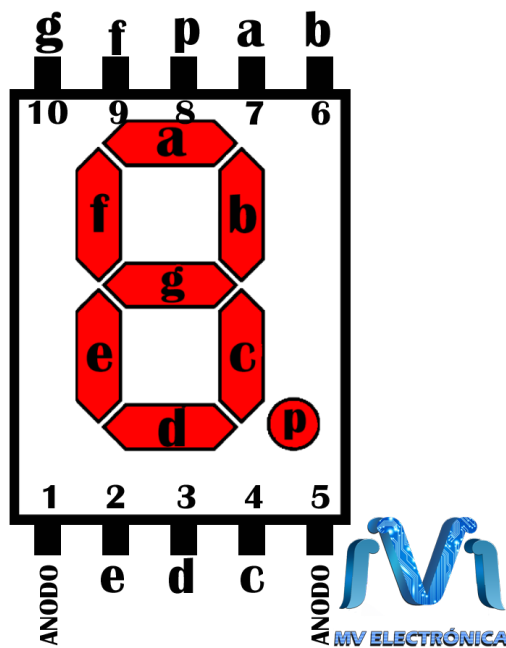


Imagen obtenida de

<https://mvelectronica.com/storage/app/OPTOELECTRONICA/DISPLAY/D7S3A/2.png>

Lo último que tenemos que tener claro es el patillaje de un diodo LED

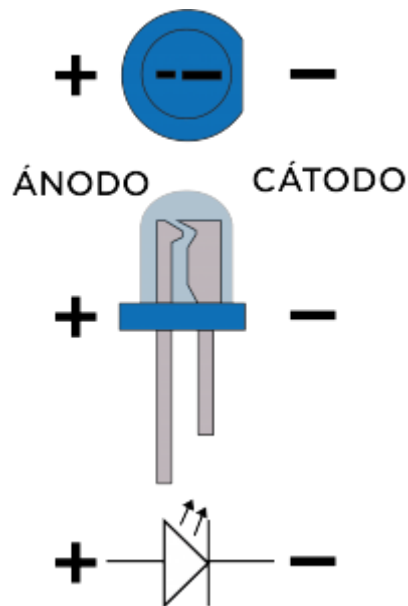


Imagen obtenida de <https://ebotics.com/es/actividad/proyecto-no-2-controlar-el-brillo-del-led/>

También recuerda:

- Ley de Ohm: $V = R * I$ (Tensión es igual a Resistencia por Intensidad) en sus respectivas unidades:
 - Tensión se mide en Voltios (V)

- Resistencia se mide en Ohmios (Ω)
- Intensidad se mide en amperios (A)
- Un diodo sólo deja pasar la corriente en un determinado sentido por ello verifica que si no se enciende está conectado en la posición correcta (si no luce también puede ser que esté quemado)

Ejemplo de cálculo

Si queremos actuar sobre algún dispositivo lo haremos a través de los pines que podemos encender y apagar a través del software. Si recurrimos a la [documentación de Raspberry Pi](#) indican "A GPIO pin designated as an output pin can be set to high (3.3V) or low (0V)." Es decir, cuando establezcamos que un pin está encendido (a HIGH, a 1, son todo sinónimos) significa que estaremos estableciendo que en ese pin hay 3,3V mientras que si establecemos que un pin está apagado (a LOW, a 0, son todo sinónimos) significa que estamos estableciendo que en ese pin hay 0V. Además la corriente máxima con la que pueden funcionar un GPIO en la Raspberry es 16mA.

Así, si queremos utilizar un diodo LED rojo que funciona con 1,7 V y 20mA tendremos un exceso de 1,6V (los 3,3V que genera la salida de la Raspberry menos los 1,7V que requiere el LED rojo) ese exceso de tensión tendremos que *dárselos* a otro componente para que no se queme el LED por ello en nuestro diseño meteremos una resistencia, que no tendrá mas objetivo que ser ella la que consume ese exceso de 1,6V. ¿Y de cuánto tendrá que ser esa resistencia? Sabemos que en esa resistencia tiene que haber 1,6V y que por ella van a circular 16mA = 0,016A (lo ideal sería 20mA pero la Raspberry nos lo limita a 16mA), usamos la Ley de Ohm $R = V/I = 1,6/0,016 = 100 \Omega$. Por lo que junto a nuestro led rojo deberemos usar una resistencia de 100 Ω que coincide con un valor standard comercial. Si el valor calculado no coincide con un valor comercial, hay que elegir el inmediato superior (el inferior proporcionaría más corriente y tenemos la limitación de los 16mA de la Raspberry que no queremos quemarlo).

La potencia que "sufrirá" esa resistencia será $P = VI = 1.6V \times 16mA = 27mW$ por lo tanto cualquier resistencia comercial es válida pues lo mínimo es 1/8W la más común son las 1/4W.

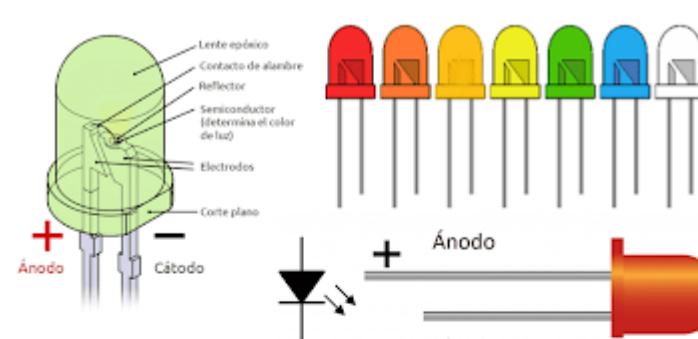
De la misma forma podemos hacer los cálculos para otros colores de Leds, ([aquí para descargar la hoja de cálculo](#)) pero como podemos ver, algunos colores no funcionarían con la Raspberry pues necesitan más tensión de la que proporciona la Raspberry

Color del diodo	Vd	Vcc de la raspberry	mA fabricante	mA Raspberry	$R = (V_{cc} - V_d)/I$ en Ohm	Valor comercial próximo que lo supere
Rojo	1,7	3,3	20	16	100	100
Rojo alta eficiencia	1,9	3,3	20	16	87,5	100
Anaranjado y amarillo	2	3,3	20	16	81,25	82
Verde	2,1	3,3	20	16	75	82
Blanco, verde brillante y azul	3,4	no funciona	12			
Azul brillante y especiales	4,6	no funciona	10			
Display	2	3,3	20	16	81,25	82

Fuente de datos Wikipedia https://es.wikipedia.org/wiki/Circuito_de_LED

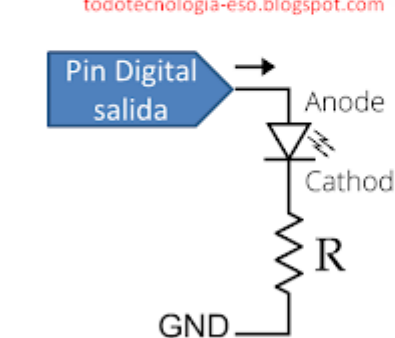
Normalmente en placas como Arduino trabaja a 5V por lo que las resistencias son mayores :

DIODO LED



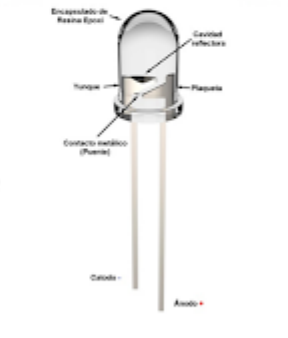
Ánodo (+) Cátodo (-)

todotecnologia-eso.blogspot.com



Pin Digital salida → Anode
Cathode → R → GND

Lógica positiva (activo a nivel alto)



Resistencia de protección: R (Para 5V)

- 220 Ω
- 330 Ω
- 470 Ω


Características diodo LED											
Tipo Receptor	Receptor luminoso Diodo emisor de luz (Light Emitting Diode)										
Tipo actuador Arduino	Salida Digital										
Diámetro:	3 – 5- 10 mm										
Símbolo											
Polarizado	Sí										
Intensidad:	10 mA ~ 20 mA										
Tensión de trabajo típica: (Caída de tensión en conducción-polarización directa)	<table><tr><td>Rojo</td><td>1,7-1,9V /20 mA</td></tr><tr><td>Ámbar</td><td>2 V /20 mA</td></tr><tr><td>Verde</td><td>2,1 V /20 mA</td></tr><tr><td>Blanco</td><td>3,4 V / 12 mA</td></tr><tr><td>Azul</td><td>4,6 V /10 mA</td></tr></table>	Rojo	1,7-1,9V /20 mA	Ámbar	2 V /20 mA	Verde	2,1 V /20 mA	Blanco	3,4 V / 12 mA	Azul	4,6 V /10 mA
Rojo	1,7-1,9V /20 mA										
Ámbar	2 V /20 mA										
Verde	2,1 V /20 mA										
Blanco	3,4 V / 12 mA										
Azul	4,6 V /10 mA										
Potencia	34 mW ~ 46 mW										
Tensión de ruptura (inversa máx.)	2 V ~ 5 V										
Vida útil	30.000 ~ 100.000 h										

Imagen de <https://todotecnologia-eso.blogspot.com/> Licencia CC-BY-SA-NC

Encender y apagar un led

Materiales

- 1 led
- 1 resistencia de 100Ω / 1/4W
- 2 cables
- placa de conexiones

Conexiones

1. Resistencia al cátodo del led
2. ánodo del led a pin 7 (pin, no GPIO)
3. Resistencia a pin 25 (tierra)

Programa

Nos dirigiremos a la terminal y allí escribiremos

```
nano led.py
```

con ello se abrirá el editor nano y estaremos trabajando sobre el fichero led.py que si no existe se creará y si existe trabajaremos sobre el existente. A continuación escribiremos el código python que aparece a continuación:

```
# Importamos la biblioteca que nos permite pausar la ejecución del programa
import time

# Importamos la biblioteca que nos permite conectar a los GPIO pins
# otros dispositivos. GPIO -> General Purpose Input Output
import RPi.GPIO as GPIO

# A partir de aquí configuramos la placa
# 1ero: borramos cualquier configuración previa
GPIO.cleanup()

# 2ndo: establecemos el modo de trabajo de la raspi
GPIO.setmode(GPIO.BOARD)

# 3ero: Indicamos que el pin 7 actuará como salida
GPIO.setup(7,GPIO.OUT)

# Bucle infinito
while True:

    # Escribo en el terminal el texto enciendo
    print("enciendo")

    # Pongo tensión en el pin 7
    GPIO.output(7,GPIO.HIGH)

    # Detengo la ejecución del programa 2 segundos
    time.sleep(2)

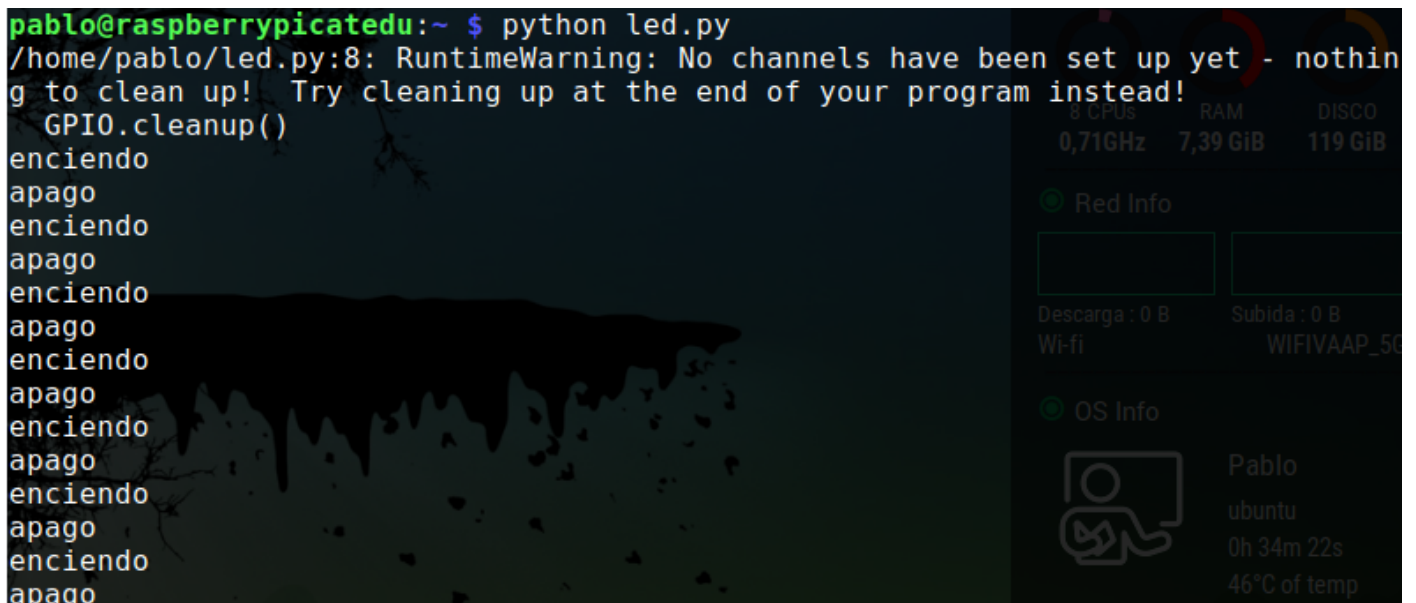
    # Escribo en el terminal
    print("apago")

    # Quito la tensión del pin 7
    GPIO.output(7,GPIO.LOW)

    # Pauso la ejecución del programa 2 segundos
    time.sleep(2)
```

para salir del editor pulsaremos `control + x` y aceptaremos los cambios. Tras ello, si ejecutamos `ls -l` veremos listado el directorio en el que nos encontramos y ahí veremos nuestro fichero. Lo ejecutaremos escribiendo `python led.py`

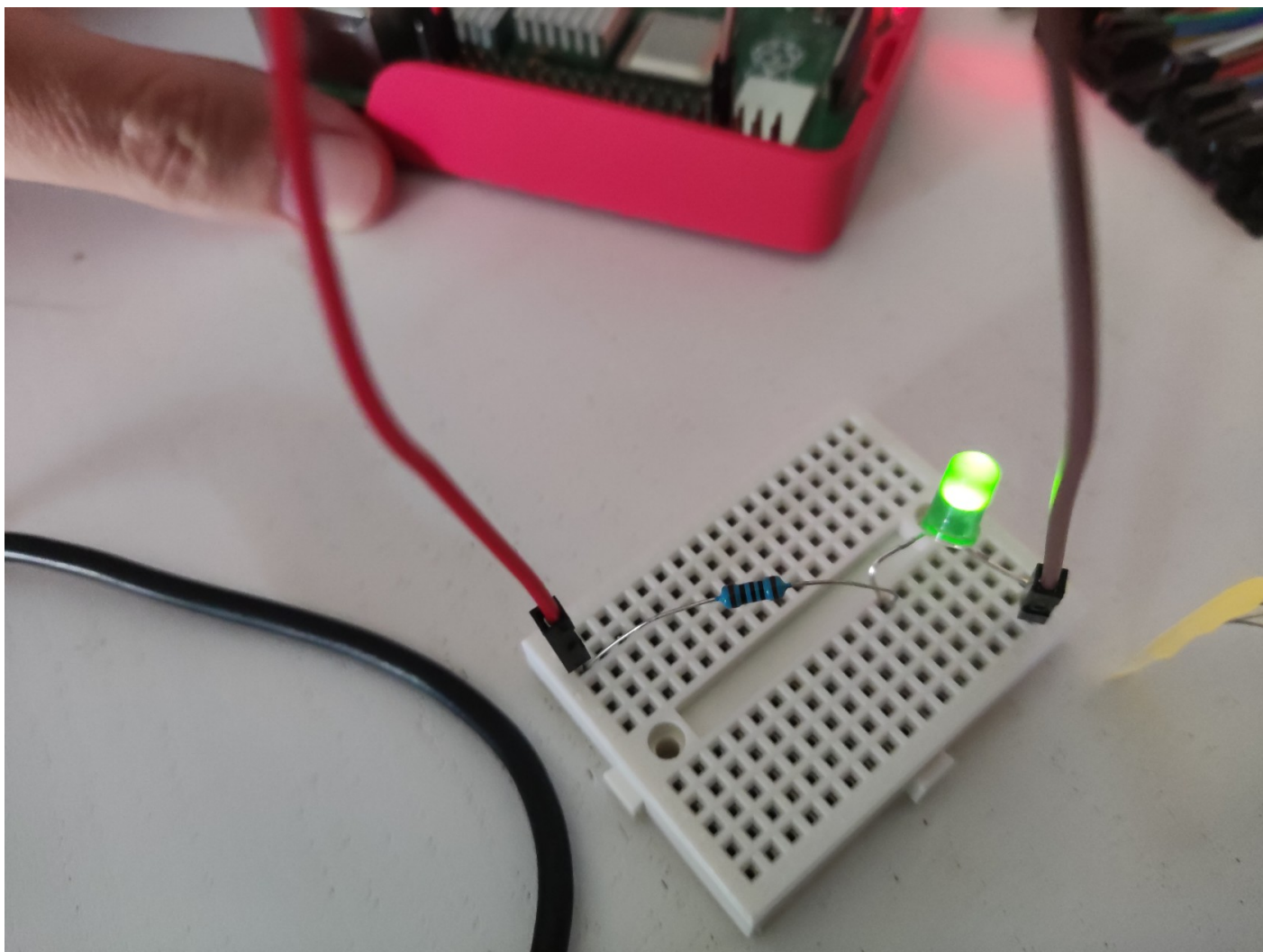
En el terminal veremos algo similar a



```
pablo@raspberrypicatedu:~ $ python led.py
/home/pablo/led.py:8: RuntimeWarning: No channels have been set up yet - nothing to clean up! Try cleaning up at the end of your program instead!
  GPIO.cleanup()
enciendo
apago
enciendo
apago
enciendo
apago
enciendo
apago
enciendo
apago
enciendo
apago
enciendo
apago
```

Elaboración propia

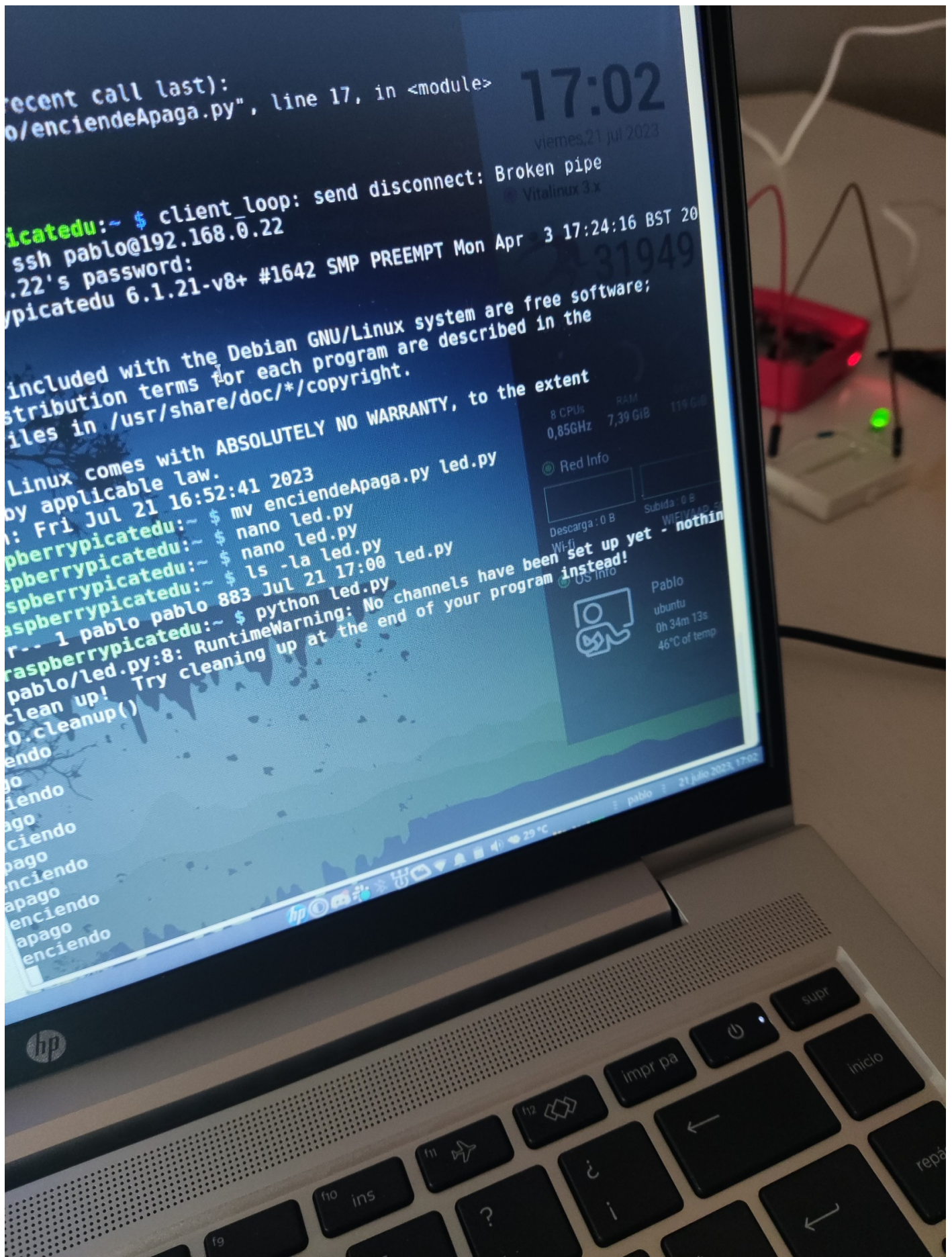
Si las conexiones que hemos realizado son correctas veremos como el led va encendiéndose y apagándose. Para detener la ejecución del programa pulsaremos `control + c`. Dejo a continuación alguna foto del programa funcionando:



Conexiones en la protoboard. Elaboración propia



Conexiones en la Raspberry Pi. Elaboración propia



Mensaje de led encendido y led encendido. Elaboración propia



Mensaje de led apagado y led apagado. Elaboración propia

Semáforo

Materiales

- 1 led rojo
- 1 led amarillo
- 1 led verde
- 1 resistencia de 100 Ω y dos de 82 Ω pero si se utilizan todos de 100 Ω no pasa nada.
- 4 cables
- placa de conexiones

Conexiones

1. Una resistencia al cátodo de cada led
2. ánodo del led rojo a pin 3 (pin, no GPIO)
3. ánodo del led amarillo a pin 5 (pin, no GPIO)
4. ánodo del led verde a pin 7 (pin, no GPIO)
5. Resistencias a pin 25 (tierra)

Programa

Al igual que en el caso anterior crearemos el fichero con nano y lo ejecutaremos con python. Esa parte no varía por lo que no me repito. El nombre que le pongáis al programa no es importante.

```

1  # Importamos la libreria que nos permite pausar la ejecucion del programa
2  import time
3  # Importamos la libreria que nos permite conectar la raspberry con otros
4  # dispositivos a traves de los GPIO (General Purpose Input Output) pins
5  import RPi.GPIO as GPIO
6  # Ahora vamos a configurar los GPIO pins
7  # lero Borramos cualquier configuracion que hubiese
8  GPIO.cleanup()
9  # Establecemos el modo de trabajo de los pines
10 GPIO.setmode(GPIO.BOARD)
11 # Establecemos el pin 3 para actuar como salida sera nuestro rojo
12 GPIO.setup(3,GPIO.OUT)
13 # Establecemos el pin 5 para actuar como salida sera nuestro amarillo
14 GPIO.setup(5,GPIO.OUT)
15 # Establecemos el pin 7 para actuar como salida sera nuestro verde
16 GPIO.setup(7,GPIO.OUT)
17 # Bucle infinito
18 while True:
19     # verde
20     GPIO.output(7,GPIO.HIGH)
21     time.sleep(4)
22     GPIO.output(7,GPIO.LOW)
23     # amarillo
24     GPIO.output(5,GPIO.HIGH)
25     time.sleep(2)
26     GPIO.output(5,GPIO.LOW)
27     # rojo
28     GPIO.output(3,GPIO.HIGH)
29     time.sleep(3)
30     GPIO.output(3,GPIO.LOW)
31 # ejecutaremos el programa del siguiente modo
32 # sudo python semaforo.py

```

Elaboración propia

Display de 7 segmentos

Esta práctica variará en función de si usamos un display de cátodo común o de ánodo común, tenlo en cuenta.

Materiales

- Display de 7 segmentos
- 4 resistencias de 82 Ω o 100 Ω
- 5 cables
- placa de conexiones

Conexiones

Si utilizas un display de cátodo común serán las siguientes:

- Cátodo del display a pin 25 (tierra)
- Pin 3 a resistencia y esta a segmento f.
- Pin 5 a resistencia y esta a segmento e.
- Pin 7 a resistencia y esta a segmento b.
- Pin 11 a resistencia y esta a segmento c.

Si utilizas un display de ánodo común serán las siguientes:

- Ánodo del display a pin 1 (3,3V)
- Pin 3 a resistencia y esta a segmento f.
- Pin 5 a resistencia y esta a segmento e.
- Pin 7 a resistencia y esta a segmento b.
- Pin 11 a resistencia y esta a segmento c.

Programa

Al igual que en el caso anterior crearemos el fichero con nano y lo ejecutaremos con python. Esa parte no varía por lo que no me repito. El nombre que le pongáis al programa no es importante.

El programa encenderá un lateral del *display* y luego otro de modo infinito. Te propongo que modifiques el programa y las conexiones para convertir esta práctica en una cuenta atrás.


```

1  # Importamos la libreria que nos permite pausar la ejecucion del programa
2  import time
3  # Importamos la libreria que nos permite conectar la raspberry con otros
4  # dispositivos a traves de los GPIO (General Purpose Input Output) pins
5  import RPi.GPIO as GPIO
6  # Ahora vamos a configurar los GPIO pins
7  # lero Borramos cualquier configuracion que hubiese
8  GPIO.cleanup()
9  # Establecemos el modo de trabajo de los pines
10 GPIO.setmode(GPIO.BOARD)
11 # Establecemos los pines a usar como salida
12 GPIO.setup(3,GPIO.OUT)
13 GPIO.setup(5,GPIO.OUT)
14 GPIO.setup(7,GPIO.OUT)
15 GPIO.setup(11,GPIO.OUT)
16 # Bucle infinito
17 while True:
18     # encendemos un lateral
19     GPIO.output(3,GPIO.HIGH)
20     GPIO.output(5,GPIO.HIGH)
21     GPIO.output(7,GPIO.LOW)
22     GPIO.output(11,GPIO.LOW)
23     # espera 2 segundos
24     time.sleep(3)
25     # encendemos el otro lateral
26     GPIO.output(3,GPIO.LOW)
27     GPIO.output(5,GPIO.LOW)
28     GPIO.output(7,GPIO.HIGH)
29     GPIO.output(11,GPIO.HIGH)
30     # espera 3 segundos
31     time.sleep(3)
32 # ejecutaremos el programa del siguiente modo
33 # sudo python contador.py

```

Elaboración propia

Otras prácticas

Con mi alumnado de FP básica otras prácticas que he llevado a cabo han sido:

- encender el led cuando no hay luz (haciendo uso del LDR)
- un detector de humedad (que suene un sonido y/o se enciende una luz cuando el circuito se cierra con la humedad)

En internet existen multitud de prácticas para realizar con la Raspberry Pi.

4.3 Domótica

De nuevo vamos a tomar prestada una definición de la wikipedia:

Se llama domótica a los sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta facilidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.

<https://es.wikipedia.org/wiki/Dom%C3%B3tica>

En nuestro caso, dado que vamos a trabajar con una Raspberry Pi, vamos a ver una solución que yo considero perfecta para este ordenador. Se trata de Home Assistant pero antes de entrar en el meollo de la cuestión vamos a hablar de protocolos.

Protocolos de comunicación

En informática y telecomunicación, **un protocolo de comunicaciones es un sistema de reglas que permiten que dos o más entidades** (computadoras, teléfonos celulares, etc.) **de un sistema de comunicación se comuniquen entre ellas para transmitir información por medio de cualquier tipo de variación de una magnitud física.** Se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como también los posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos.

También se define como un conjunto de normas que permite la comunicación entre ordenadores, estableciendo la forma de identificación de estos en la red, la forma de transmisión de los datos y la forma en que la información debe procesarse.

Los sistemas de comunicación utilizan formatos bien definidos (protocolo) para intercambiar mensajes. Cada mensaje tiene un significado exacto destinado a obtener una respuesta de un rango de posibles respuestas predeterminadas para esa situación en particular. Normalmente, el comportamiento especificado es independiente de cómo se va a implementar. Los protocolos de comunicación tienen que estar acordados por las partes involucradas. Para llegar a dicho acuerdo, un protocolo puede ser desarrollado dentro de estándar técnico(...)

https://es.wikipedia.org/wiki/Protocolo_de_comunicaciones

En este punto puede ser que os estéis preguntando por qué os estoy contando qué son los protocolos de comunicaciones. En este apartado vamos a trabajar con distintos *gadget*, sensores, que pueden utilizar diferentes protocolos. Además, algunos de los sensores que voy a comentar funcionan con protocolos que la Raspberry Pi no soporta por lo que requieren de algún hardware adicional. Vamos a enumerar algunos de los protocolos que me parecen mas relevantes de cara a usar domótica en la Raspberry Pi:

- **Wi-Fi:** Soportado directamente por la raspberry pi.
- **Bluetooth:** Soportado directamente por la raspberry pi.
- **Zigbee:** No soportado por la raspberry pi directamente. Se trata de un protocolo de comunicación inalámbrica de bajo consumo. Ideal para dispositivos que requieren comunicaciones seguras con baja tasa de envío de datos y una gran vida útil de sus baterías. Necesita de un adaptador (USB) que conectaremos a la Raspberry Pi y que le dará a la Raspberry la posibilidad de comunicarse a través de este protocolo.
- **MQTT** (alias *mosquito*): es un protocolo de mensajería ligero con un consumo muy bajo de energía. Requiere de un *broker de mensajería*.

Zigbee

Mi adaptador zigbee-USB, un CC2531, lo adquirí por unos 6 € aquí <https://es.aliexpress.com/item/32971386349.html> requiere ser programado (*flasheado*). Se puede flashear "facilmente" con un dispositivo similar a <https://es.aliexpress.com/item/32976509073.html> que cuesta unos 8€ y estando obligados/as a utilizar Windows. También se puede flashear desde la Raspberry Pi utilizando este método https://www.zigbee2mqtt.io/guide/adapters/flashing/alternative_flashing_methods.html (yo no lo he probado)

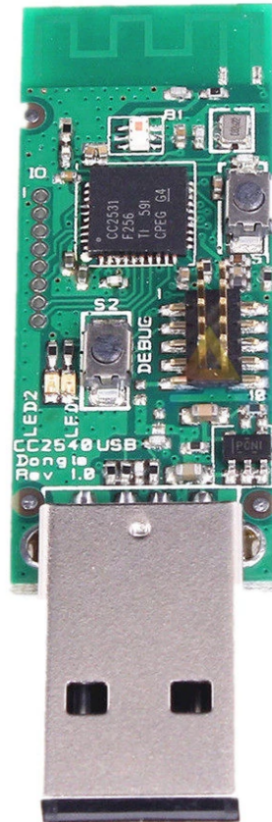


Imagen obtenida de <https://es.aliexpress.com/item/32971386349.html>

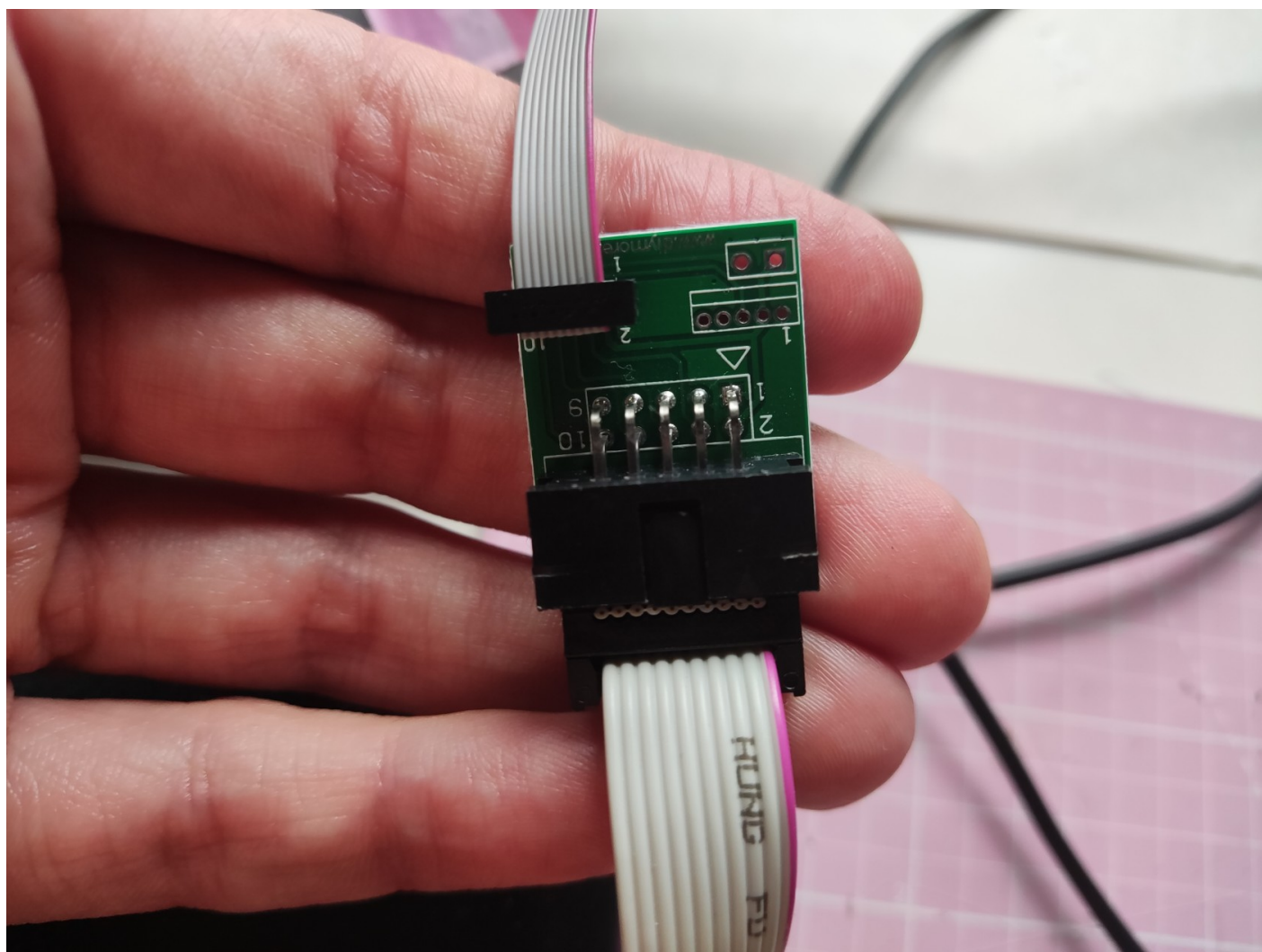
Puede ser que el dispositivo que tengas entre tus manos ya esté flasheado si otro compañero/a lo ha flasheado en una edición anterior del curso. Si así lo deseas puedes volver a flashearlos sin dañarlo.

Os dejo un videotutorial que explica como *flashear* el dispositivo por si os animáis a utilizar la tecnología Zigbee. Os dejo también el enlace al vídeo ya que hay veces que no carga incrustado en este manual <https://youtu.be/70Eav-prgMk> En esta página del curso he anexionado (aparecen arriba a la izquierda) también los ficheros necesarios por si en el futuro dejan de estar disponibles en su ubicación actual.

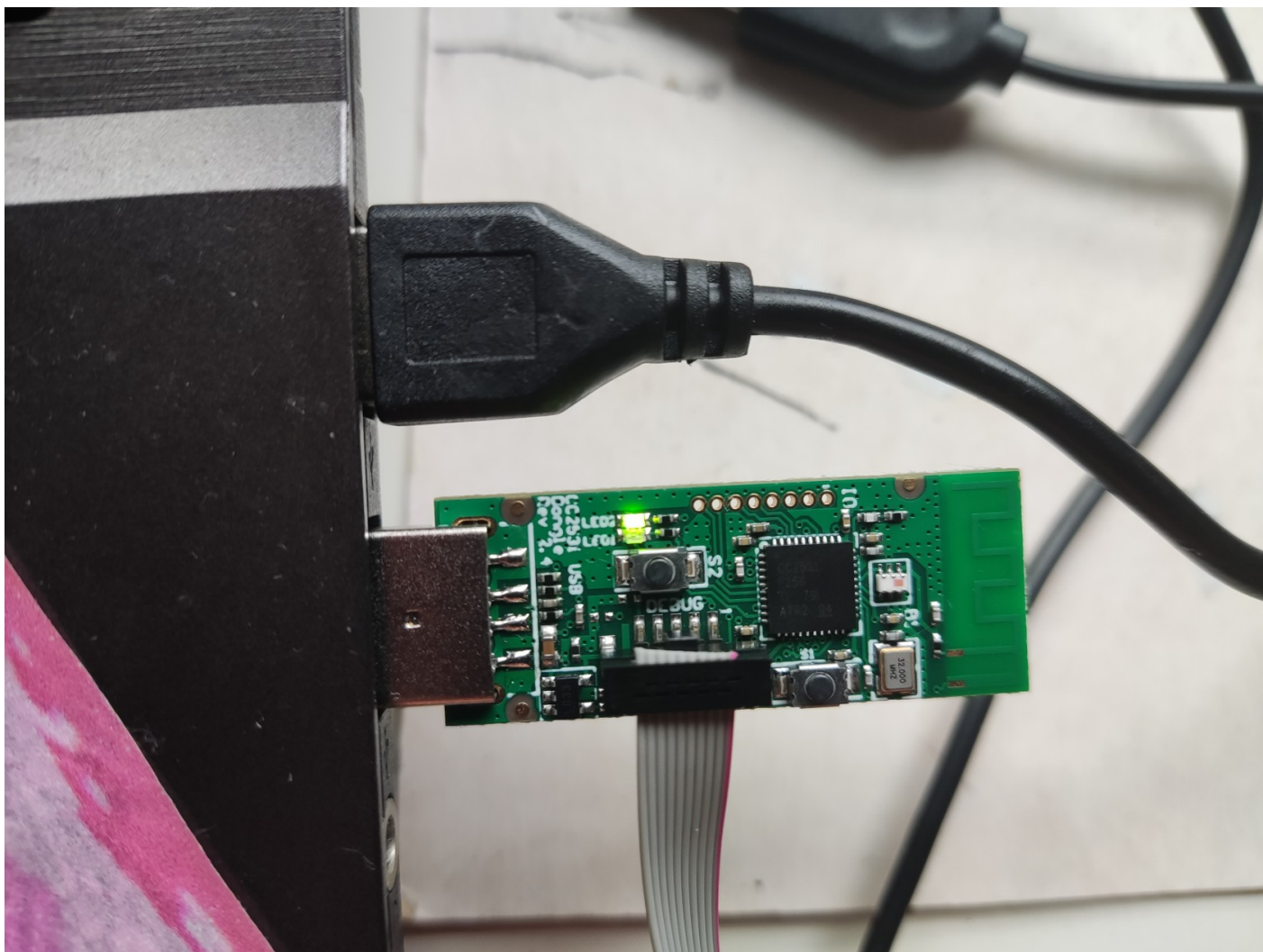
<https://www.youtube.com/embed/70Eav-prgMk>

Os dejo alguna foto de las conexiones del proceso de flasheo por si no lo veis correctamente en el vídeo:









Elaboración propia

El ccdebugger en ocasiones le cuesta MUCHO cambiar el led de rojo a verde. Es cuestión de insistir, reiniciar,... La primera vez que lo hice me costó 2 minutos. La segunda vez que lo hice me costó 20 minutos.

Os dejo unas capturas de pantalla del programa de flasheo:



What do you want to program?

Program CCxxxx SoC or MSP430

System-on-Chip | MSP430

EB ID	Chip type	EB type	EB firmware ID	EB firmware rev
2407	CC2531	CC Debugger	05CC	0041

Interface:

Fast

Flash image: C:\Users\Pablo\Desktop\Pablo - Borrarme\Z-Stack-firmware-master\coordina...

Read IEEE | Write IEEE

Location

☒ Primary ☐ Secondary

IEEE 0x

☒ Retain IEEE address when reprogramming the chip

View Info Page

Actions

- ☐ Erase
- ☐ Erase and program
- ☒ Erase, program and verify
- ☐ Append and verify
- ☐ Verify against hex-file
- ☐ Read flash into hex-file

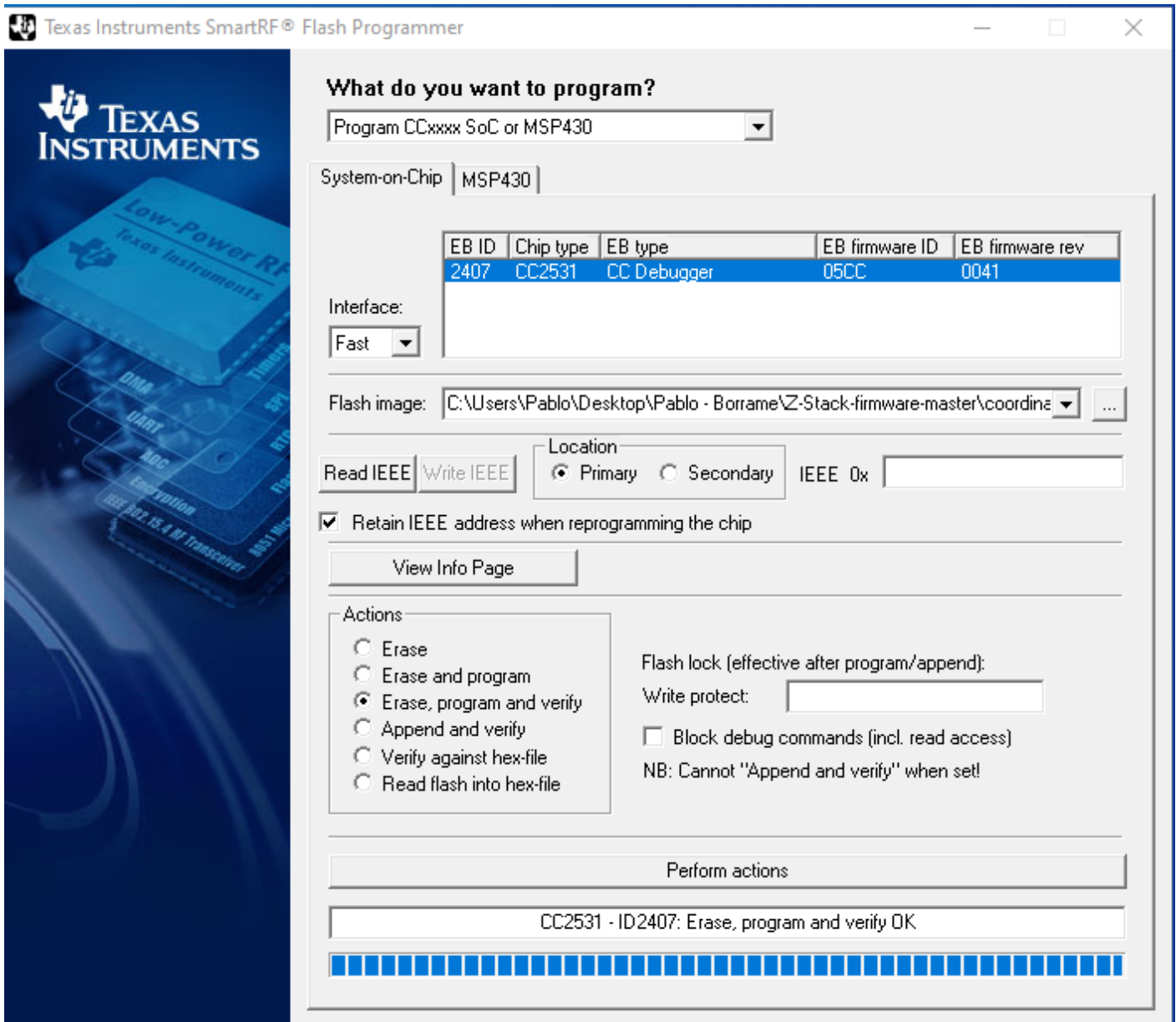
Flash lock (effective after program/append):

Write protect:

☐ Block debug commands (incl. read access)

NB: Cannot "Append and verify" when set!

Perform actions



Elaboración propia

Tipos de dispositivos IoT

A la hora de adquirir dispositivos **IoT** los encontraremos de 2 tipos:

- conectados a la luz
- con batería/pilas

en algunos casos tiene sentido que estén conectados a la luz, es el caso de un termostato de la calefacción o un motor para subir/bajar las persianas, mientras que otros, como un sensor de temperatura o un actuador de radiador, tiene mas sentido que funcionen con batería/pilas en lugar de tener que llevar un enchufe a cada radiador o a cada punto dónde queramos medir

Además de la clasificación anterior, nos encontraremos básicamente dispositivos de 2 tipos: Wi-Fi y Zigbee pero no Bluetooth ¿por qué? por el consumo. Los dispositivos que utilizan el protocolo Bluetooth consumen mucha mas batería que los que usan, por ejemplo, zigbee. Nadie quiere un dispositivo IoT al que tiene que cambiarle las pilas cada mes cuando existe uno equivalente al que puede cambiarle la pila cada 2 años.

Privacidad

Una cuestión que yo considero MUY importante es el que el acceso a nuestros datos o a la funcionalidad del hardware no requiera de tener acceso a la nube de una determinada empresa por cuestiones ya no solo de privacidad sino por el hecho de que si esa empresa cierra su nube o establece un servicio de pago estamos condenados a pagar o cambiar el dispositivo con las implicaciones medioambientales y económicas que ello supone. Muchos fabricantes, como [tuya](#), plantean que para usar sus dispositivos Zigbee debes adquirir su hub (concentrador) y posteriormente este hub, a través de su app o web, te dará los datos que recopila o te permitirá interactuar con tus dispositivos. Frente a esta filosofía está la posibilidad de utilizar un software propio al que conectemos nuestros dispositivos (WiFi o Zigbee) y de la compañía que queramos (tuya, sonoff,...) sin depender de nubes de terceros. Además, con soluciones como las que hemos visto en el capítulo 3 para la creación de VPNs, podremos acceder a nuestro dispositivo desde cualquier lugar. Por ello en el siguiente apartado vamos a hablar de Home Assistant.

Home Assistant



Imagen obtenida de <https://design.home-assistant.io/#brand/logo>

Hassio, por si mismo, requeriría de un curso dedicado de una cantidad importante de horas por ello vamos a centrarnos en aquello mas relevante.

Antes de comenzar voy a mostraros mi instalación de Home Assistant con los diferentes sensores, gráficas y automatizaciones que tengo así como por qué empecé a montar este sistema y qué

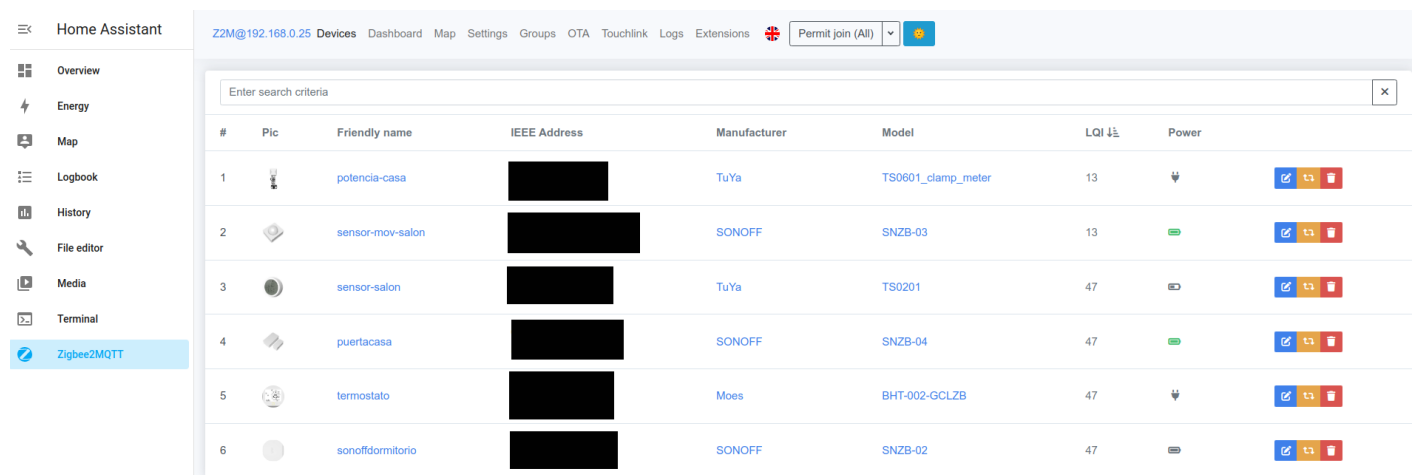
quiero hacer en el futuro.

Inicialmente monté esto para ubicar en cada dormitorio un sensor de temperatura y posteriormete ubicar un actuador de apertura/cierre en cada radiador. La idea era abrir o cerrar cada radiador de modo automático en función de la temperatura de cada estancia. Con posterioridad leí que estos actuadores con tanta apertura/cierre acababan estropeandose y por ello desistí de instalar estos actuadores en los radiadores pero la mecha ya estaba prendida.

En la actualidad tengo los siguientes sensores/actuadores:

- sensor de apertura: en la puerta de entrada a mi domicilio
- sensor de presencia: en la primera estancia a la que se accede desde la puerta de entrada
- sensor de temperatura y humedad en salón
- sensor de temperatura y humedad en el dormitorio principal
- termostato
- sensor de consumo: en el cuadro eléctrico de casa
- sensor de apertura: frigorífico
- receptor/emisor de infrarrojos
- detección de aparatos en red de casa

Todos, excepto el último, funcionan con zigbee mientras que el último hace uso de wifi.

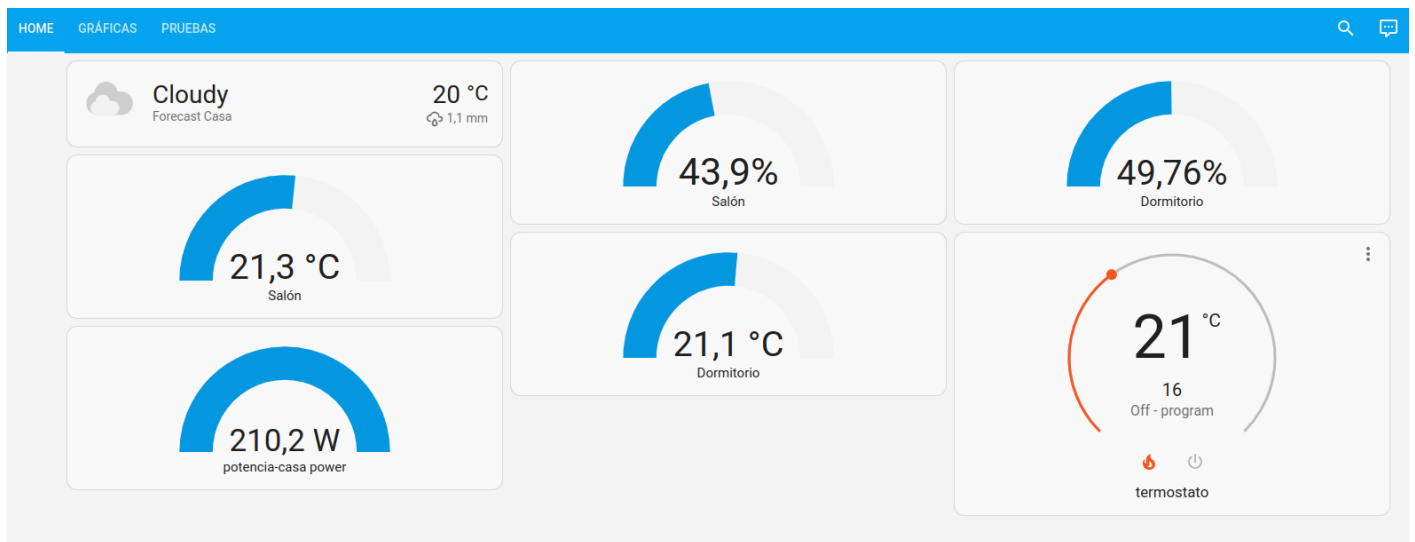


The screenshot shows the Home Assistant interface with the Zigbee2MQTT integration. The left sidebar contains navigation options: Overview, Energy, Map, Logbook, History, File editor, Media, and Terminal. The main area displays a table of devices connected via Zigbee2MQTT. The table has columns for #, Pic, Friendly name, IEEE Address, Manufacturer, Model, LQI, and Power. There are 6 devices listed, each with a corresponding icon, name, and status indicators.

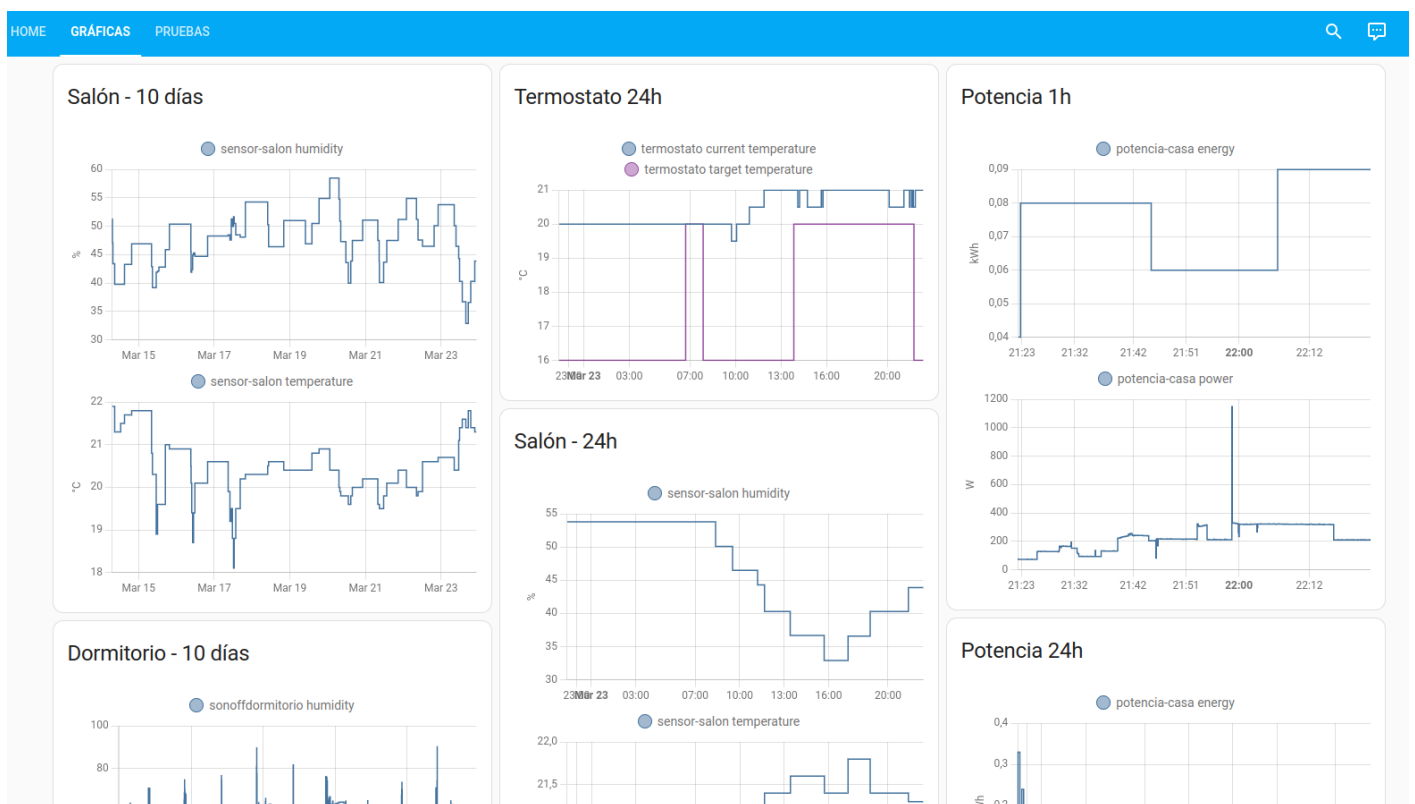
#	Pic	Friendly name	IEEE Address	Manufacturer	Model	LQI	Power
1		potencia-casa	[REDACTED]	TuYa	TS0601_clamp_meter	13	
2		sensor-mov-salon	[REDACTED]	SONOFF	SNZB-03	13	
3		sensor-salon	[REDACTED]	TuYa	TS0201	47	
4		puertacasa	[REDACTED]	SONOFF	SNZB-04	47	
5		termostato	[REDACTED]	Moes	BHT-002-GCLZB	47	
6		sonoffdormitorio	[REDACTED]	SONOFF	SNZB-02	47	

Elaboración propia

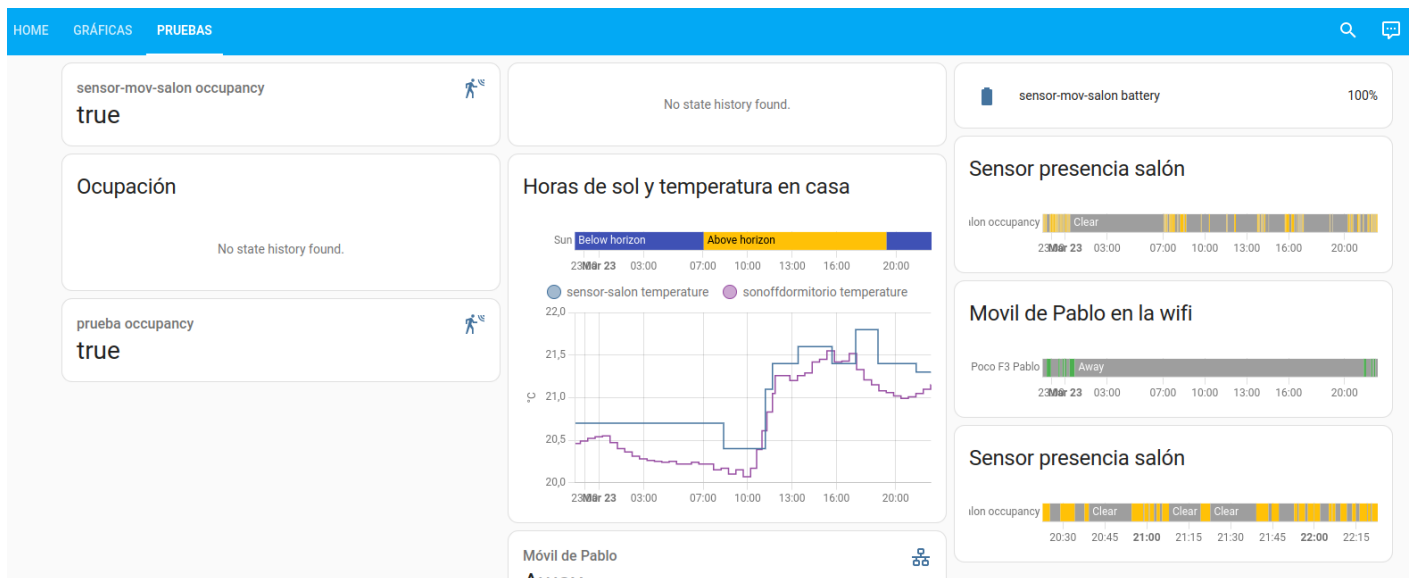
Estos sensores me permiten obtener valores del instante en que realice la consulta pero también históricos. También me permite actuar directamente sobre los sensores como el termostato. Todo ello se configura en unos dashboards muy sencillos como los que vemos a continuación



Elaboración propia



Elaboración propia



Elaboración propia

A partir de lo anterior se pueden crear diferentes automatizaciones. Por ejemplo yo tengo creadas las siguientes:

- Frigo abierto: Mi frigorífico no tiene sensor avisador si te dejas la puerta abierta por lo que intenté hacer un avisador en caso de que estuviese abierto mas de 10s. Mis hijas arrancaron el sensor antes de ponerlo en marcha.
- Frigo cerrado:
- Puerta de casa abierta: Me manda un mensaje por telegram cada vez la puerta de casa se abre. Lo uso a modo de alarma
- Puerta de casa cerrada:
- Termostato temperatura medida cambió por debajo de 19°C: Lo tengo creado para pruebas con el termostato



Automations

Scenes

Scripts

Blueprints



Search



Name

Last triggered



Frigo abierto

Apr 26, 07:39



Frigo cerrado

Dec 11, 12:18



Puerta de casa abierta

1 hour ago



Puerta de casa cerrada

1 hour ago

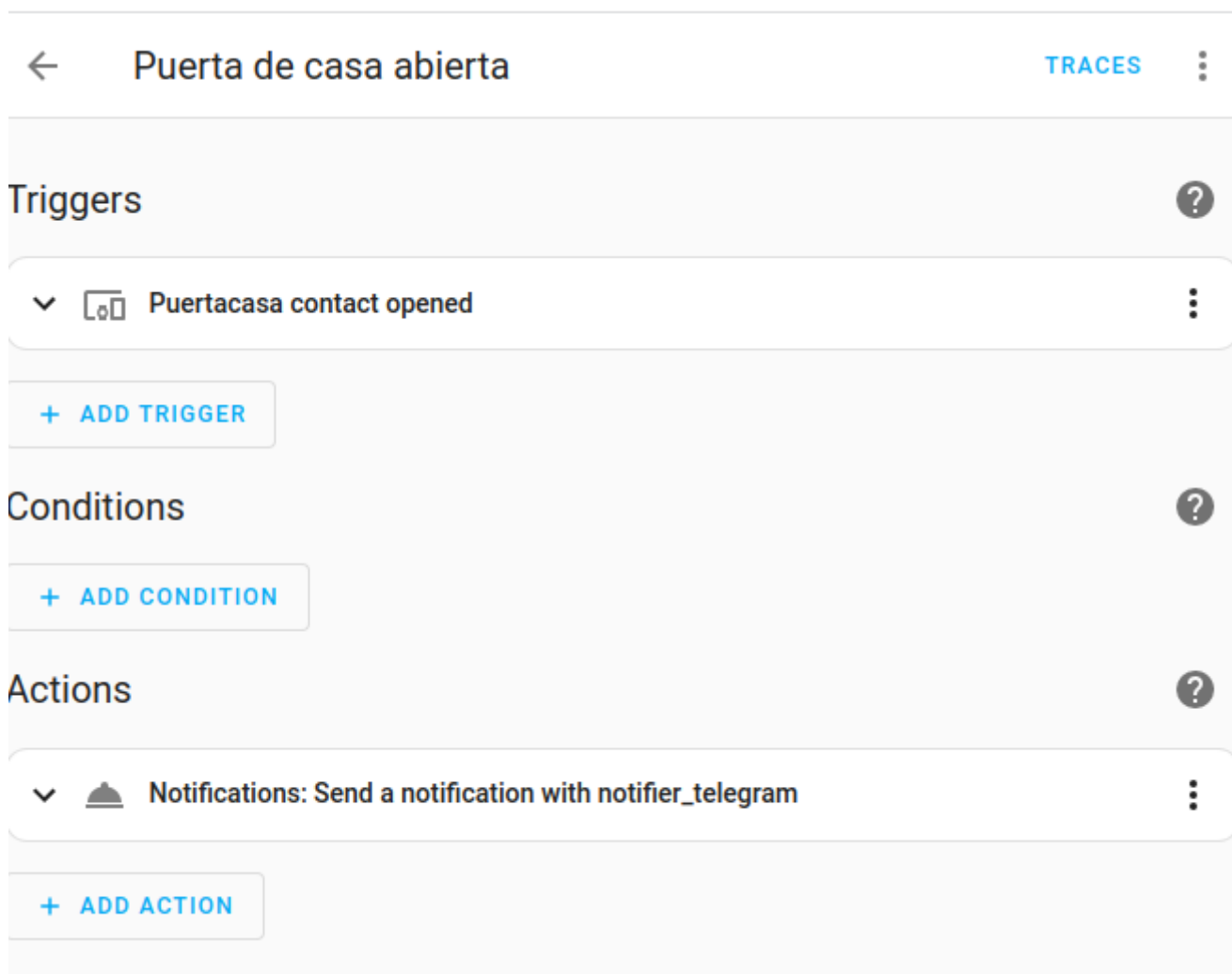


Termostato temperatura medida cambió por ...

2 days ago



Elaboración propia



Elaboración propia

Visto lo que yo tengo es momento de empezar por la **instalación**: Home Assistant (<https://www.home-assistant.io/>) puede ser instalado dockerizado o puede ser instalado directamente como un sistema operativo en una tarjeta SD dedicada. En mi caso opto por la 2ª opción pues como vemos en la imagen posterior el hecho de tenerlo instalado de este modo me permite disfrutar de todas las posibilidades de la herramienta.

Compare Installation Methods

	OS	Container	Core	Supervised
Automations	✓	✓	✓	✓
Dashboards	✓	✓	✓	✓
Integrations	✓	✓	✓	✓
Blueprints	✓	✓	✓	✓
Uses container	✓	✓	✗	✓
Supervisor	✓	✗	✗	✓
Add-ons	✓	✗	✗	✓
Backups	✓	✓ ¹	✓ ¹	✓
Managed OS	✓	✗	✗	✗

1: Backups for Home Assistant Core and Home Assistant Container is provided by the [backup integration](#).

Imagen obtenida de <https://www.home-assistant.io/installation/>

En la dirección <https://www.home-assistant.io/installation/> tenemos a nuestra disposición las diferentes formas de instalación. En el caso de elegir la instalación vía sistema operativo descargaremos la imagen y la instalaremos tal como vimos en el [capítulo 2.1.1](#). Os dejo también un tutorial por si os resulta de utilidad <https://domoticaencasa.es/video-tutorial-instalacion-de-home-assistant/>

<https://www.youtube.com/embed/Ncg9unb2gZs>

Si vais a trabajar con dispositivos Zigbee necesitaréis conocer qué es Zigbee2Mqtt por ello os dejo este enlace <https://domoticaencasa.es/video-funciona-mqtt-integrar-home-assistant/> a un tutorial que explica como funciona y como integrarlo en Home Assistant. Os dejo también un vídeo:

<https://www.youtube.com/embed/YPS5xI6Bx3I>

El protocolo MQTT se usa en robótica educativa, por ejemplo [aquí en el curso ESP32 en el Aula con Arduinoblocks](#)

Si te quedas con ganas de mas voy a recomendarte esta serie de 8 vídeos dónde indican como proceder con hassio y zigbee:

<https://www.youtube.com/embed/FGTLpdgvfPI>

Una función muy interesante es la de que hassio nos envíe notificaciones en Telegram por ello os dejo un enlace a este tutorial <https://domoticaencasa.es/home-assistant-9-notificaciones-telegram/> dónde explica como hacerlo. Este canal os puede resultar también de gran utilidad para los servicios que hemos desplegado a lo largo del capítulo 3.

Integraciones

Si además de las integraciones que pone por defecto a tu disposición Home Assistant quieres hacer uso de otras no oficiales puedes instalar HACS que nos permite acceder a cientos, si no miles, de proyectos que la comunidad ha creado. Para instalar HACS es suficiente con seguir estas instrucciones:

<https://www.youtube.com/embed/4-ujjhKU8CE>