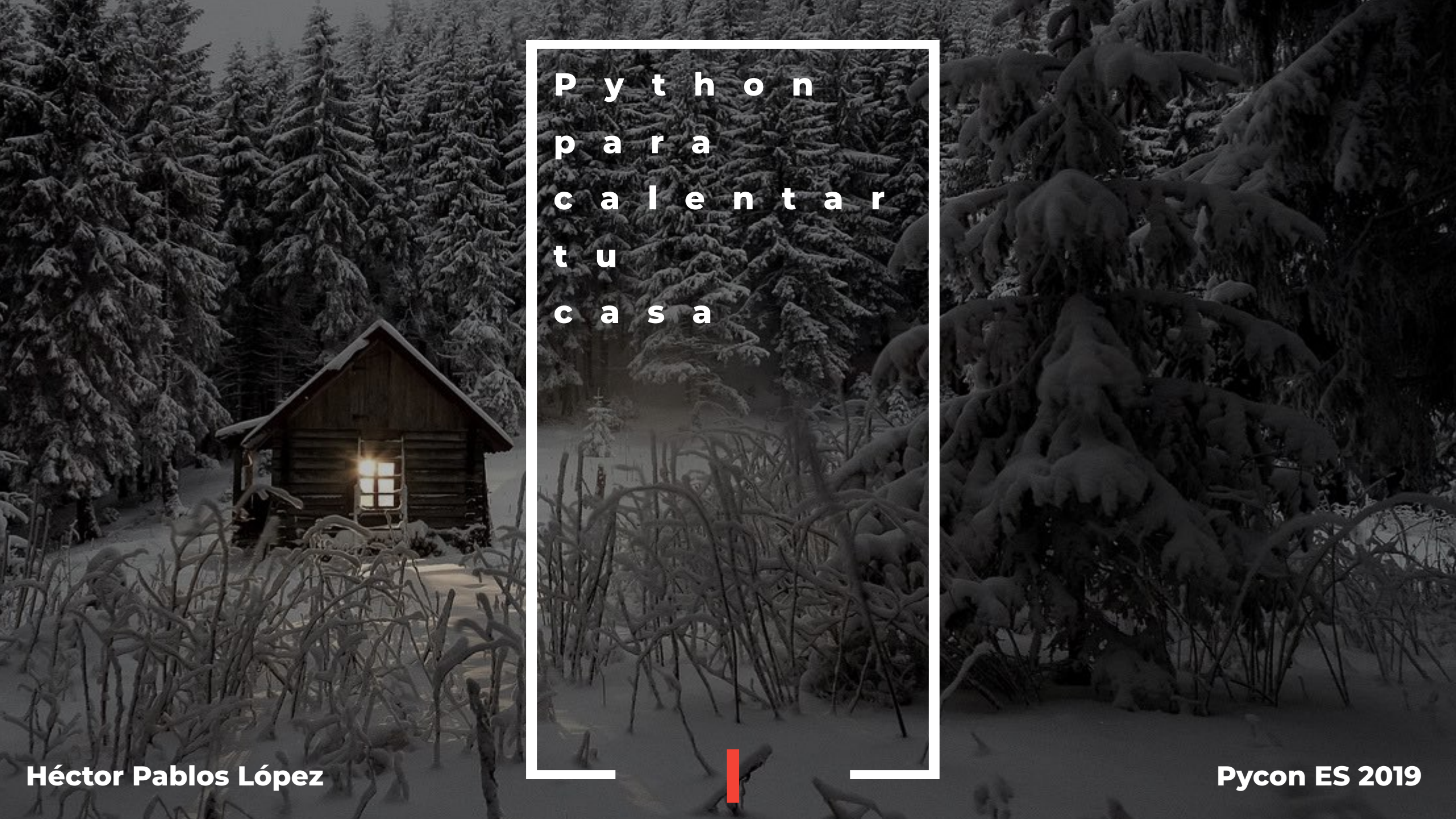
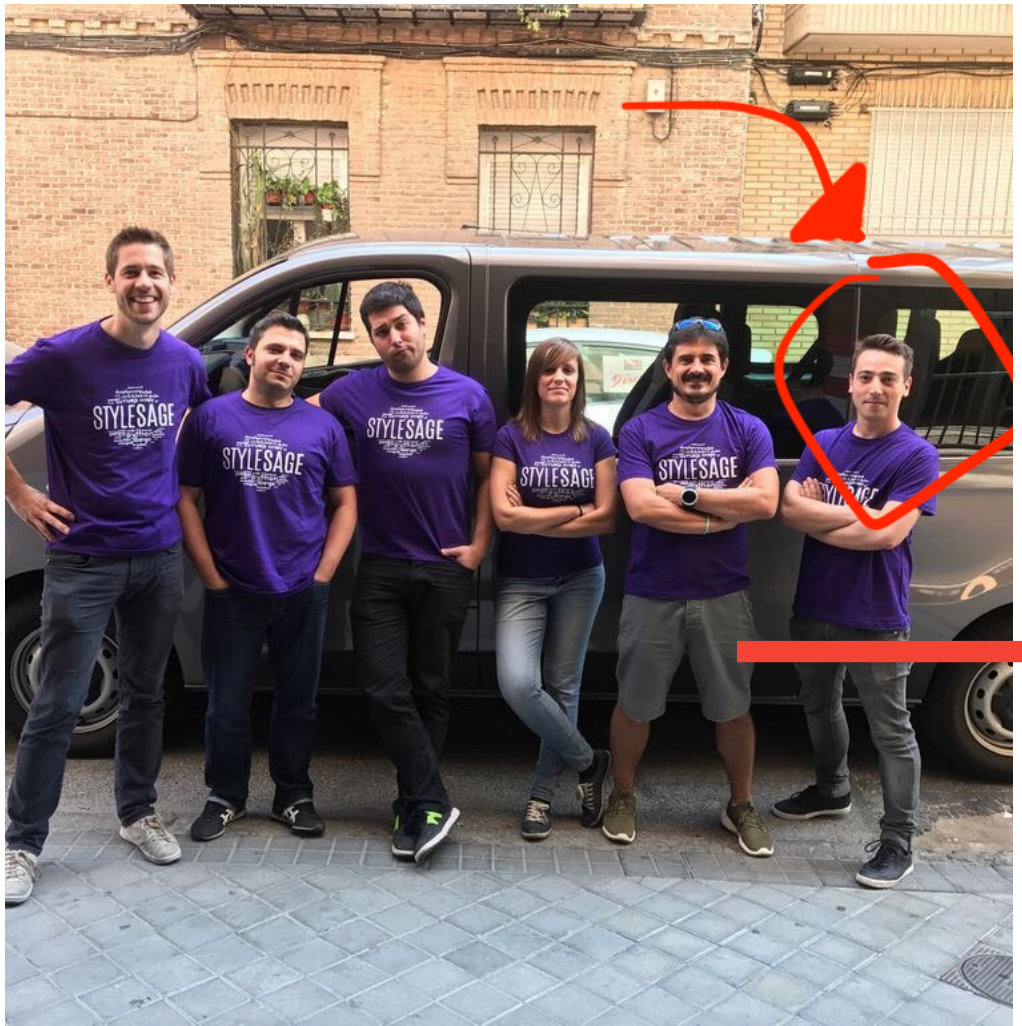


Héctor Pablos López

Pycon ES 2019



We're HIRING

Héctor Pablos López

Ingeniero de Software

STYLESAGE

https://stylesage.co/

hector@stylesage.co

Índice de contenidos

03

#Tres

Ventajas e inconvenientes

Qué nos perdemos por no comprar un termostato comercial y qué ganamos

04

#Cuatro

Preguntas

Termostatos Inteligentes

01

Multitud de alternativas comerciales

Google Nest Learning, Google Nest E·,
Tado, Somfy, Honeywell, Netatmo, Toon...

Entre 150€ y 250€

Acceso remoto

Interactuar con el termostato desde cualquier lugar

Encendido programable

Fijar reglas para apagar y encender la calefacción en días y horas concretos

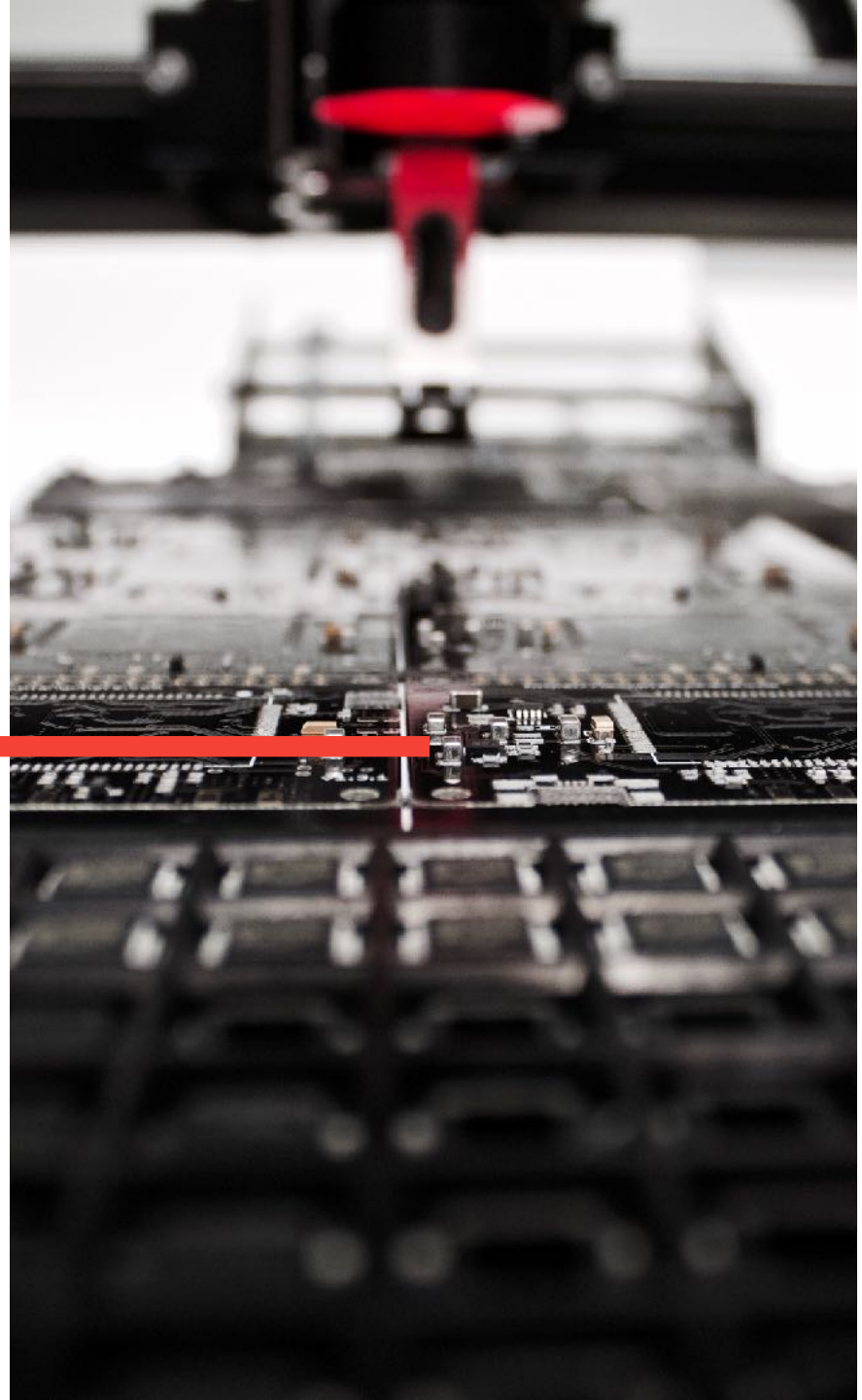
Ahorro de energía

Optimización del tiempo de encendido de la caldera para disminuir el gasto de energía

Visualización de datos

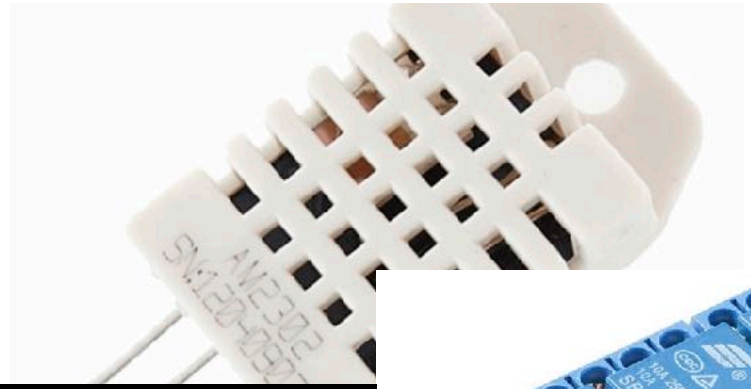
Histórico de temperaturas, tiempo de encendido...

Creación de Nuestro Termostato



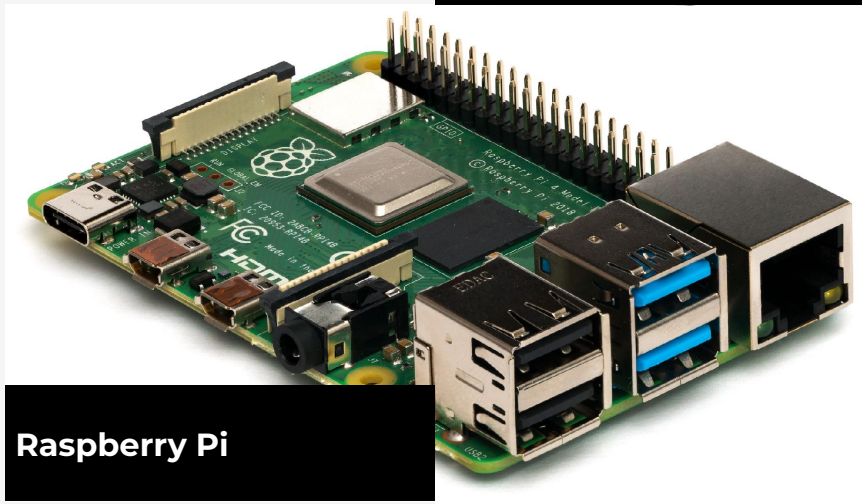
02

Partes de Nuestro Sistema

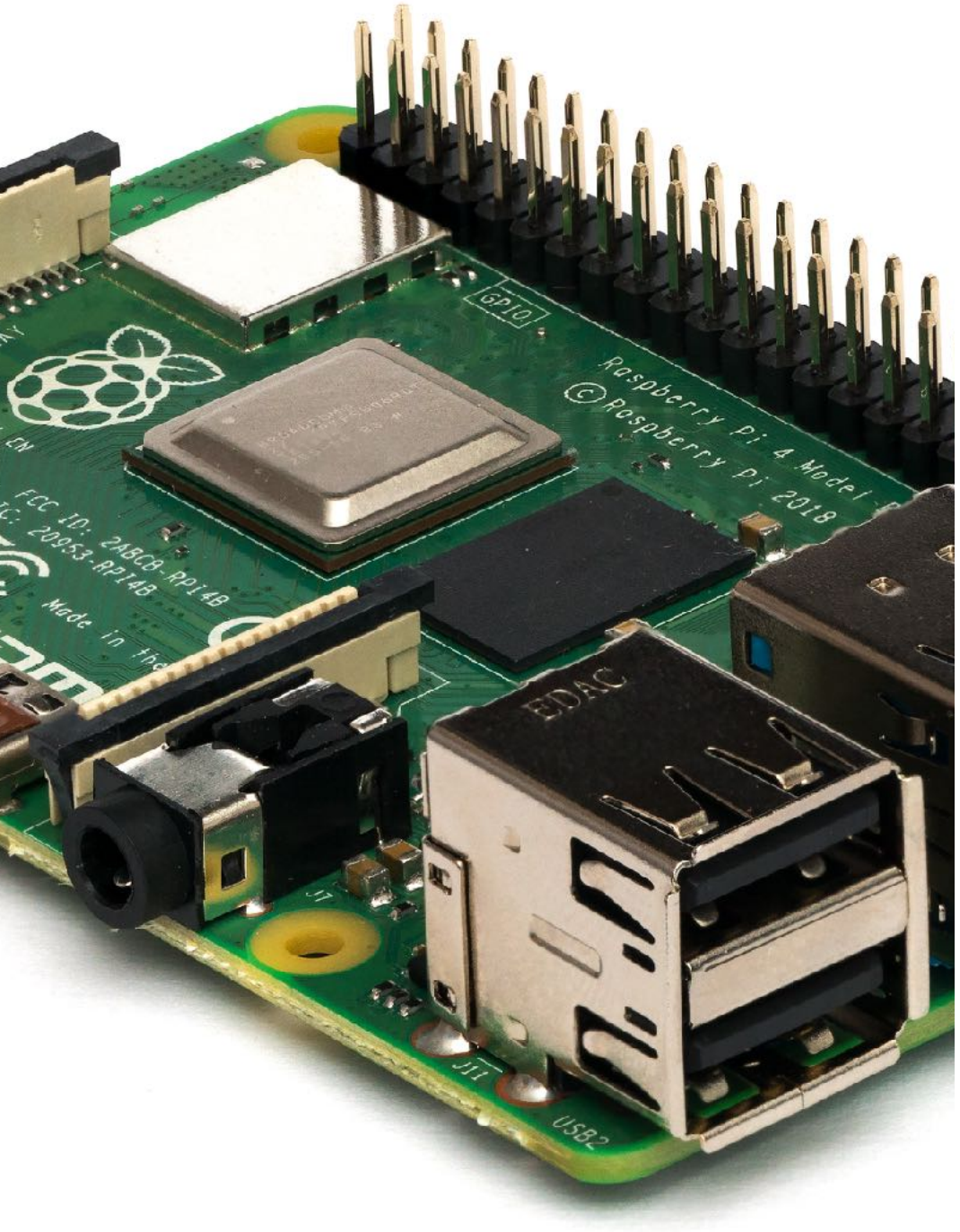


Medición temperatura
Sensor DHT22

Control de la caldera
ELEGOO Módulos de Relés



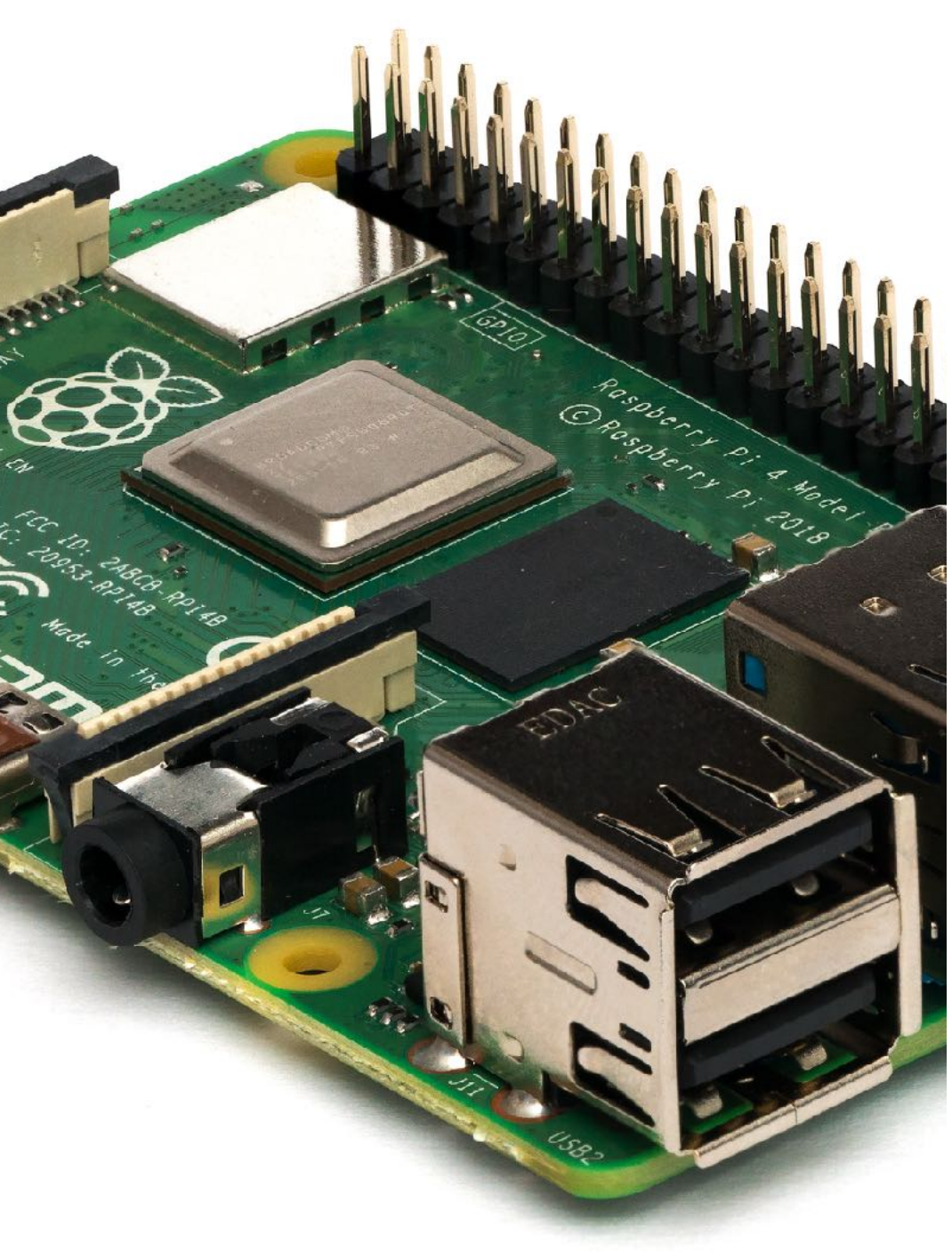
Raspberry Pi



Recomendaciones Raspberry Pi

- Conectada a internet
- Acceso por SSH
- IP fija dentro de la red local
- Instalar paquetes de desarrollo... Python!

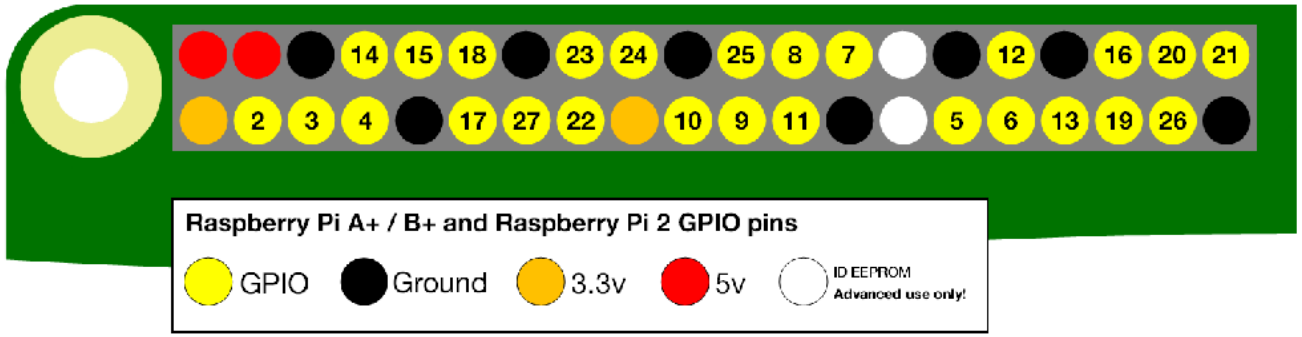
```
sudo apt-get install python3-dev python3-pip
```



Recomendaciones Raspberry Pi

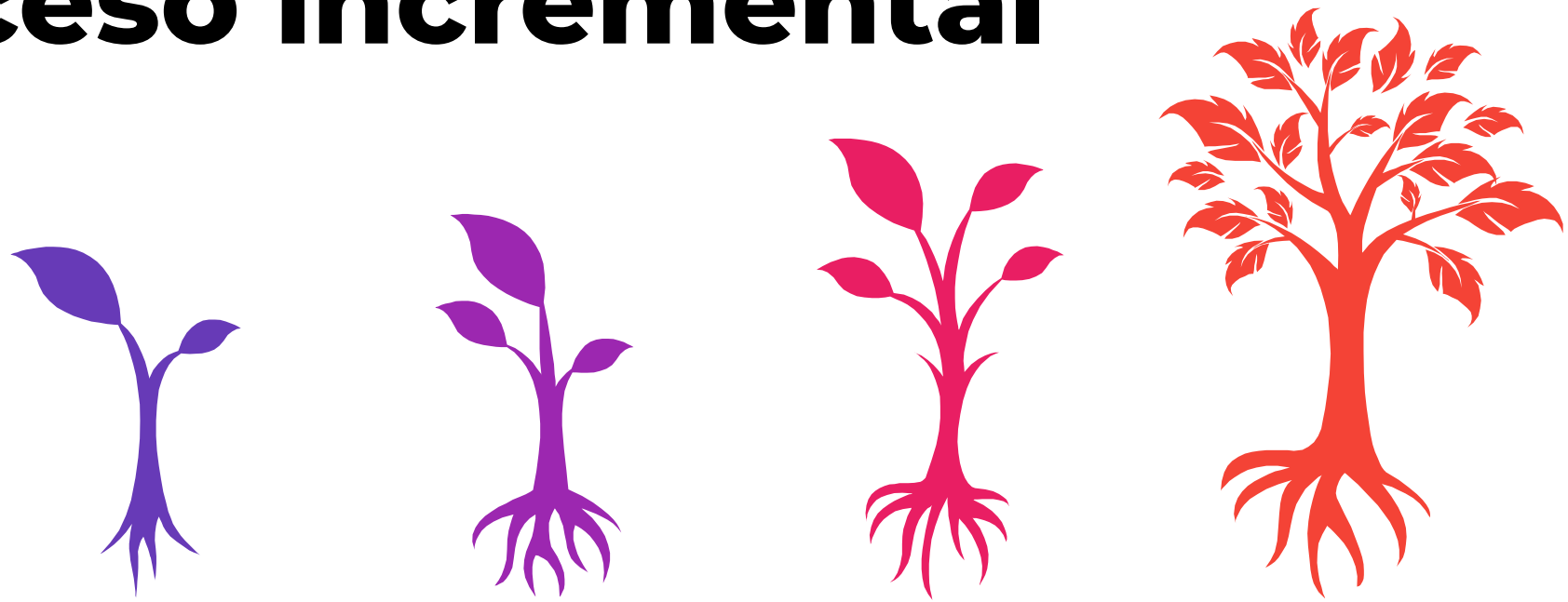
Conoce los pines

```
# pinout
```



Fuente: https://www.raspberrypi.org/documentation/usage/gpio/

Proceso incremental



01 - Medir Temperatura

Conectar nuestra raspberry a un sensor de temperatura y leer las mediciones

02 - Almacenar Mediciones

Guardar las mediciones de temperatura en un almacenamiento persistente

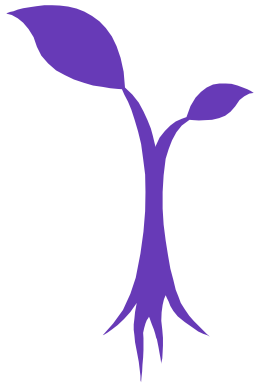
03 - Encender y apagar calefacción

Cómo interactuar con la caldera

04 - Acceso remoto y termostato

Proporcionamos la funcionalidad controlar la calefacción de forma remota

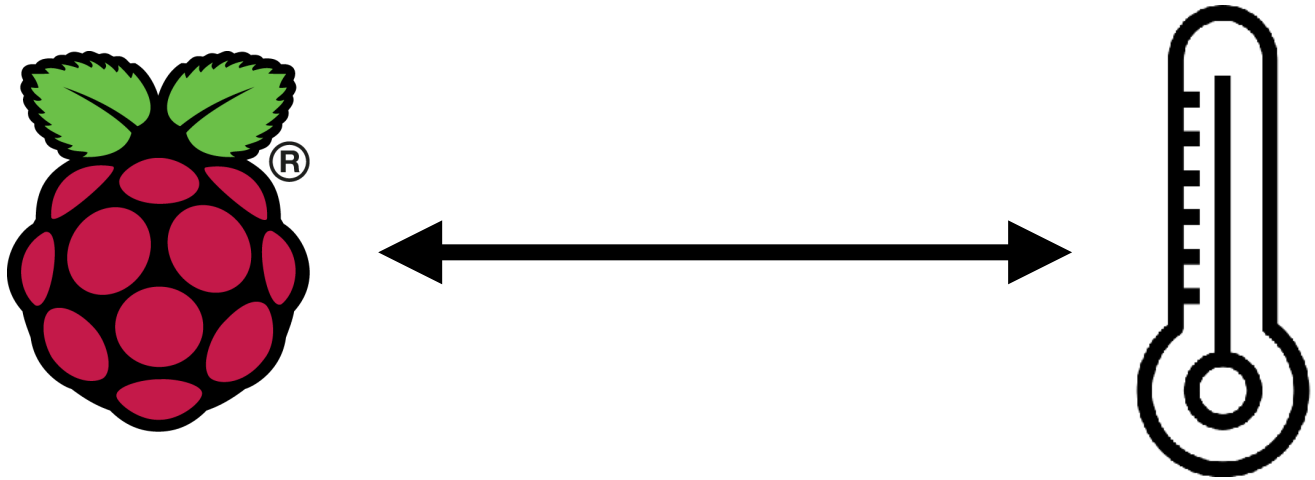
Proceso incremental



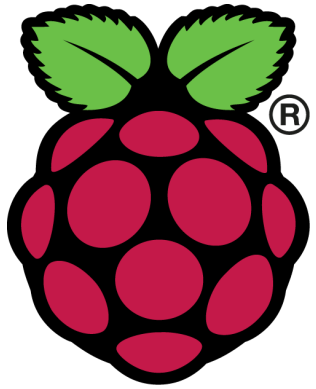
01 - Medir Temperatura

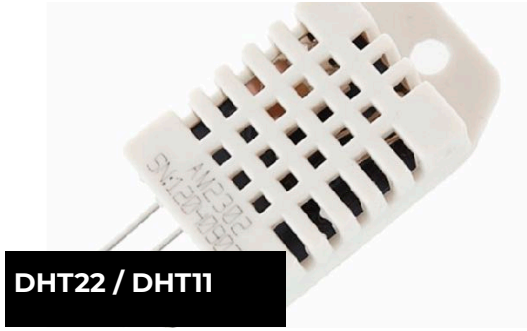
Conectar nuestra raspberry a un sensor de temperatura y leer las mediciones

Medición de temperatura

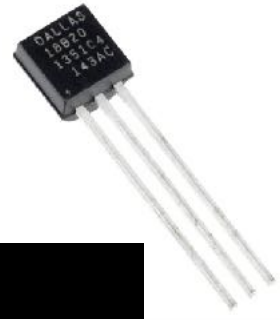


Medición de temperatura



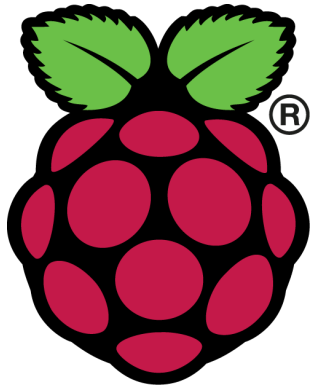


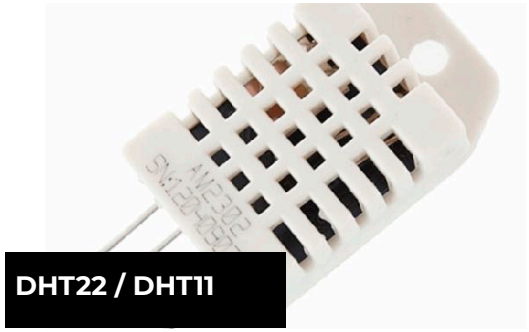
DHT22 / DHT11



DS18B20

Medición de temperatura

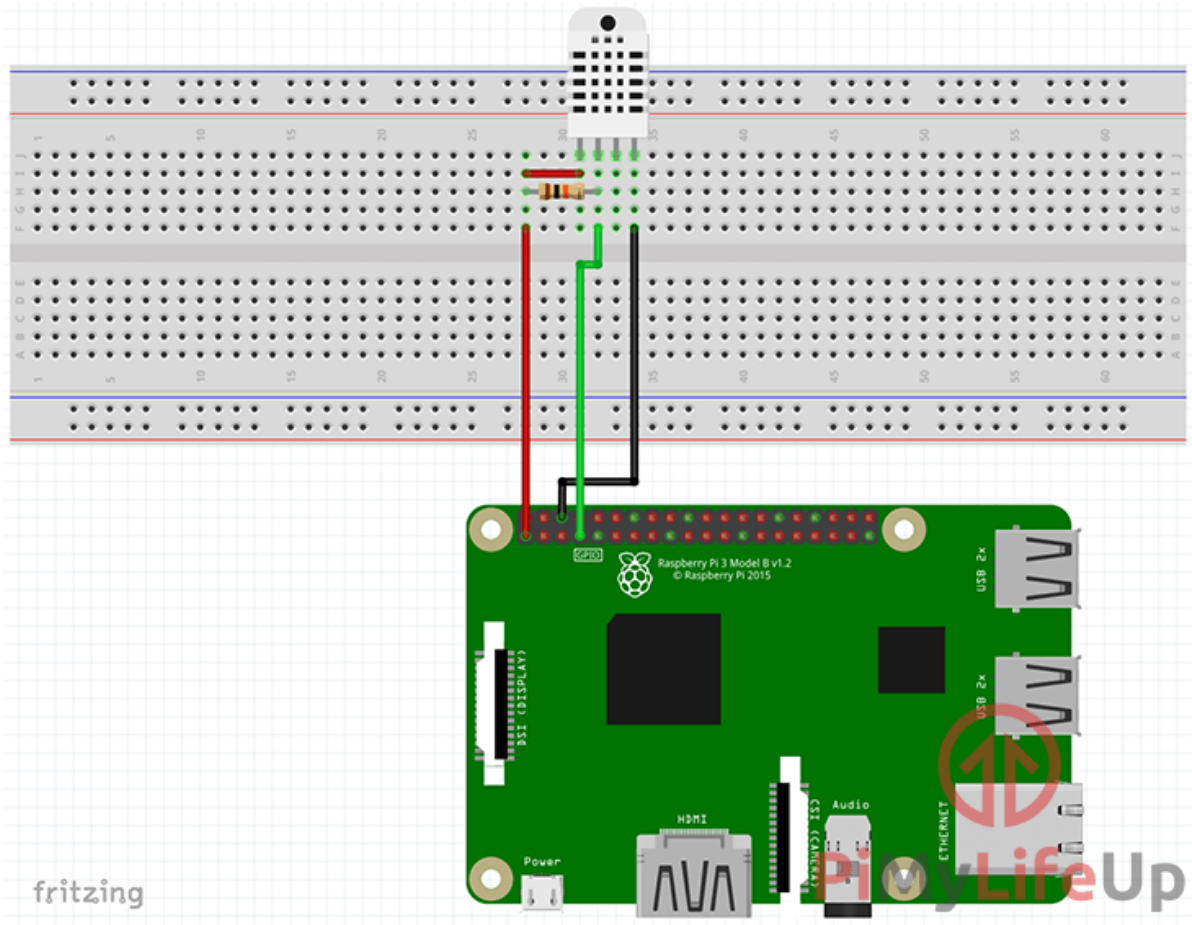




DHT22 / DHT11

Medición de temperatura

Modo cableado



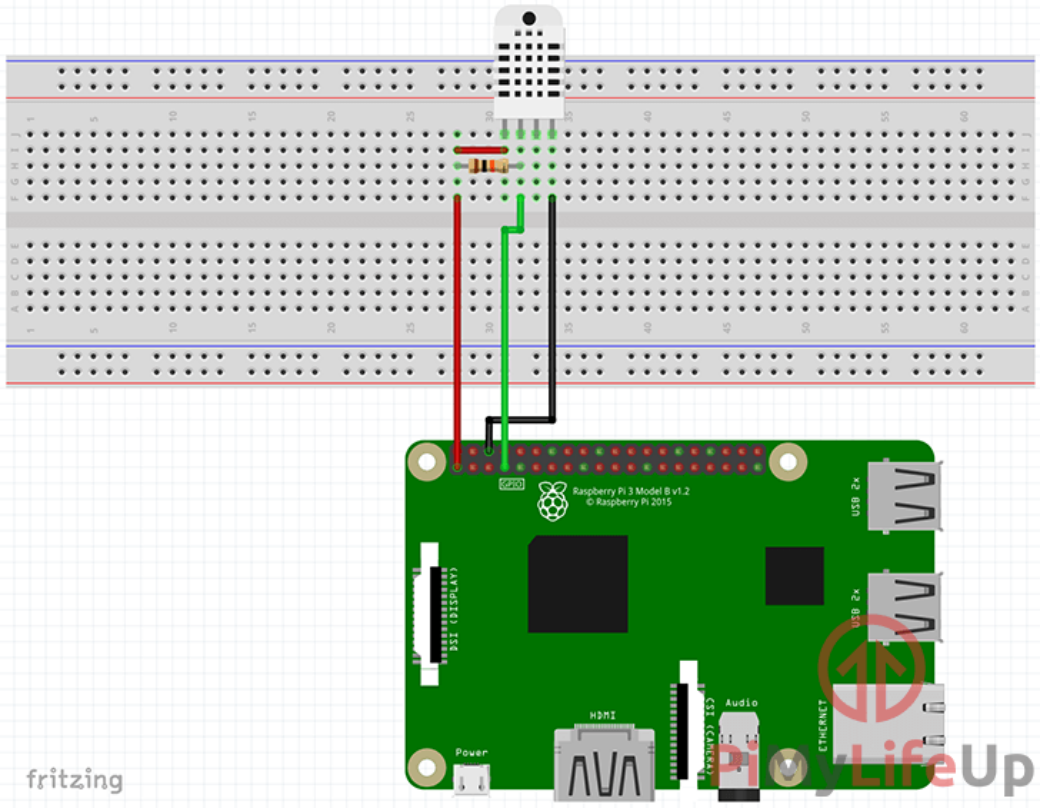
Fuente: https://pimylifeup.com/raspberry-pi-humidity-sensor-dht22/

Medición de temperatura

Modo cableado

- Sensor DHT22
- Placa de conexión
- Resistencia de 10kΩ
- Cables jumper

Alrededor de 20€



Fuente: https://pimylifeup.com/raspberry-pi-humidity-sensor-dht22/

Medición de temperatura

Modo cableado

Adafruit_Python_DHT: https://github.com/adafruit/Adafruit_Python_DHT

```
# pipenv install Adafruit_DHT
```

```
SENSOR_OUT_PIN = 2
humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT22, SENSOR_OUT_PIN)
```

Fuente: https://pimylifeup.com/raspberry-pi-humidity-sensor-dht22/

Medición de temperatura

Modo cableado

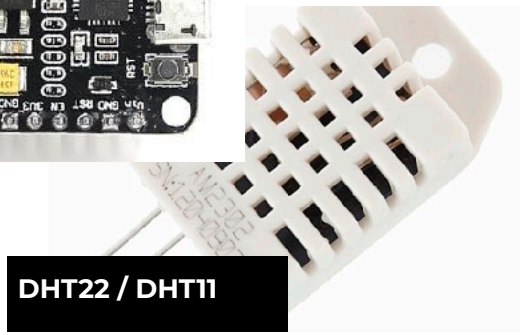
```
from time import sleep

import Adafruit_DHT


if __name__ == '__main__':

    while True:
        humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT22, 2)

        if humidity is not None and temperature is not None:
            print("Temp={0:0.1f}*C  Humidity={1:0.1f}%".format(temperature, humidity))
        else:
            print('Failed to get reading. Trying again after poll time.')

        sleep(20)
```

Fuente: https://pimylifeup.com/raspberry-pi-humidity-sensor-dht22/

Medición de temperatura

Modo wireless

ESP8266

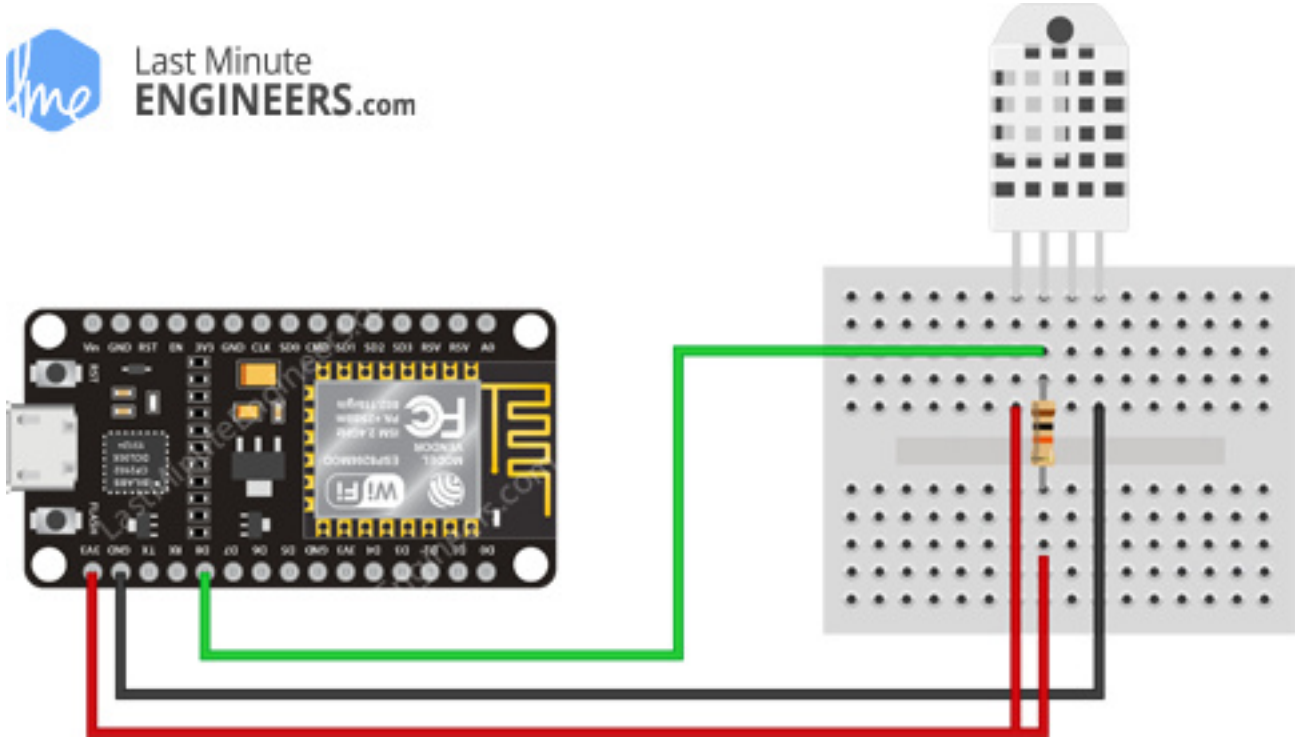


DHT22 / DHT11

Fuente: https://lastminuteengineers.com/esp8266-dht11-dht22-web-server-tutorial/

Medición de temperatura

Modo wireless



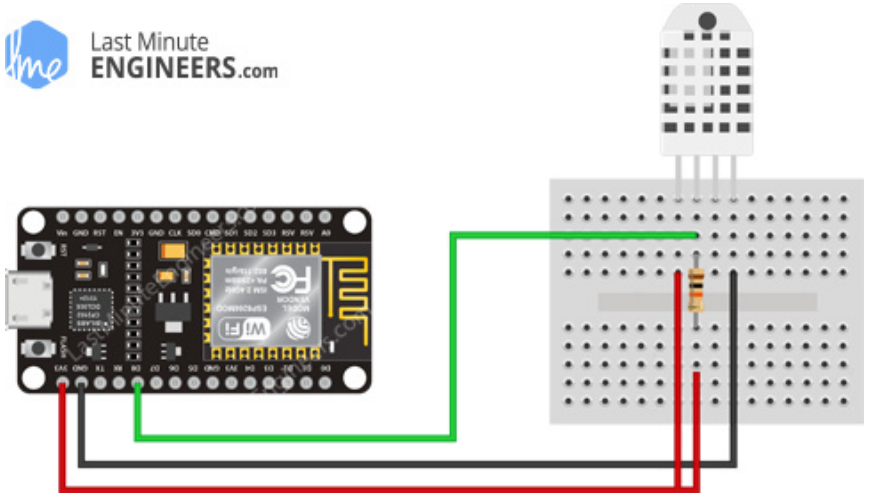
Fuente: https://lastminuteengineers.com/esp8266-dht11-dht22-web-server-tutorial/

Medición de temperatura

Modo wireless

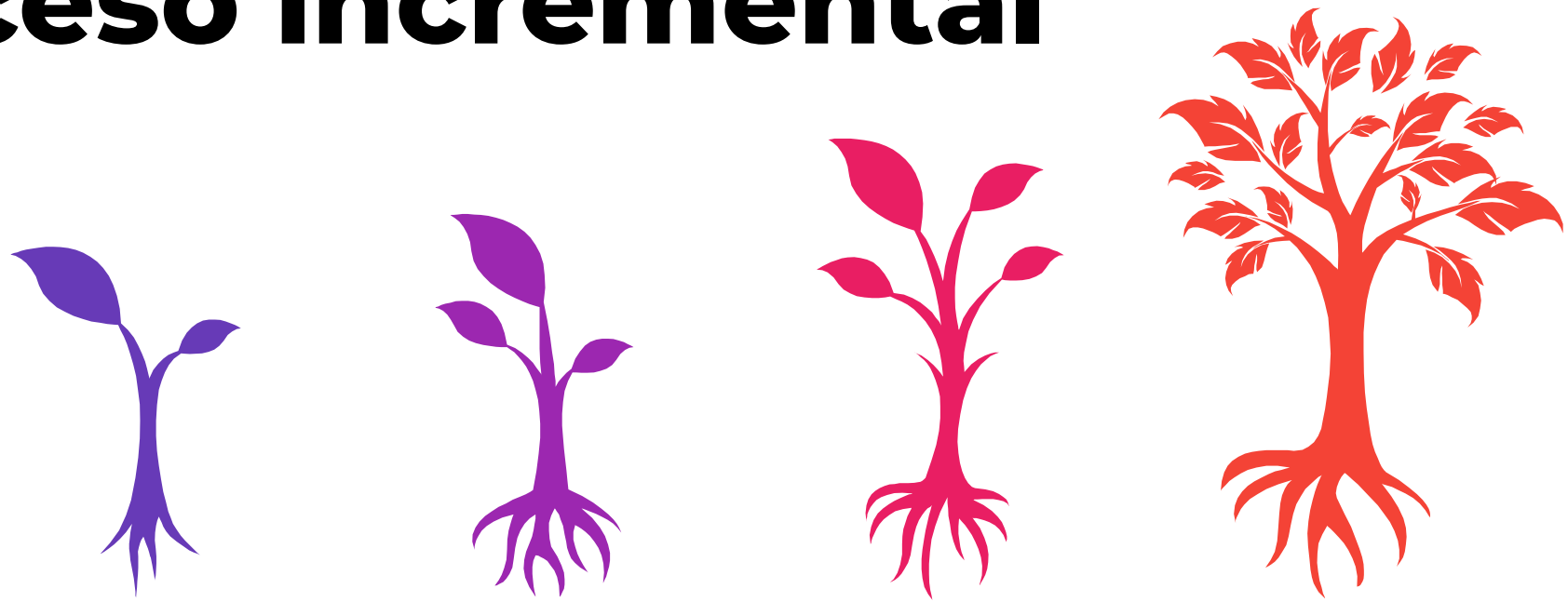
- Sensor DHT22
- Placa de conexión
- Resistencia de 10kΩ
- Cables jumper
- **ESP8266 NodeMCU**
- **Fuente de alimentación**

Alrededor de 35€



Fuente: https://lastminuteengineers.com/esp8266-dht11-dht22-web-server-tutorial/

Proceso incremental



01 - Medir Temperatura

Conectar nuestra raspberry a un sensor de temperatura y leer las mediciones

02 - Almacenar Mediciones

Guardar las mediciones de temperatura en un almacenamiento persistente

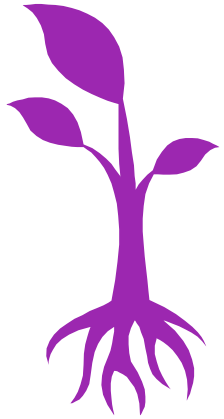
03 - Encender y apagar calefacción

Cómo interactuar con la caldera

04 - Acceso remoto y termostato

Proporcionamos la funcionalidad controlar la calefacción de forma remota

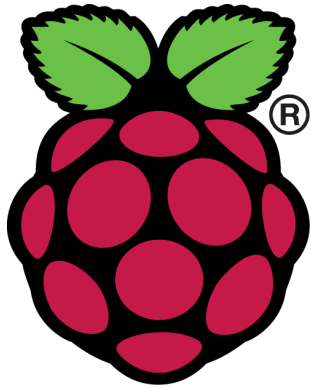
Proceso incremental

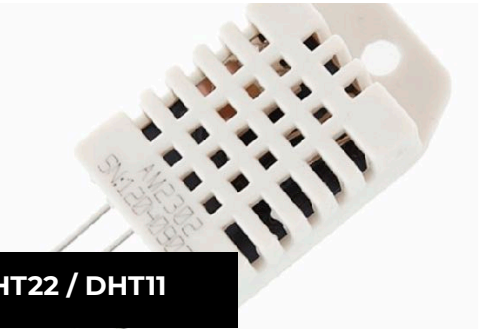


02 - Almacenar Mediciones

Guardar las mediciones de temperatura en un almacenamiento persistente

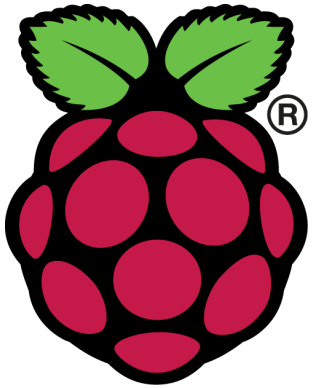
Almacenamiento de temperatura

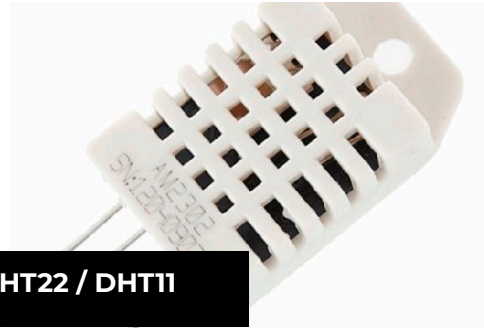




DHT22 / DHT11

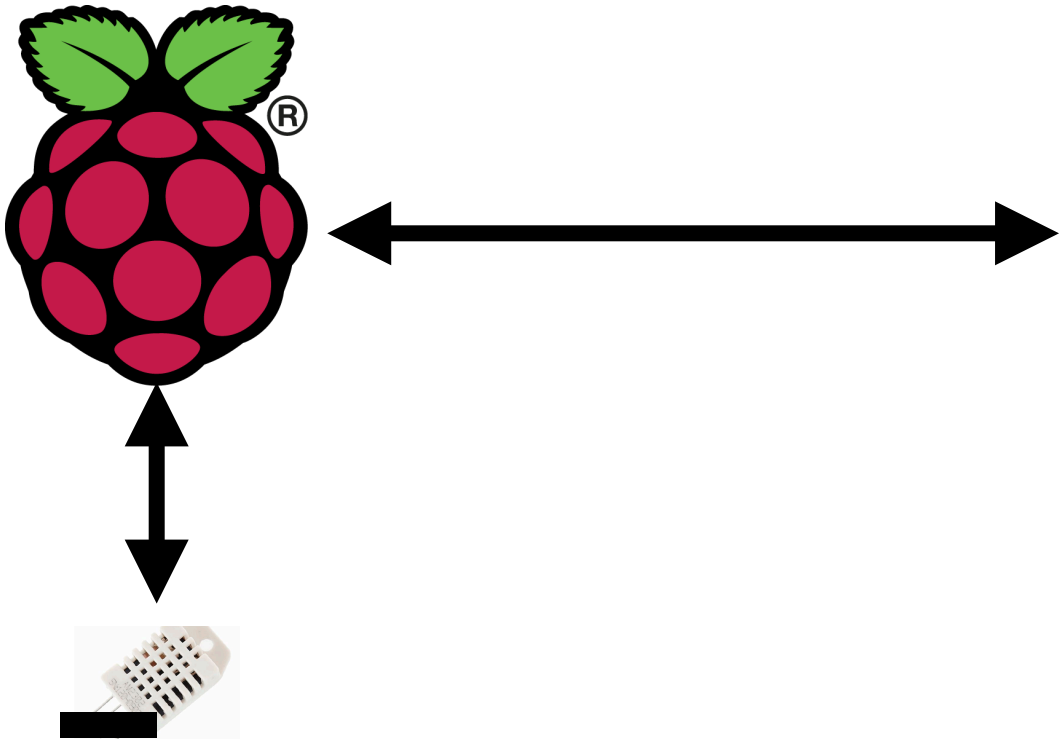
Almacenamiento de temperatura

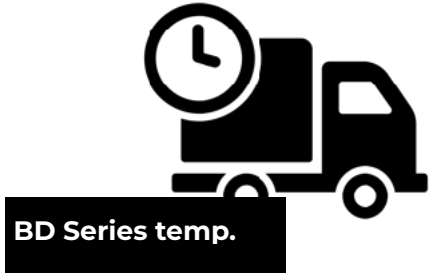




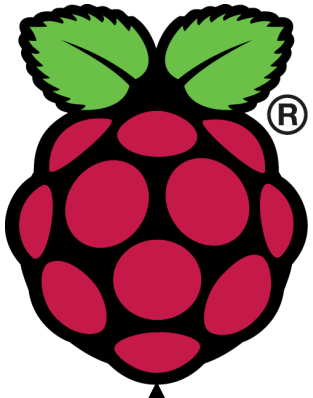
DHT22 / DHT11

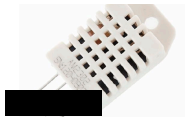
Almacenamiento de temperatura



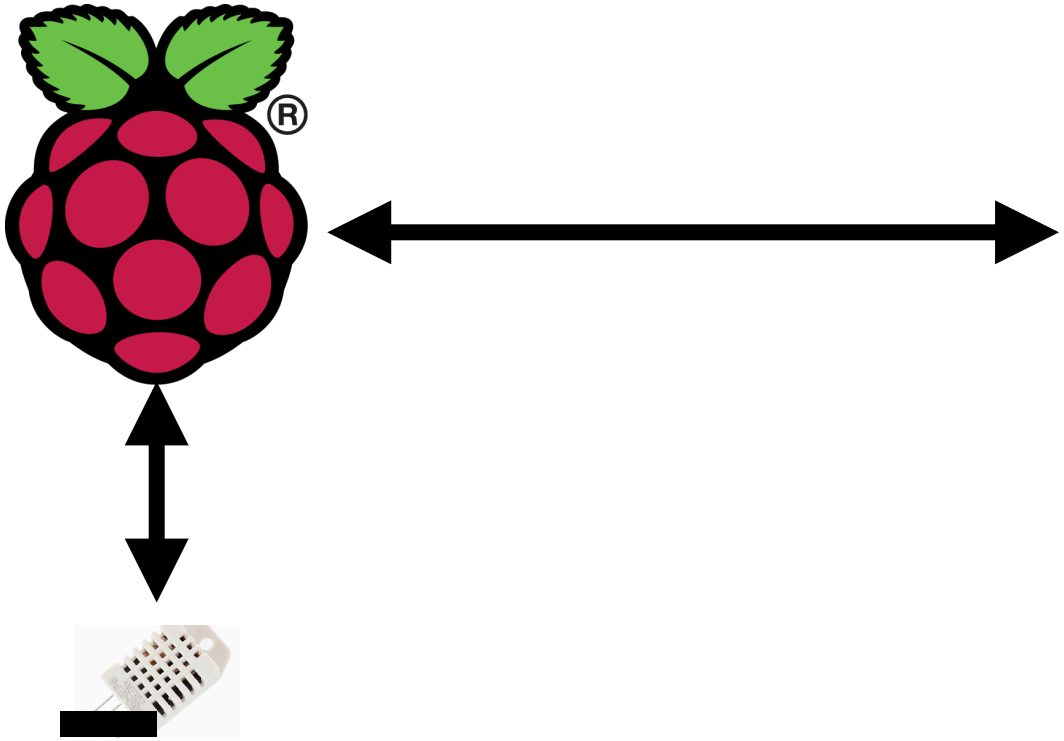


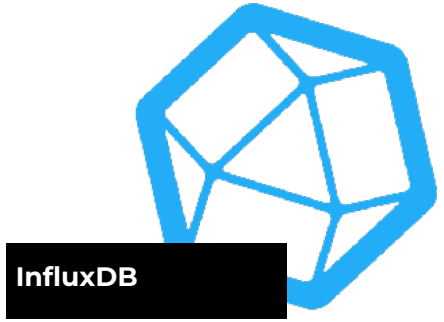
Almacenamiento de temperatura





Almacenamiento de temperatura





Almacenamiento de temperatura

Instalación: https://docs.influxdata.com/influxdb/v1.7/introduction/installation/

Influxdb-python: https://github.com/influxdata/influxdb-python

```
# pipenv install influxdb
```

```
from influxdb import InfluxDBClient


client = InfluxDBClient(
    INFLUX_SETTINGS['HOST'],
    INFLUX_SETTINGS['PORT'],
    INFLUX_SETTINGS['USER'],
    INFLUX_SETTINGS['PASSWORD'],
    INFLUX_SETTINGS['DATABASE']
)
```

InfluxDB Docs: https://docs.influxdata.com/influxdb/v1.7/

Almacenamiento de temperatura

```python
from influxdb import InfluxDBClient


client = InfluxDBClient(
    # settings
)

humidity, temperature = Adafruit_DHT.read_retry(settings.DHT_SENSOR, settings.SENSOR_OUT_PIN)
json_body = [
    {
        "measurement": "environment",
        "tags": {
            "room": "living_room"
        },
        "fields": {
            "temperature": temperature,
            "humidity": humidity
        }
    }
]
client.write_points(json_body)
```

InfluxDB Docs: https://docs.influxdata.com/influxdb/v1.7/

Almacenamiento de temperatura

Ejemplo detallado

GitLab

Repositorio git: https://gitlab.com/hctpbl/dht-sensor-measurement-storage

Almacenamiento de temperatura

Supervisor

/etc/supervisor/conf.d/measure_temp.conf

```
[program:store_temp]
directory=/home/hector/projects/temp_monitor
command=/home/hector/.local/bin/pipenv run python store_temp.py
autostart=true
autorestart=true
stderr_logfile=/var/log/store_temp.err.log
stdout_logfile=/var/log/store_temp.out.log
user=hector
```

```
# sudo supervisorctl start store_temp
```

Almacenamiento de temperatura

Configuración adicional

Downsampling y retención de datos

- Granularidad de datos alta para mediciones recientes
- Menor granularidad a mayor antigüedad de datos
- Menor espacio ocupado en disco

https://docs.influxdata.com/influxdb/v1.7/guides/downsampling_and_retention/

Almacenamiento de temperatura

Configuración adicional

Visualización

chronograf

Chronograf: https://docs.influxdata.com/chronograf/v1.7/

Grafana

Grafana: https://grafana.com/docs/

Almacenamiento de temperatura

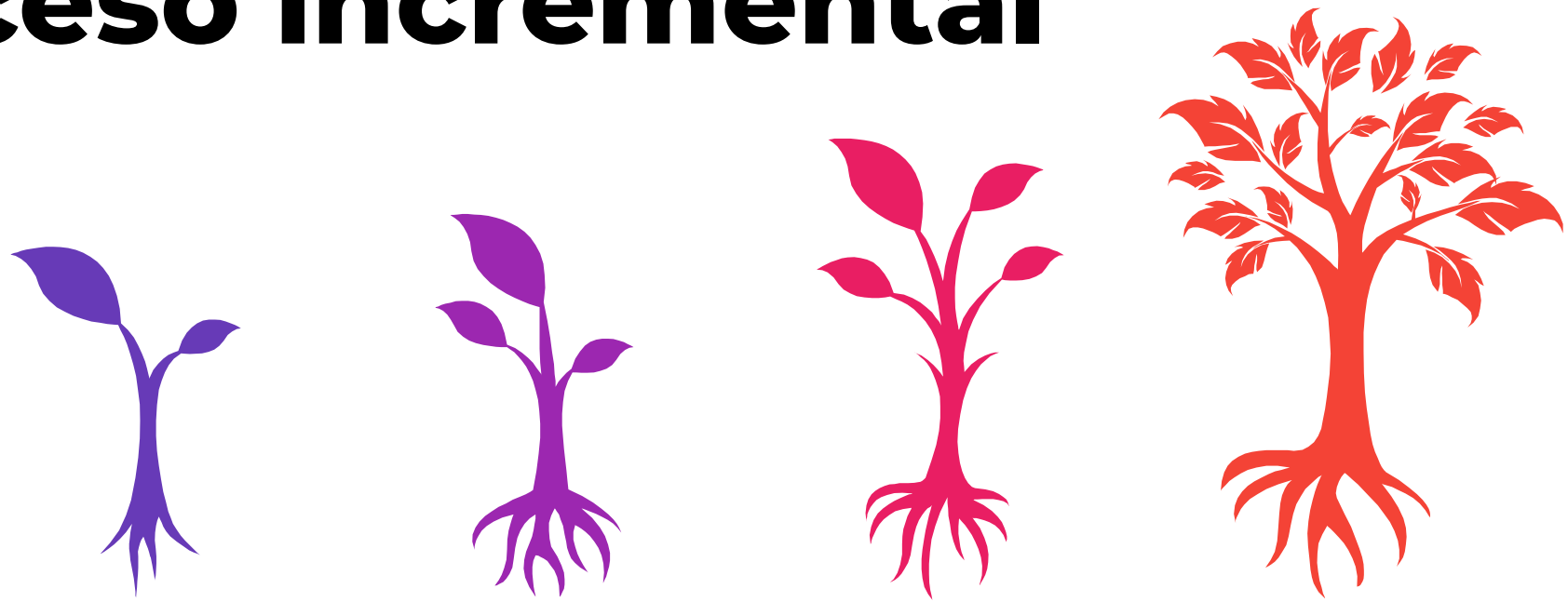
Configuración adicional

Alertas

telegraf

Telegraf: https://docs.influxdata.com/telegraf/v1.12/

Proceso incremental



01 - Medir Temperatura

Conectar nuestra raspberry a un sensor de temperatura y leer las mediciones

02 - Almacenar Mediciones

Guardar las mediciones de temperatura en un almacenamiento persistente

03 - Encender y apagar calefacción

Cómo interactuar con la caldera

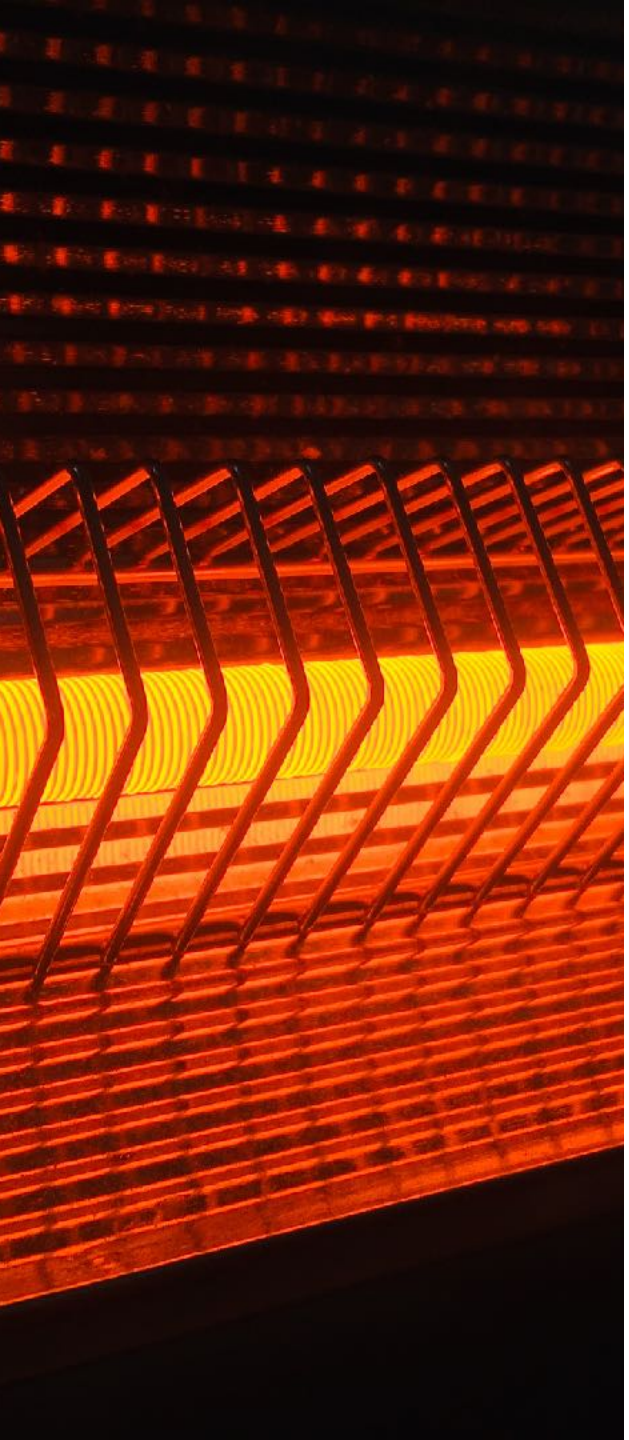
04 - Acceso remoto y termostato

Proporcionamos la funcionalidad controlar la calefacción de forma remota

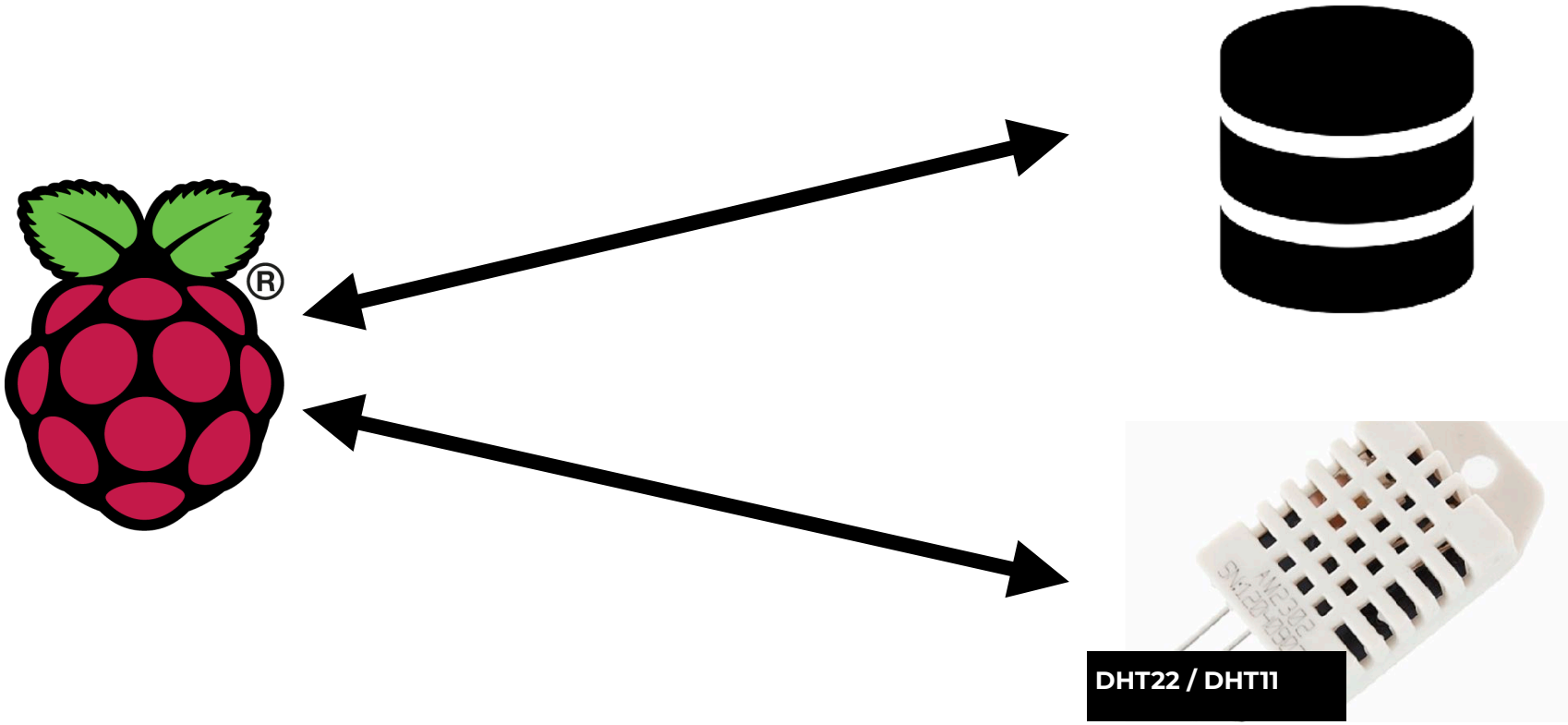
Proceso incremental

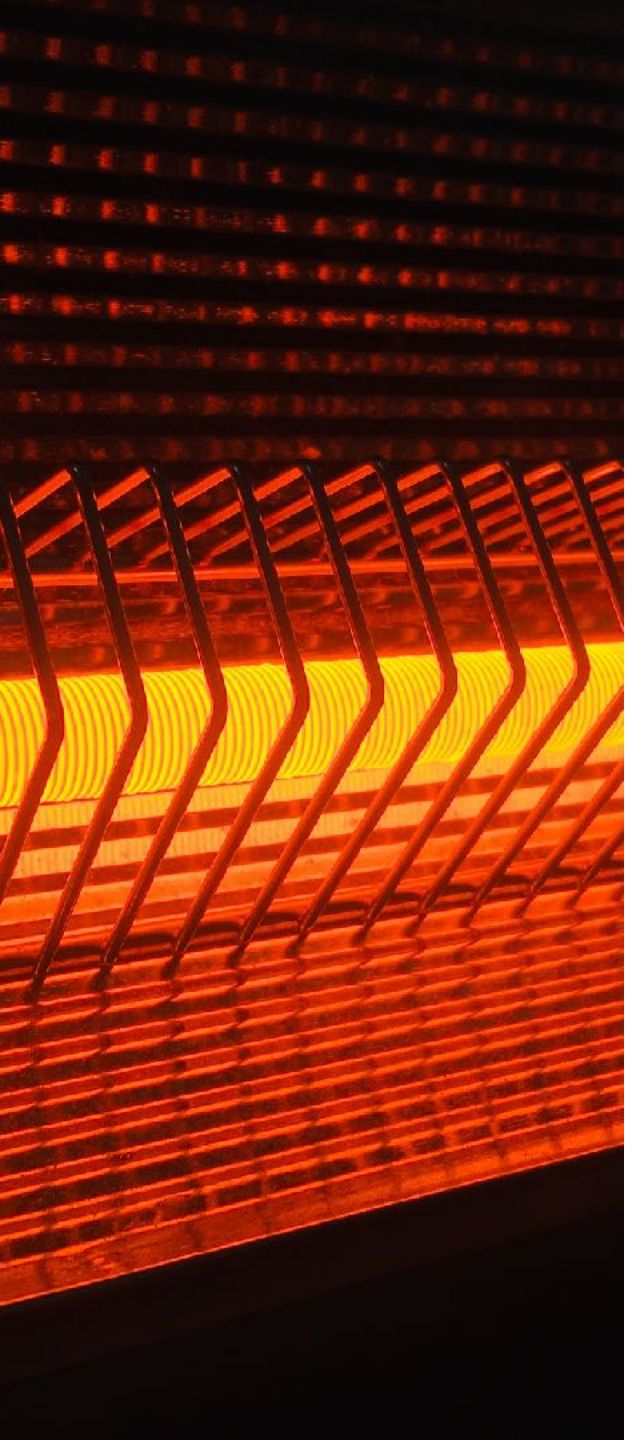
03 - Encender y apagar calefacción

Cómo interactuar con la caldera

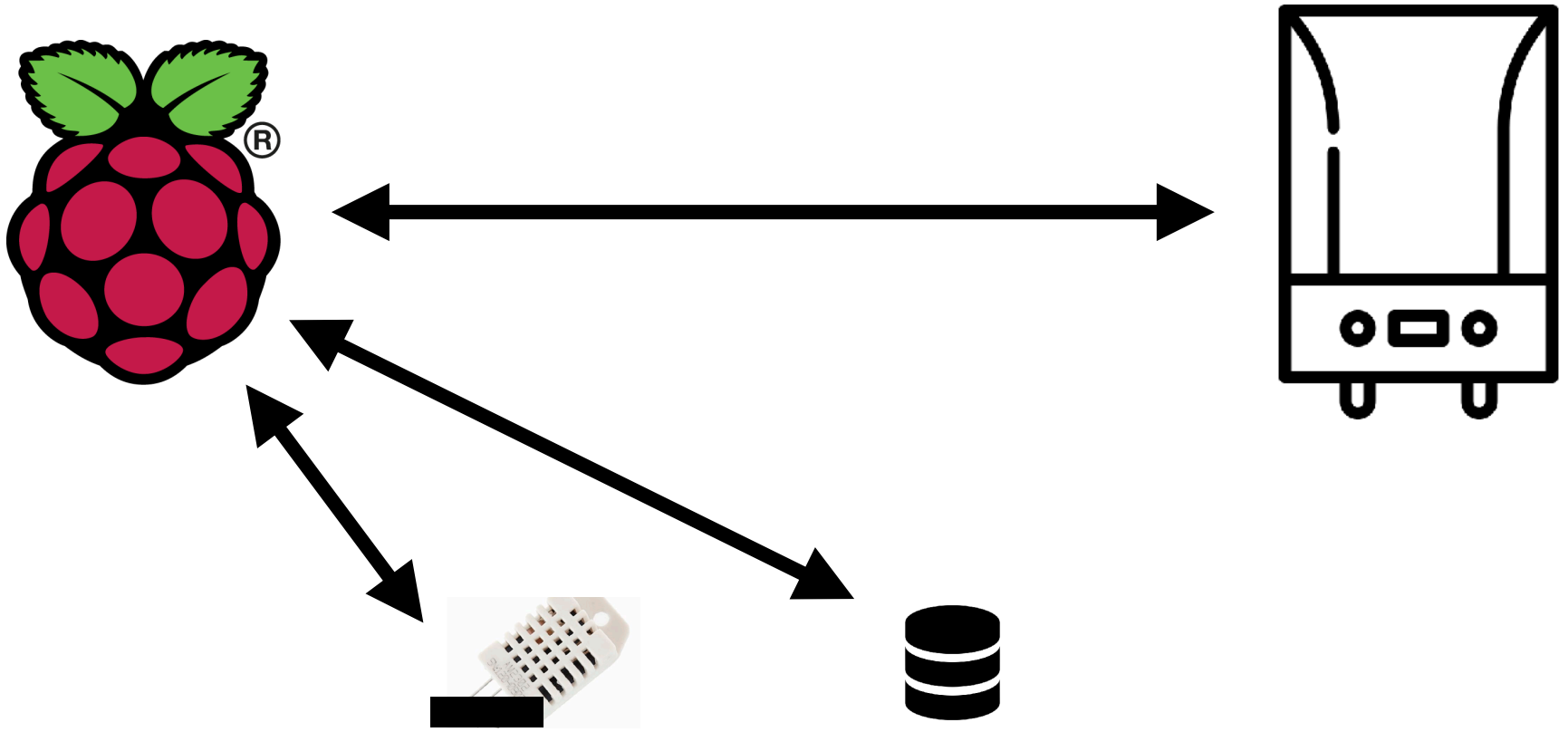


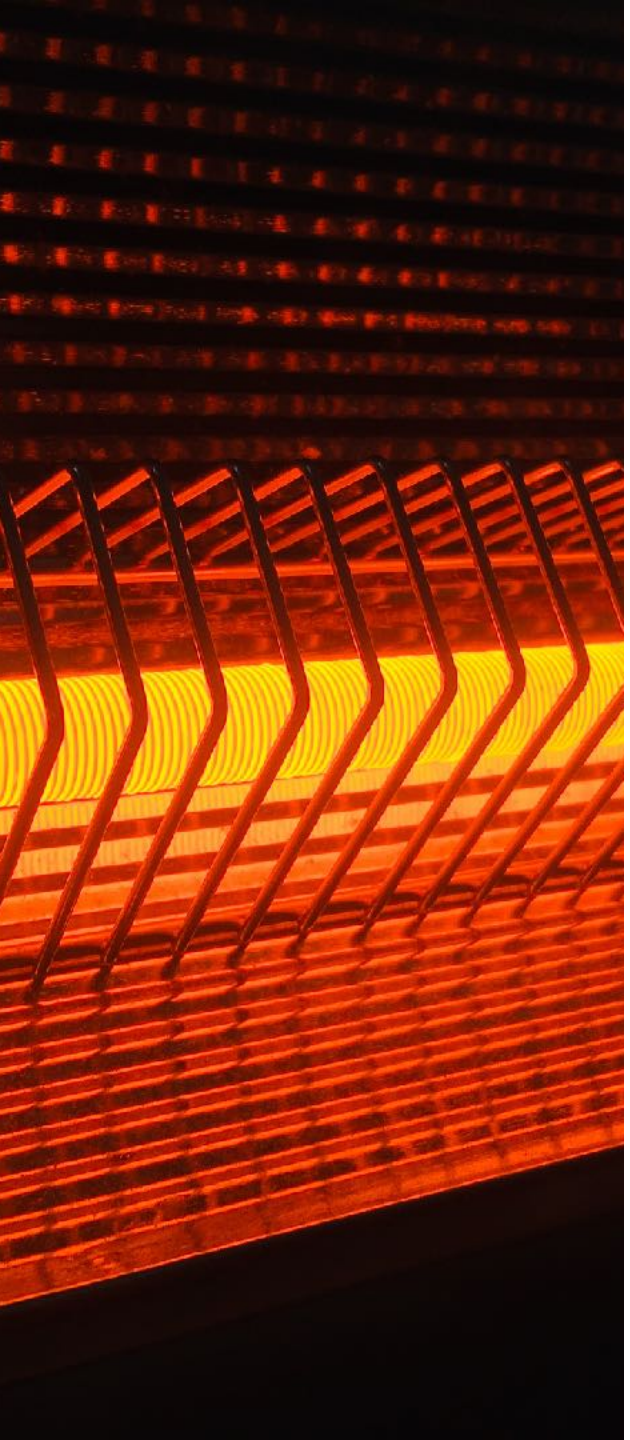
Control de la caldera



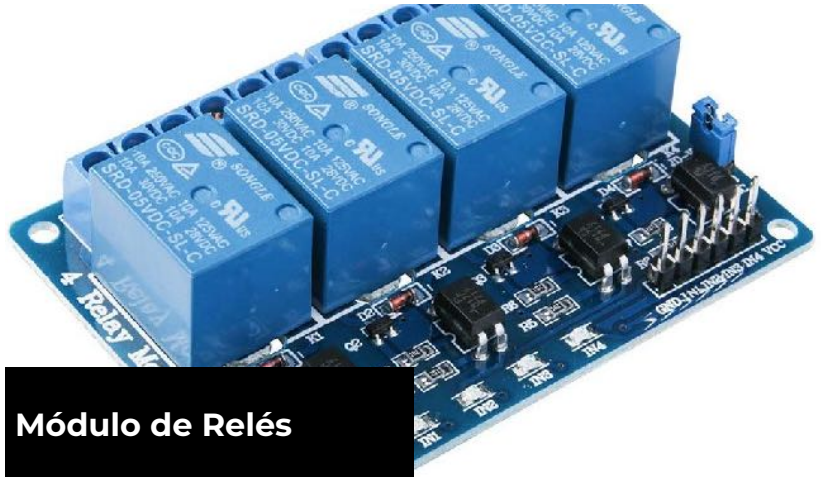


Control de la caldera

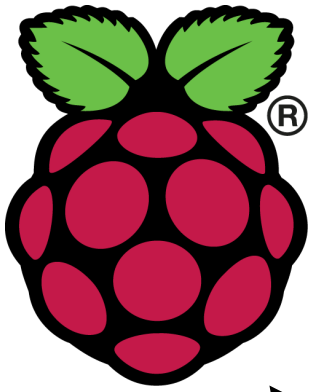


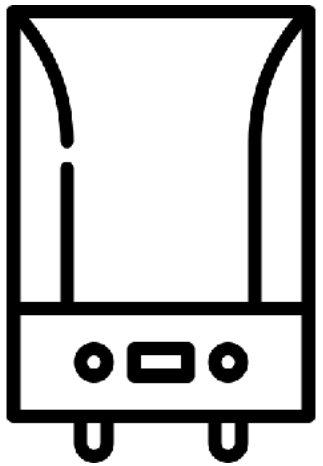


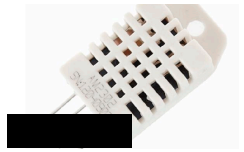
Control de la caldera

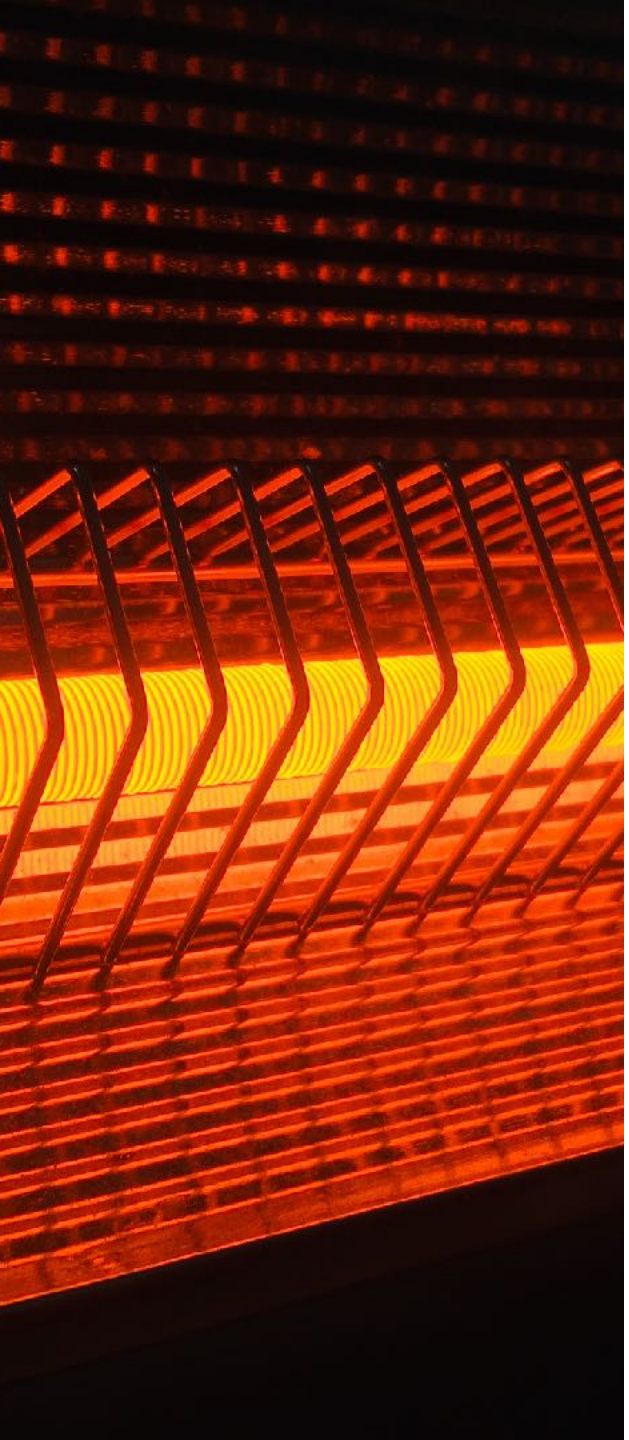


Módulo de Relés









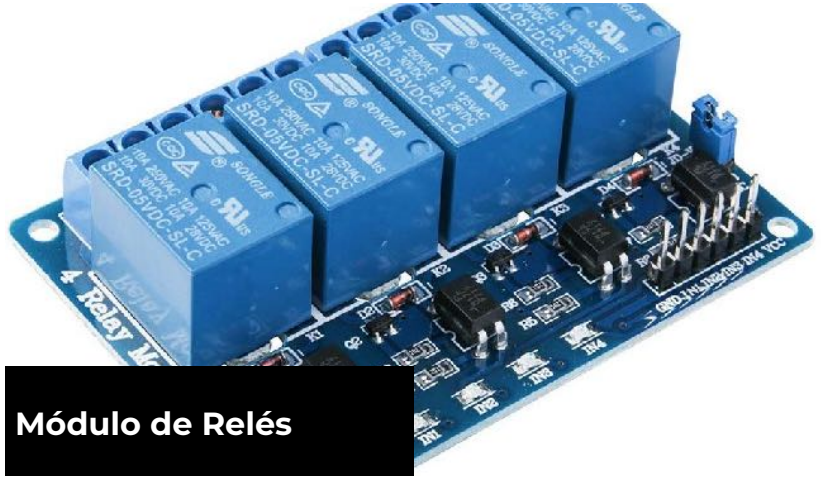
Control de la caldera

¿Qué es un relé?

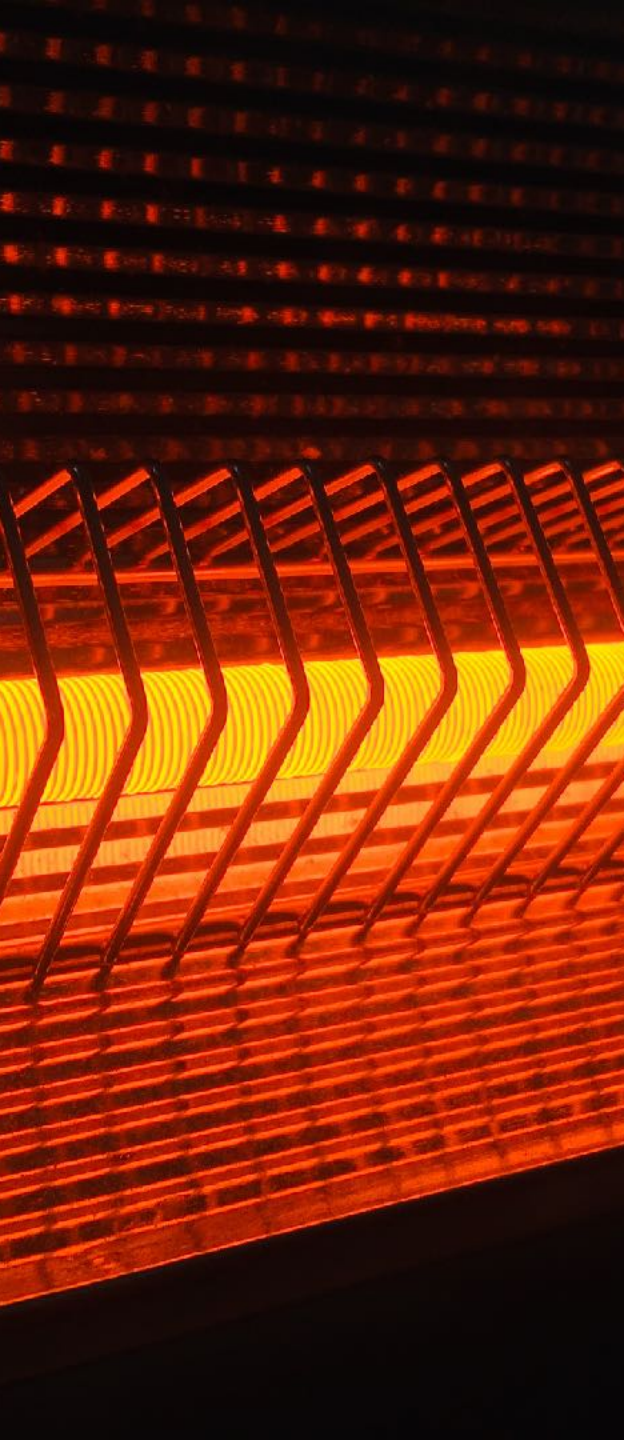
Un “interruptor”

- Permite el paso de corriente de alto voltaje si recibe corriente de bajo voltaje.

- La raspberry proporciona o elimina la corriente de bajo voltaje, abriendo y cerrando el paso de corriente de alto voltaje a la caldera

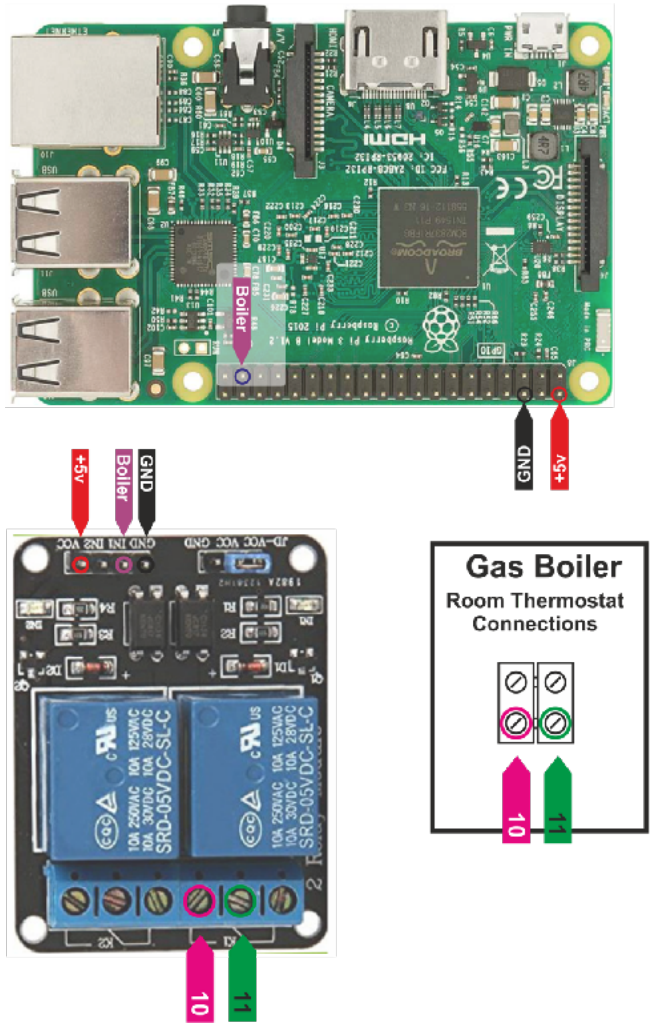


Módulo de Relés

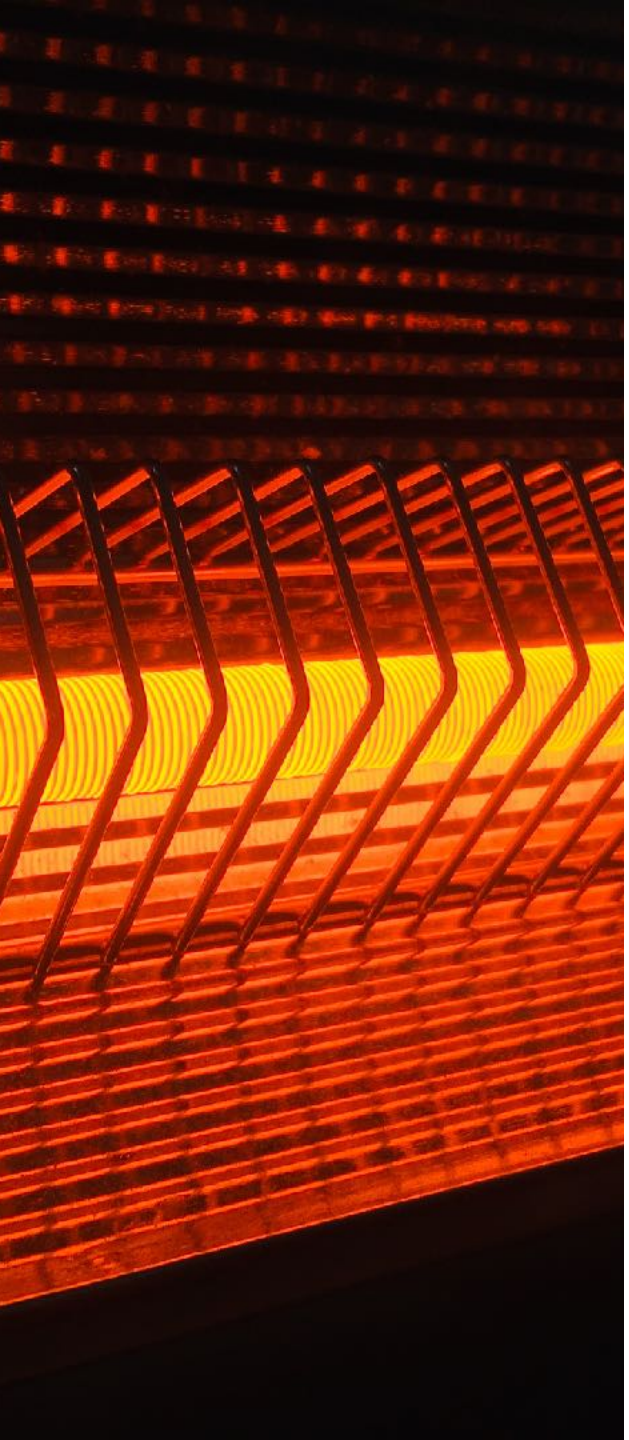


Control de la caldera

Modo cableado



Fuente:
http://www.pihome.eu/2018/03/01/raspberry-pi-heating-boiler-control-system-relay/

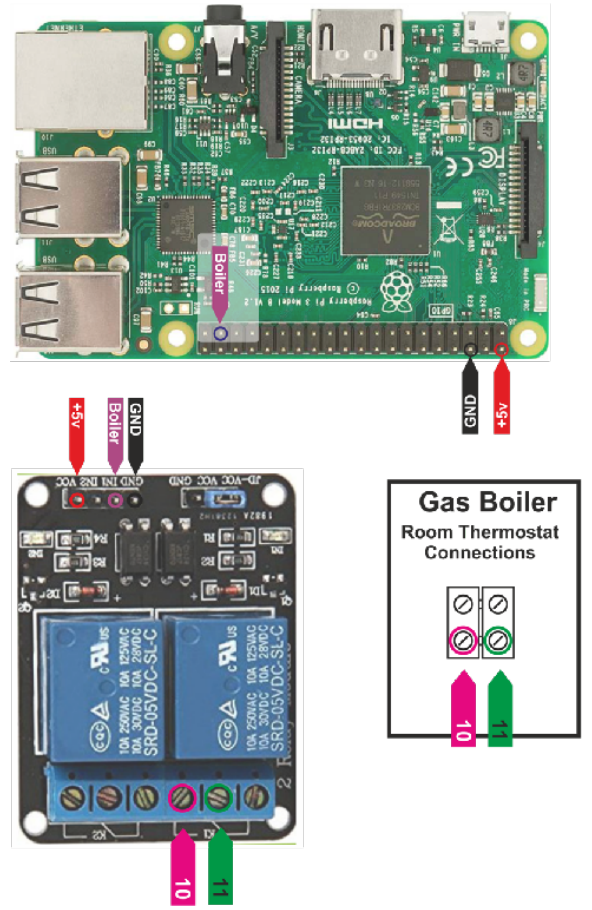


Control de la caldera

Modo cableado

- Relé o módulo de relés
  - Activo a 5V
  - Soportar la potencia de la conexión del termostato a nuestra caldera
- Cables jumper (rPi a relé)
- Cables de corriente (relé a caldera)

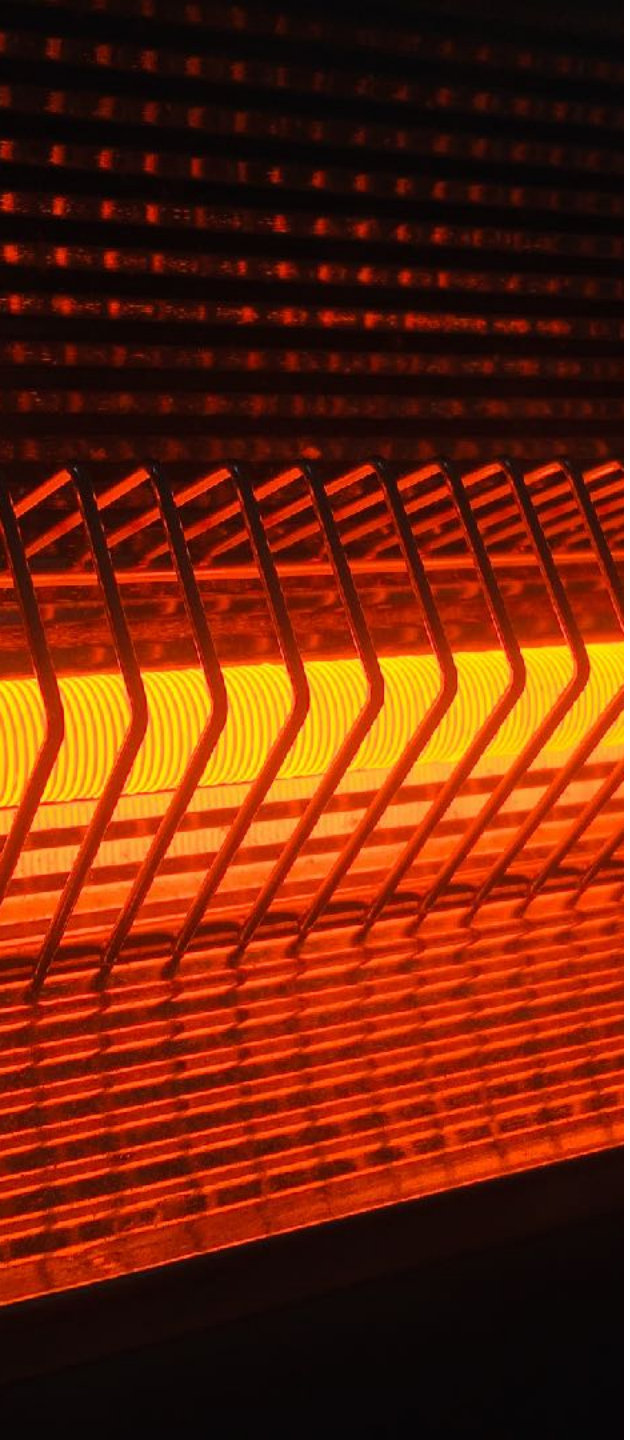
Alrededor de 15€



Ejemplo de relé compatible:
https://www.elegoo.com/product/elegoo-4-channel-dc-5v-relay-module-with-optocoupler/

¡Precaución!

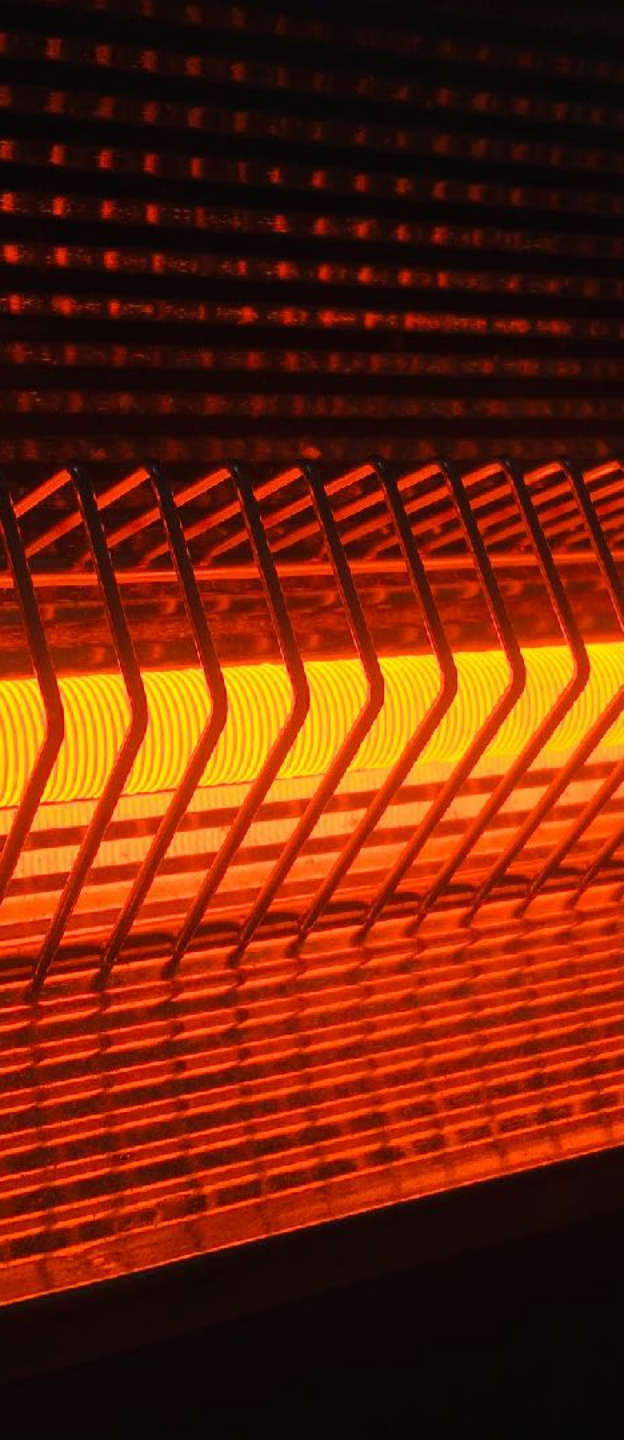
¡Alto voltaje!



Control de la caldera

Modo cableado

- **Contacta con un profesional**
- Lee el manual de tu caldera
- Corta la electricidad antes de manipularla



Control de la caldera

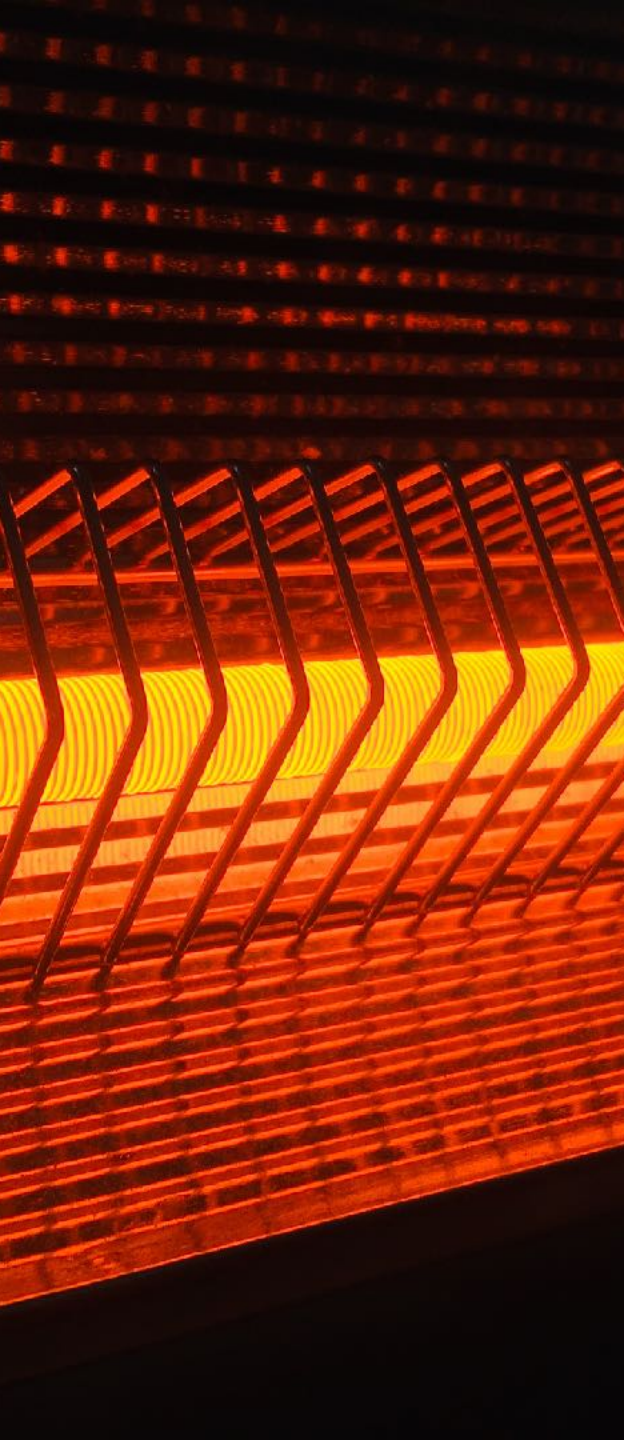
Modo cableado

gpiozero: https://gpiozero.readthedocs.io/en/stable/

```
# pipenv install gpiozero
```

```
from signal import pause
import gpiozero

relay_control_pin_number = 14  # Use the pin number the relay is connected to
relay = gpiozero.OutputDevice(
  relay_control_pin_number,
  active_high=False,
  initial_value=False
  )
relay.on()

pause()
```

Snippet: https://gitlab.com/hctpbl/heating-api/snippets/1855684



Control de la caldera

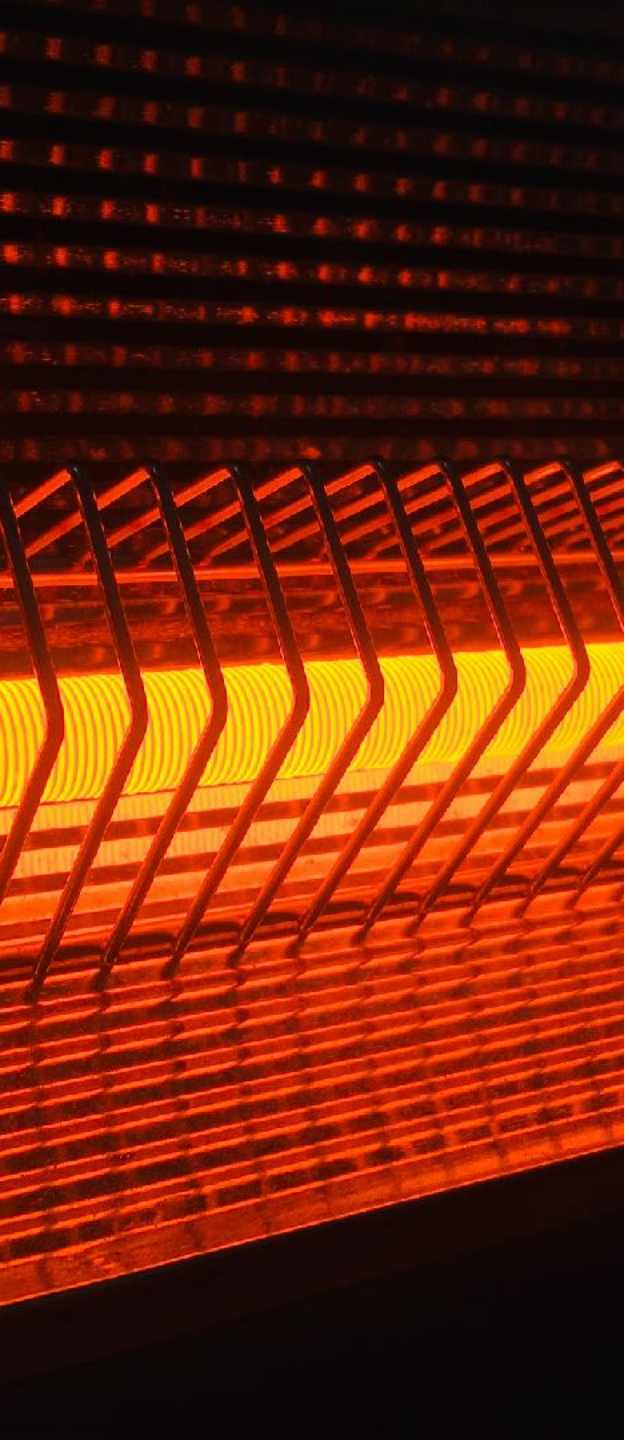
Modo cableado

Supervisor

```
[program:heating]
directory=/home/hector/projects/boiler
command=/home/hector/.local/bin/pipenv run python boiler_on.py
autostart=false
autorestart=false
stderr_logfile=/var/log/heating.err.log
stdout_logfile=/var/log/heating.out.log
user=hector
stopsignal=INT
stopwaitsecs=20
```

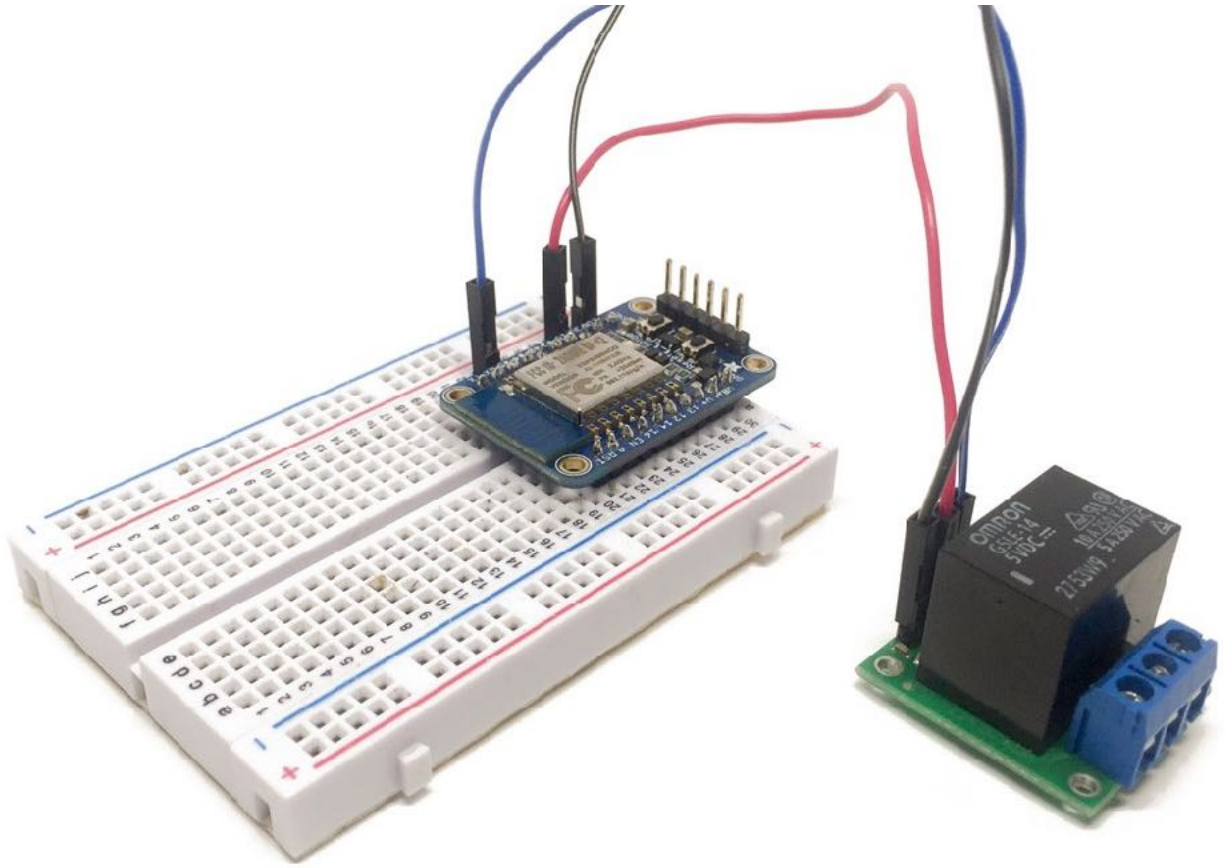
Encender caldera: `# sudo supervisorctl start heating`

Apagar caldera: `# sudo supervisorctl stop heating`

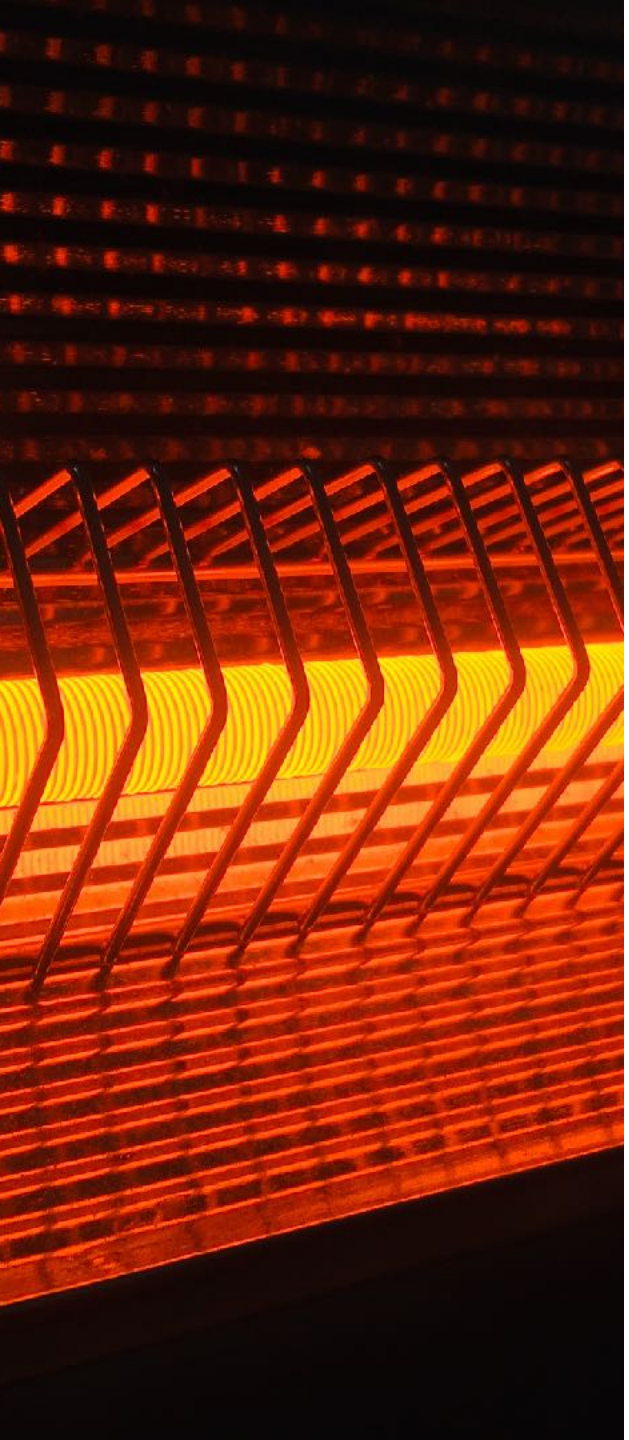


Control de la caldera

Modo wireless



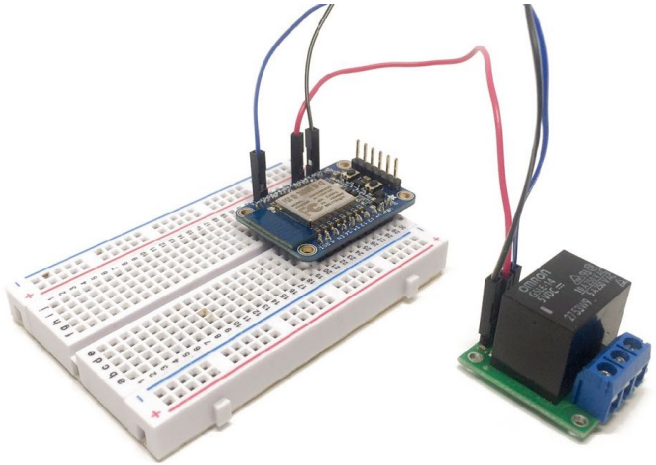
Fuente: https://openhomeautomation.net/control-relay-anywhere-esp8266



Control de la caldera

Modo wireless

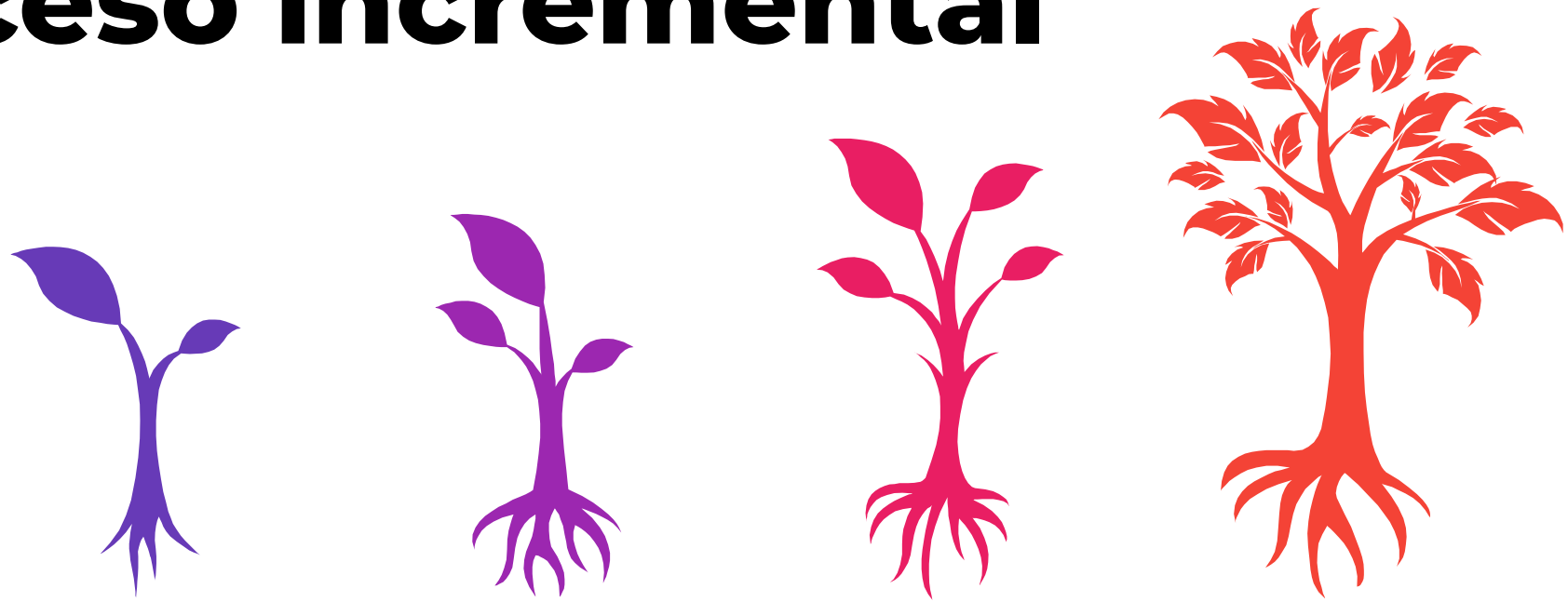
- Relé o módulo de relés
  - Activo a 5V
  - Soporte 230V AC
- Cables jumper (rPi a relé)
- Cables de corriente (relé a caldera)
- **Placa de conexión**
- **ESP8266 NodeMCU**
- **Fuente de alimentación**



Alrededor de 30€

Fuente: https://openhomeautomation.net/control-relay-anywhere-esp8266

Proceso incremental



01 - Medir Temperatura

Conectar nuestra raspberry a un sensor de temperatura y leer las mediciones

02 - Almacenar Mediciones

Guardar las mediciones de temperatura en un almacenamiento persistente

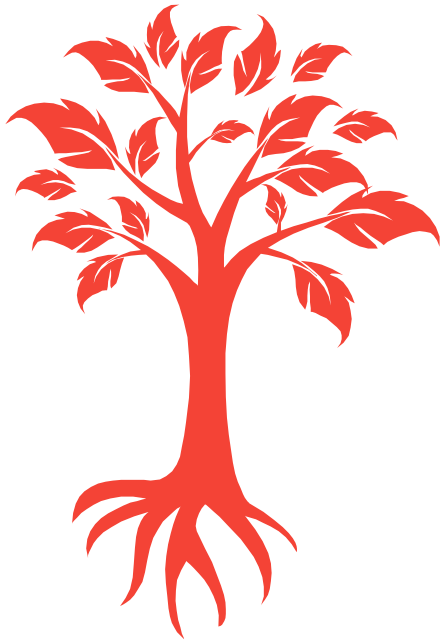
03 - Encender y apagar calefacción

Cómo interactuar con la caldera

04 - Acceso remoto y termostato

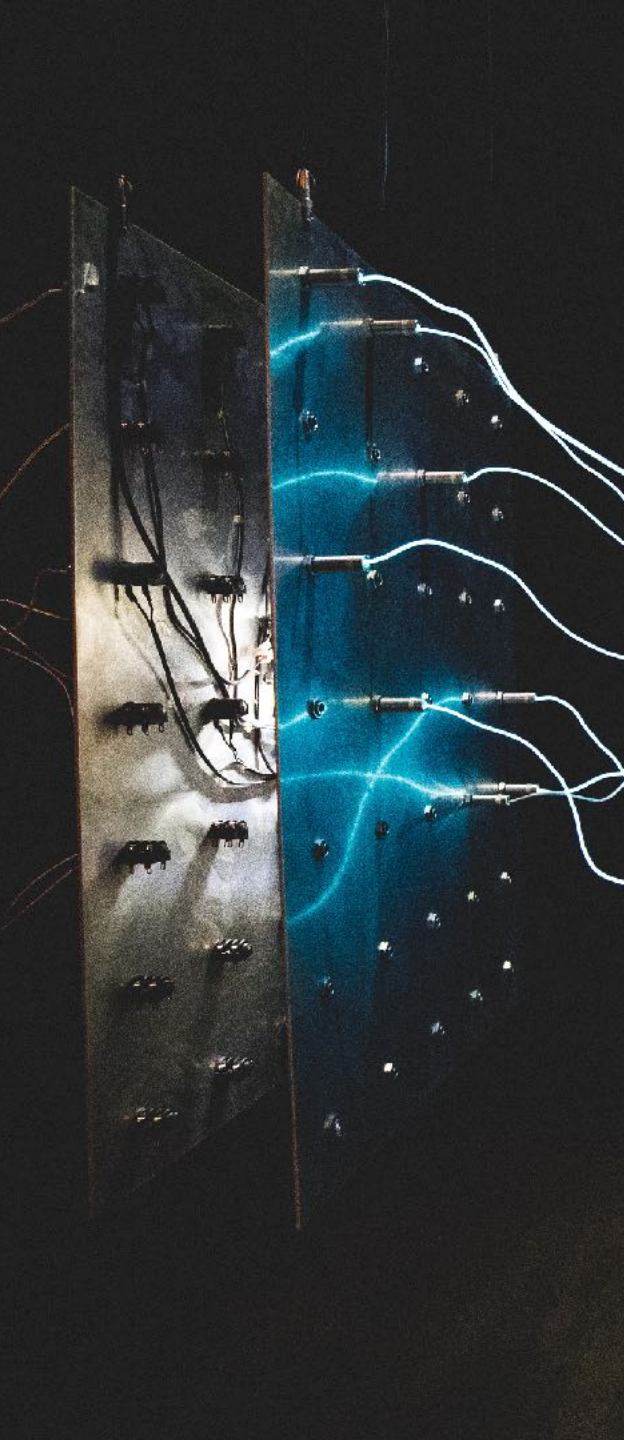
Proporcionamos la funcionalidad controlar la calefacción de forma remota

Proceso incremental

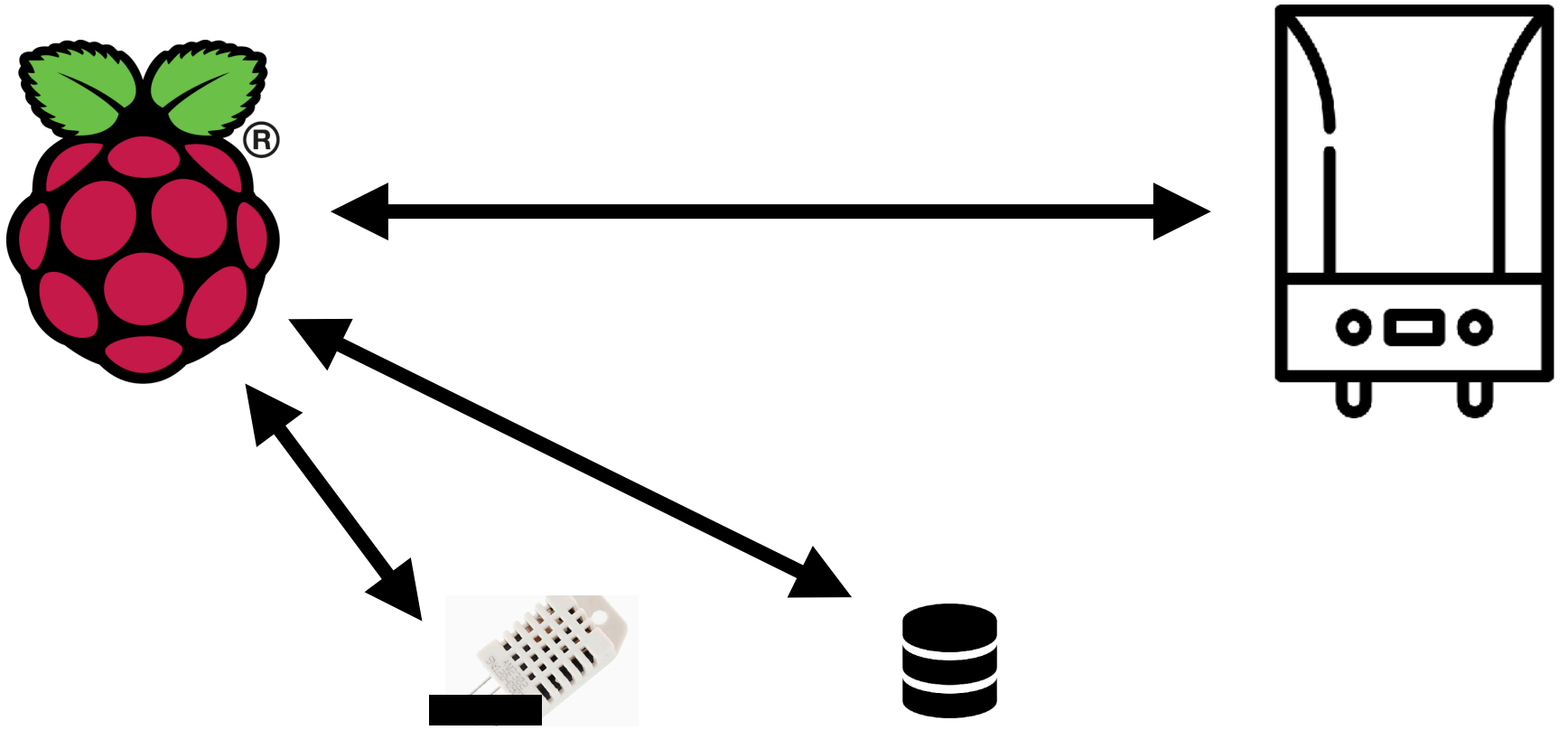


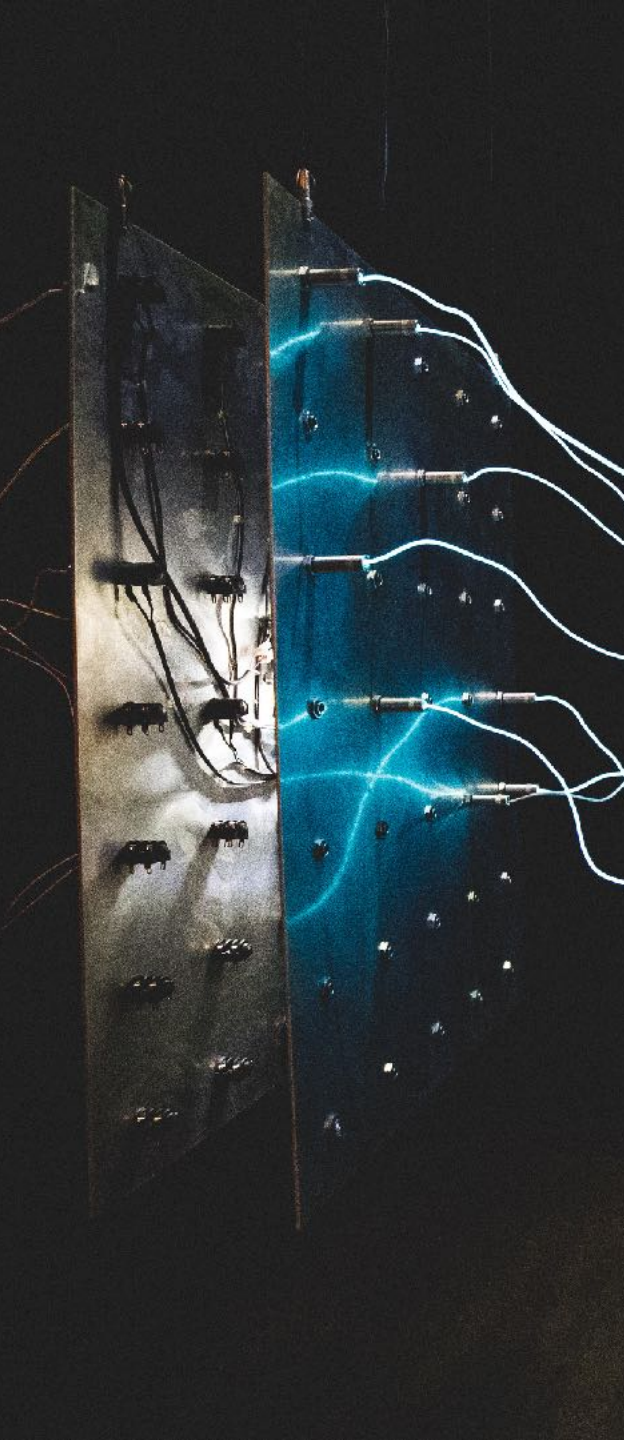
04 - Acceso remoto y termostato

Proporcionamos la funcionalidad controlar la calefacción de forma remota

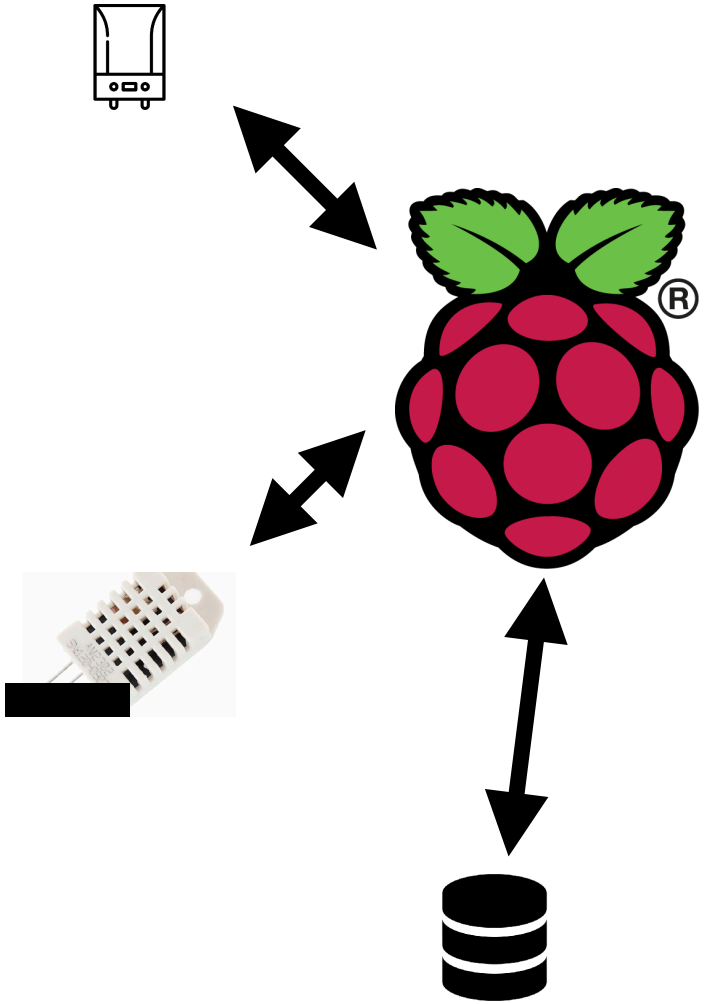


Acceso remoto y termostato





Acceso remoto y termostato



Desde la red local

Encender caldera:

```
# sudo supervisorctl start heating
```

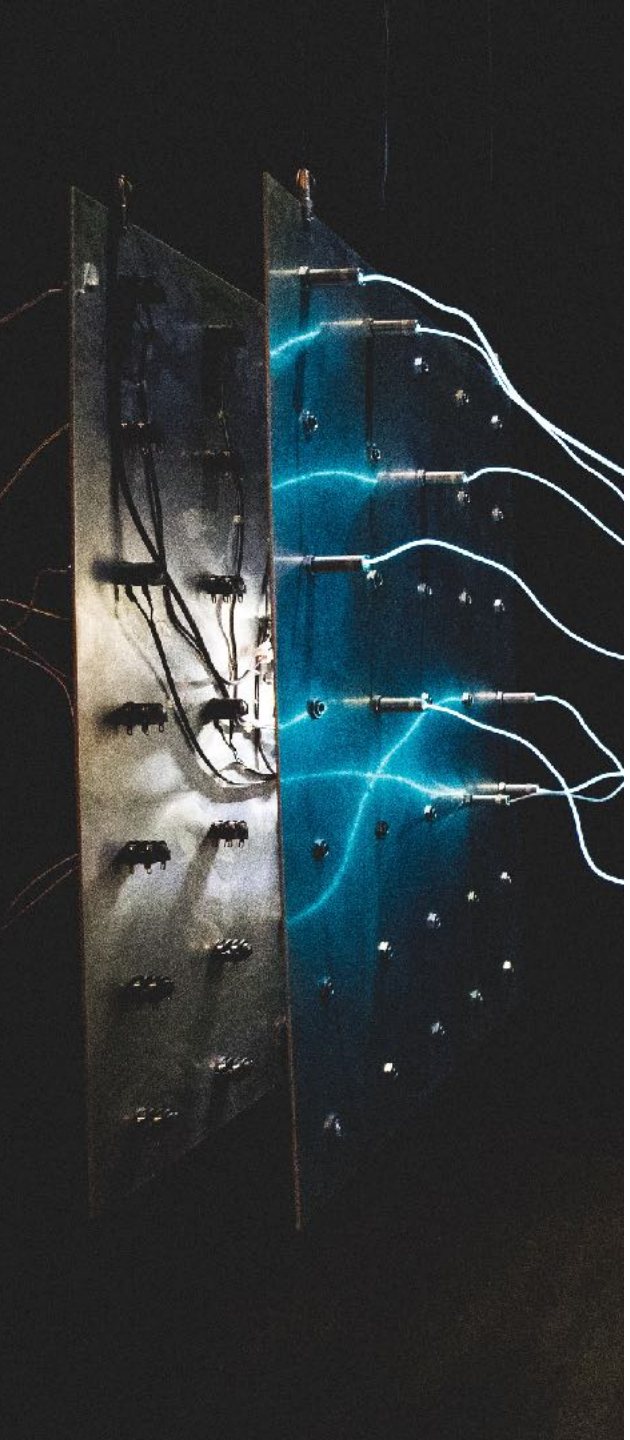
Apagar caldera:

```
# sudo supervisorctl stop heating
```

Consultar temperaturas:

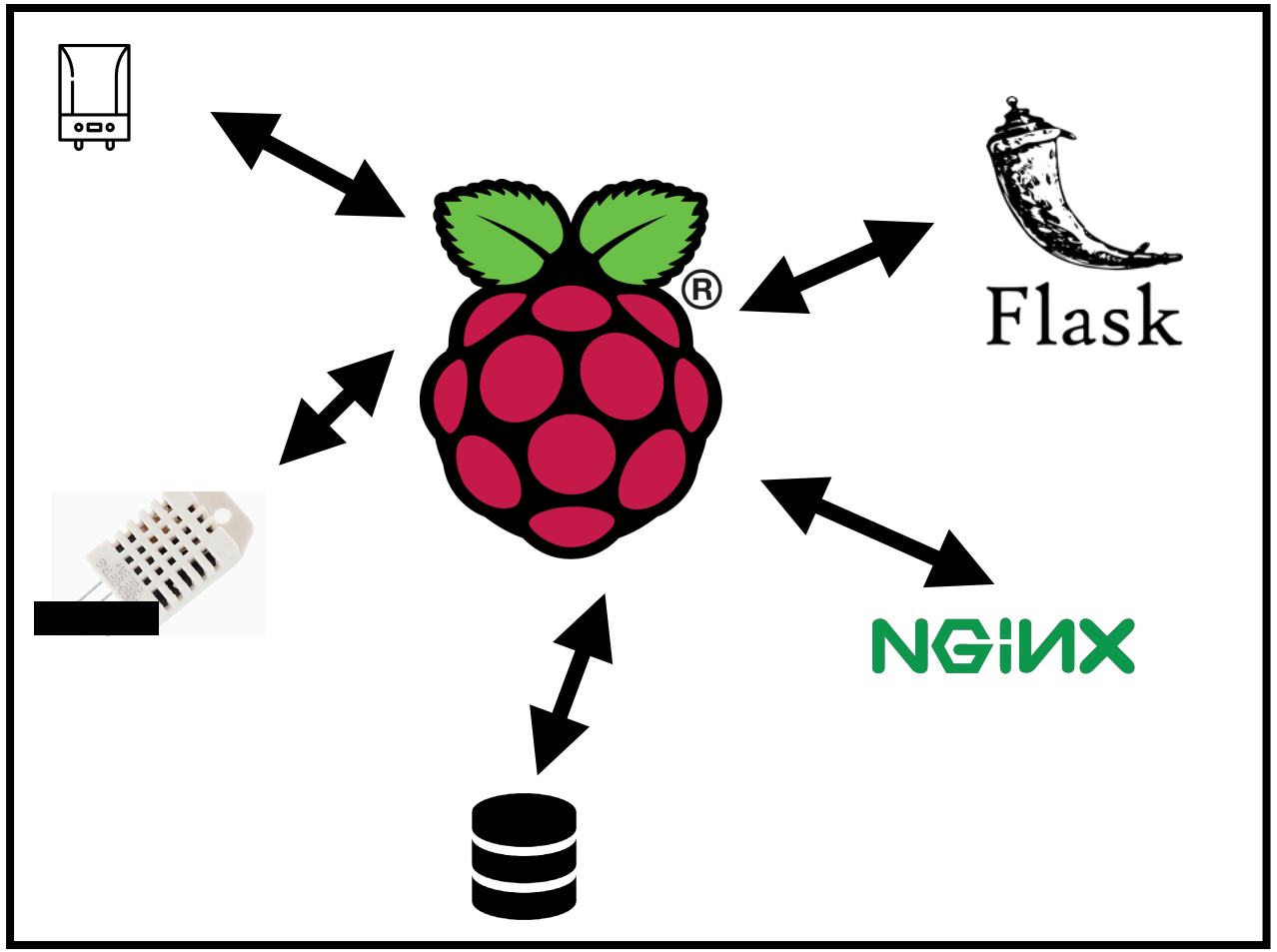
```
influxdb >
select temperature, humidity
from environment
order by time desc
limit 1;
```

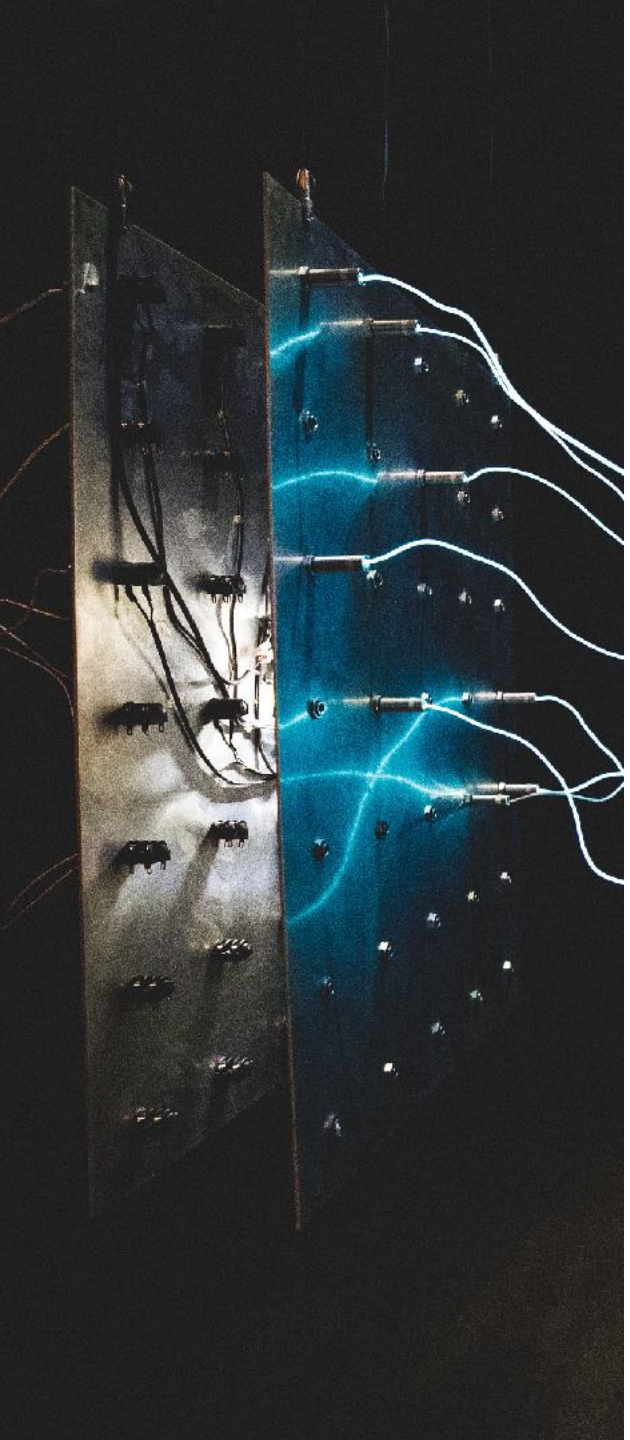
chronograf



Acceso remoto y termostato

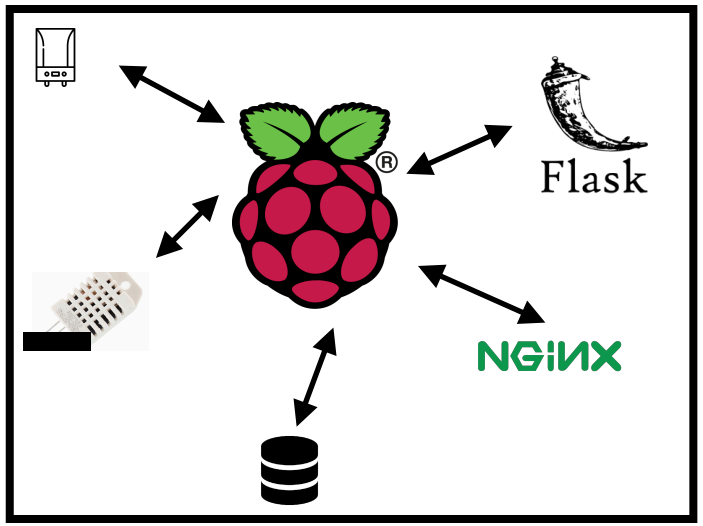
HTTP API





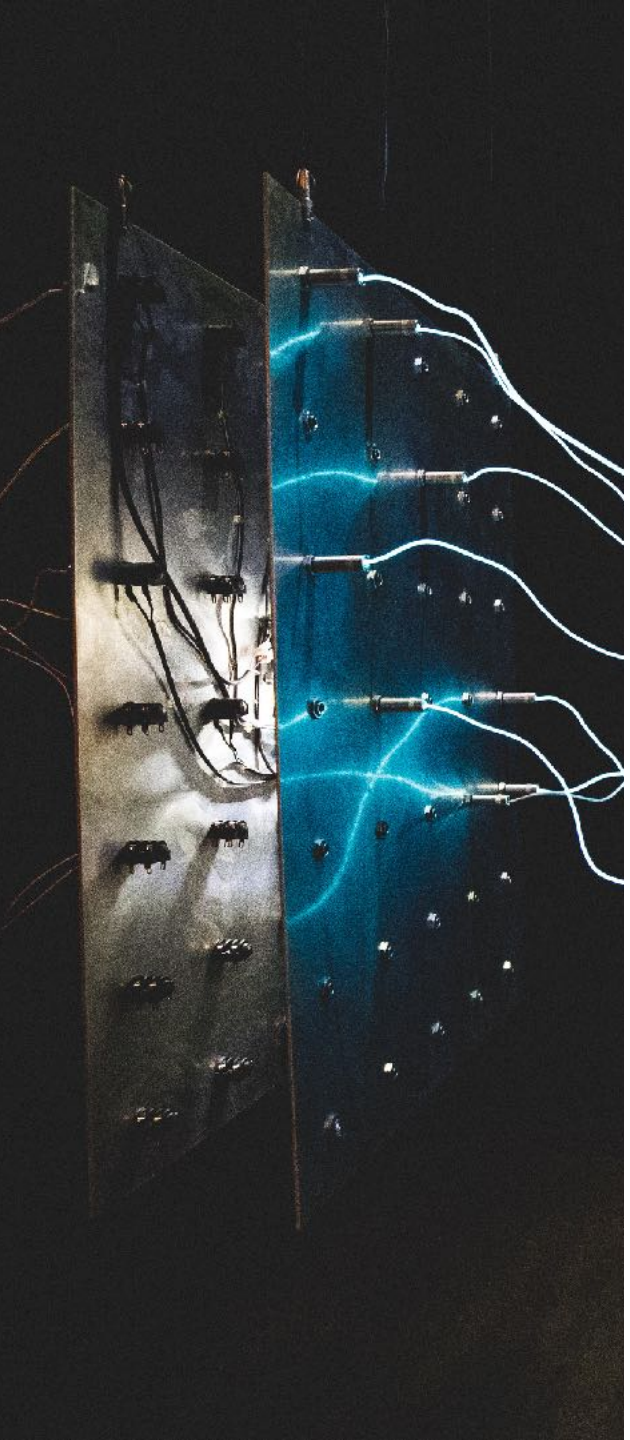
Acceso remoto y termostato

REST API



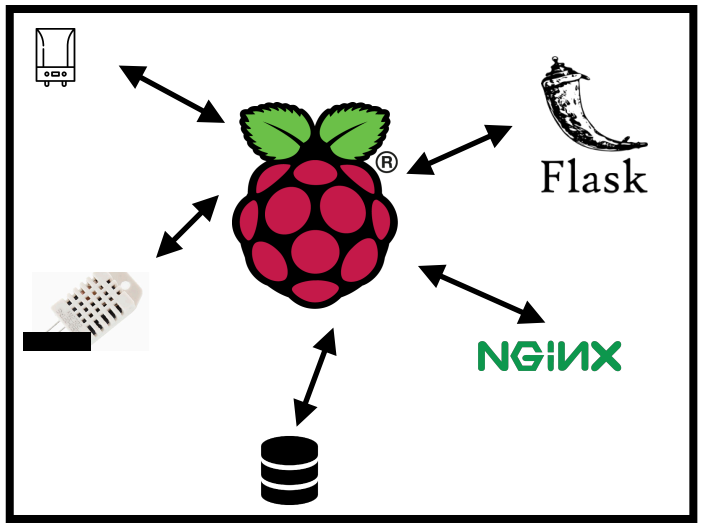
Pasos previos recomendados:

- Apertura de puertos 80 y 443 en el router
- Nombre de dominio
  https://www.noip.com/
- Cliente de DNS dinámico
  - Dynamic DNS Update Client (DUC)
  - ddclient
- HTTPS
  https://letsencrypt.org/es/



Acceso remoto y termostato

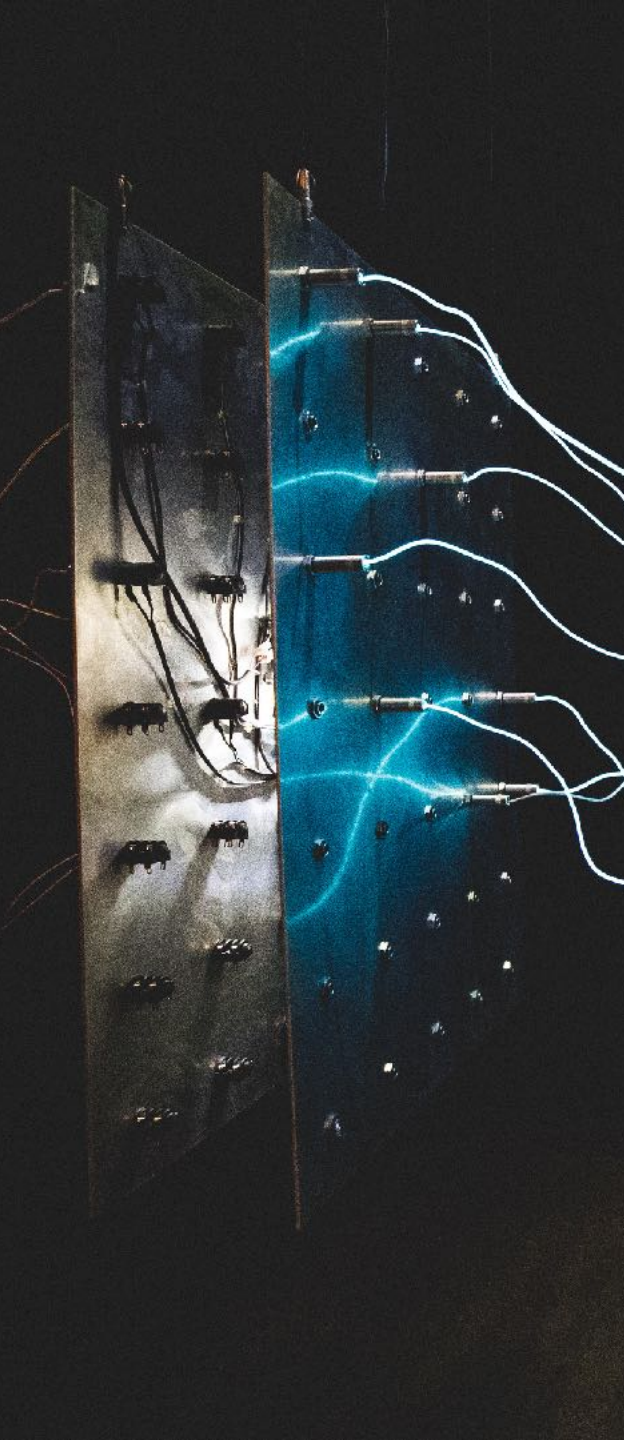
REST API



Consultar temperaturas:

```
influxdb >
select temperature, humidity
from environment
order by time desc
limit 1;
```

chronograf

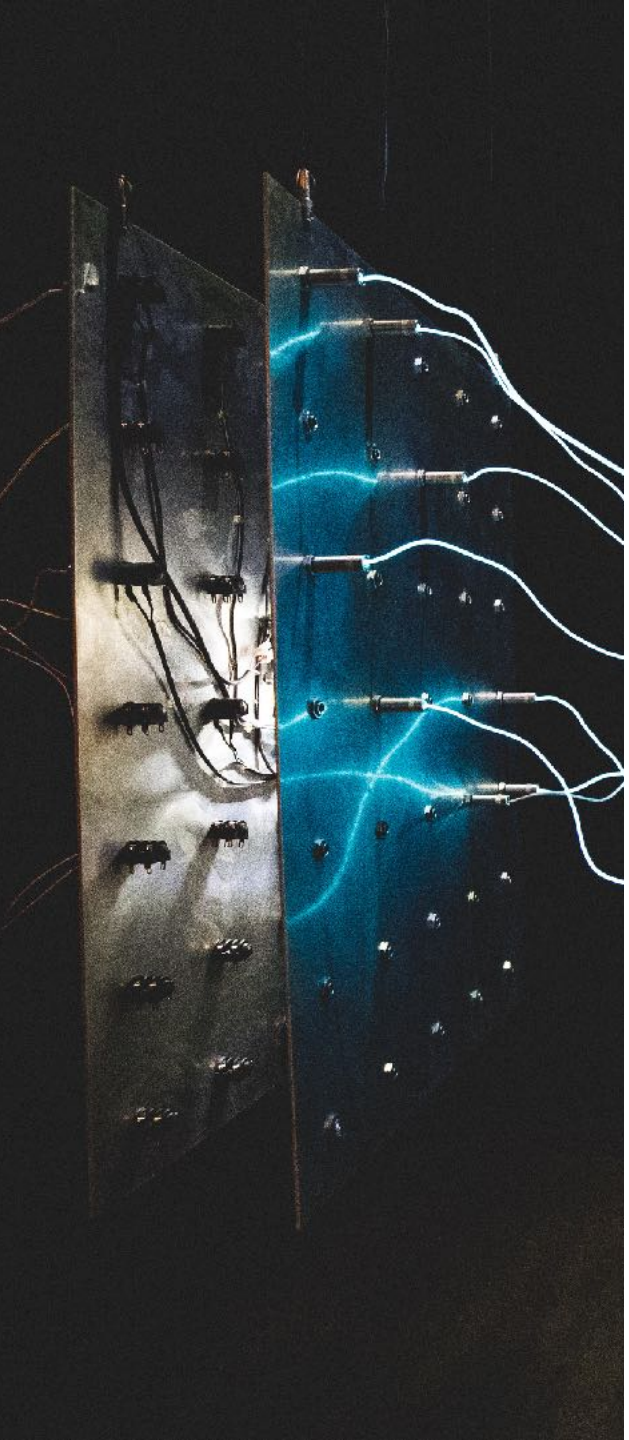


Acceso remoto y termostato

REST API

Consultar temperaturas:

```
from flask import jsonify
from flask_login import login_required
from influxdb import InfluxDBClient

from heating_api import app

@app.route('/environment', methods=['GET'])
@login_required
def get_environmental_data():
    influx_config = app.config['INFLUX_DB']
    client = InfluxDBClient(
        host=influx_config['HOST'],
        port=influx_config['PORT'],
        database='home'
    )
    result = client.query(
        'select temperature, humidity from environment order by time desc limit 1;'
    )
    temperature_data = list(result.get_points())[0]
    temperature_data['temperature'] = round(temperature_data['temperature'], 2)
    temperature_data['humidity'] = round(temperature_data['humidity'], 2)
    return jsonify(temperature_data), 200
```



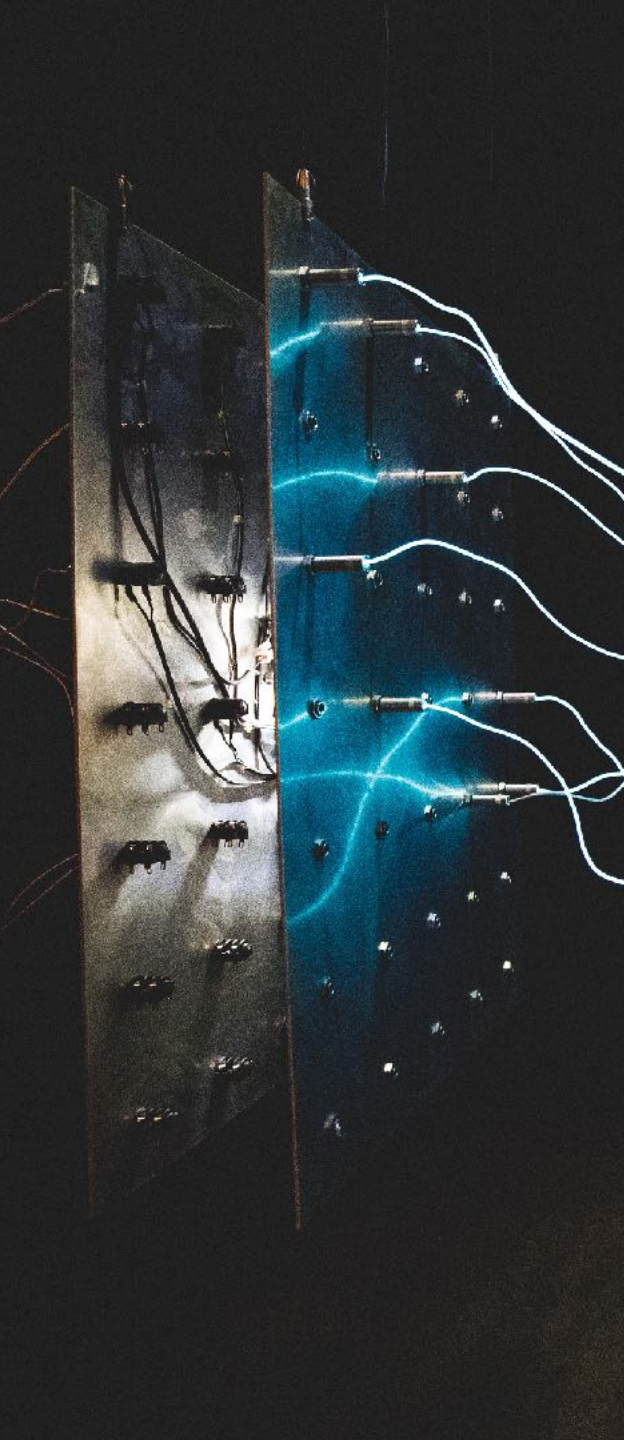
Acceso remoto y termostato

REST API

Consultar temperaturas:

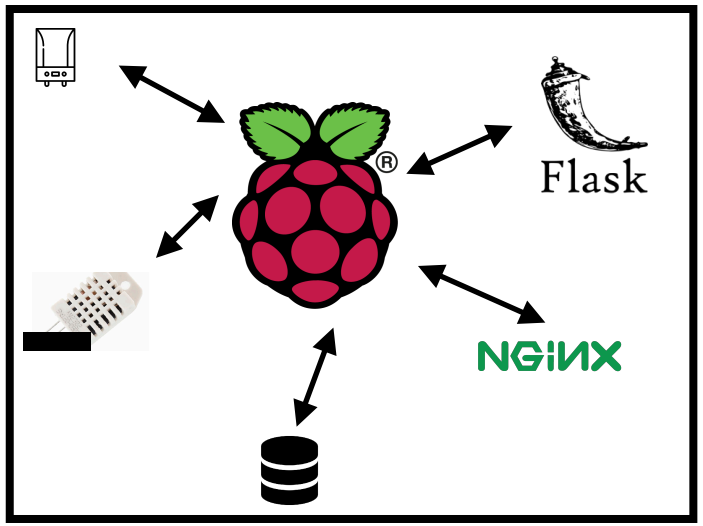
```
GET https://nuestraapi.com/environment
```


```
{
   "humidity": 55.2,
   "temperature": 23.9,
   "time": "2019-09-22T00:00:18.059798955Z"
}
```



Acceso remoto y termostato

REST API



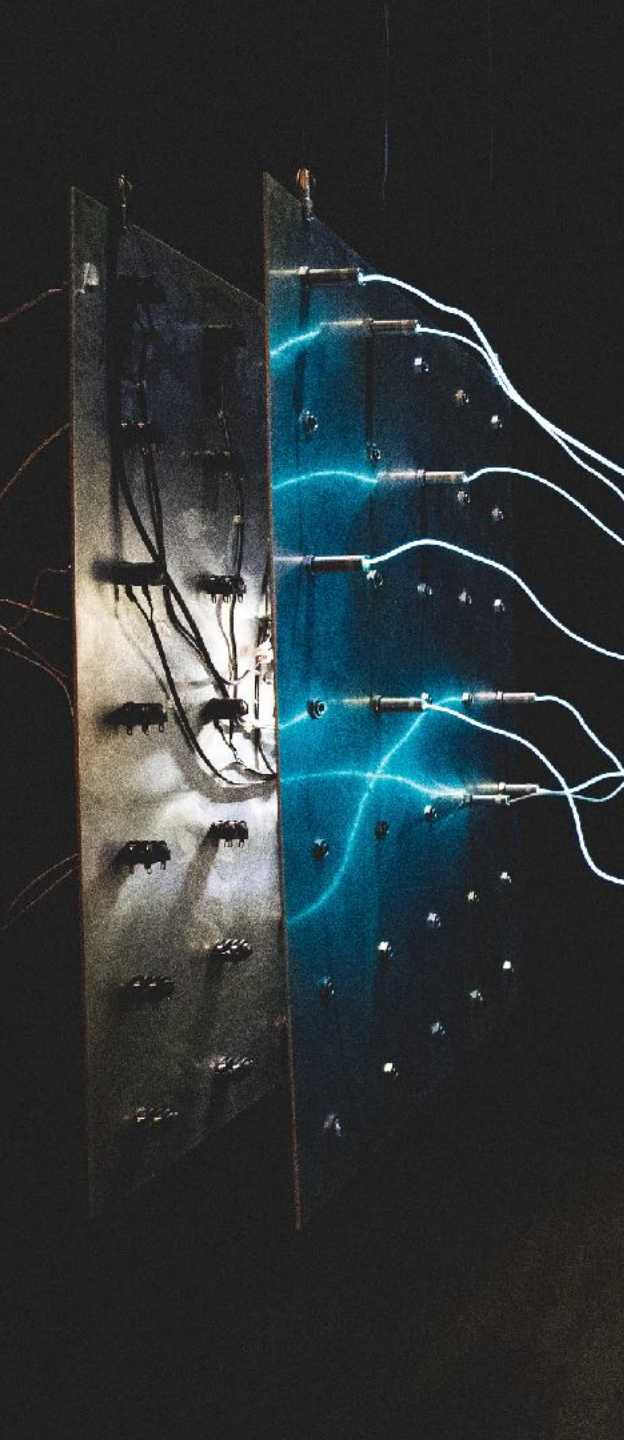
Control de la caldera:

Encender caldera:

```
# sudo supervisorctl start heating
```

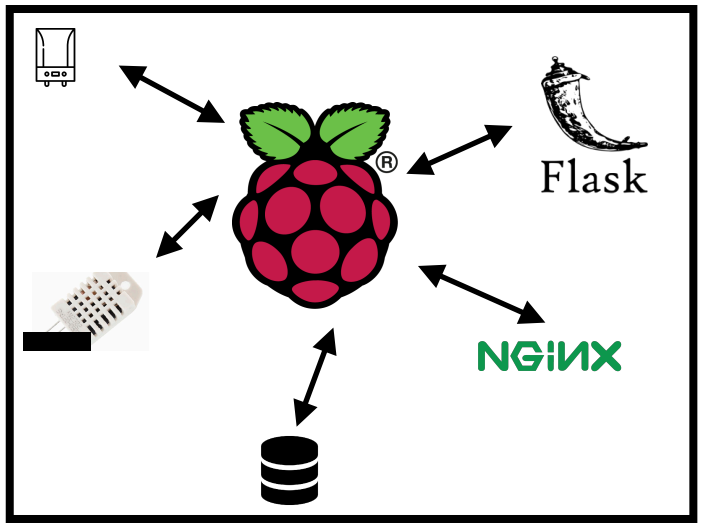
Apagar caldera:

```
# sudo supervisorctl stop heating
```



Acceso remoto y termostato

REST API



Control de la caldera:

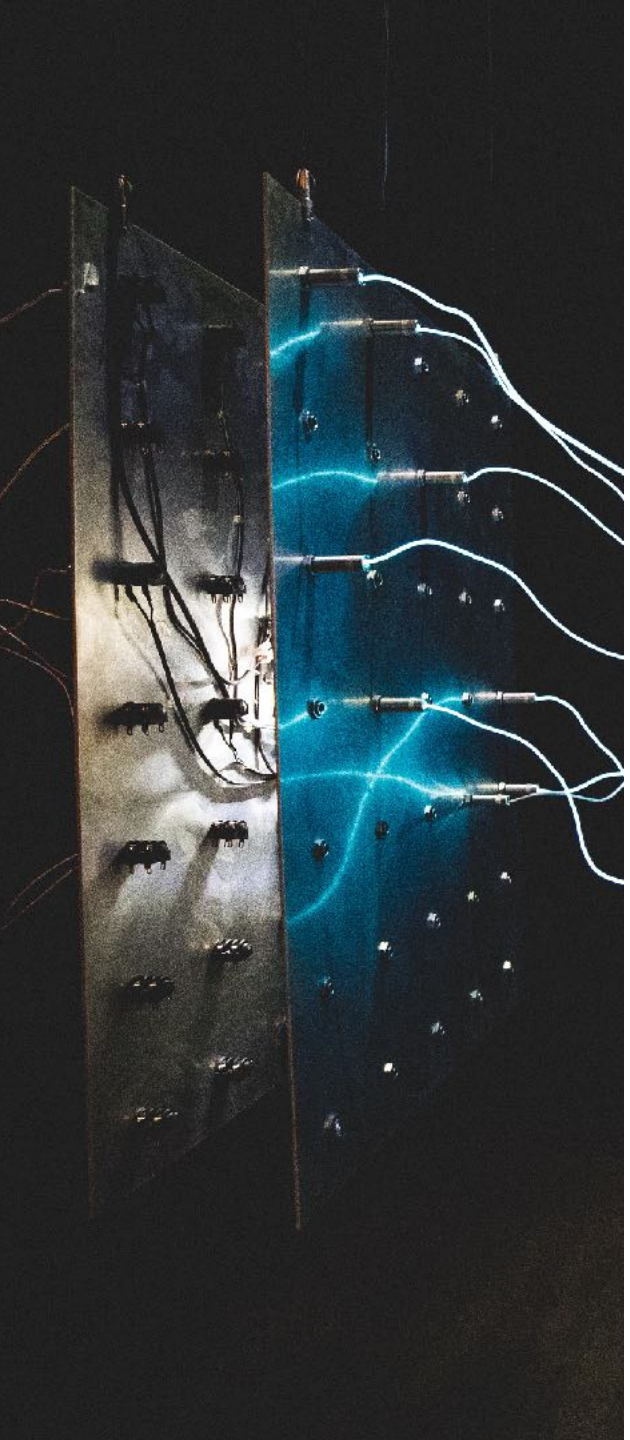
Supervisor XML-RPC API

http://supervisord.org/api.html

supervisord.conf

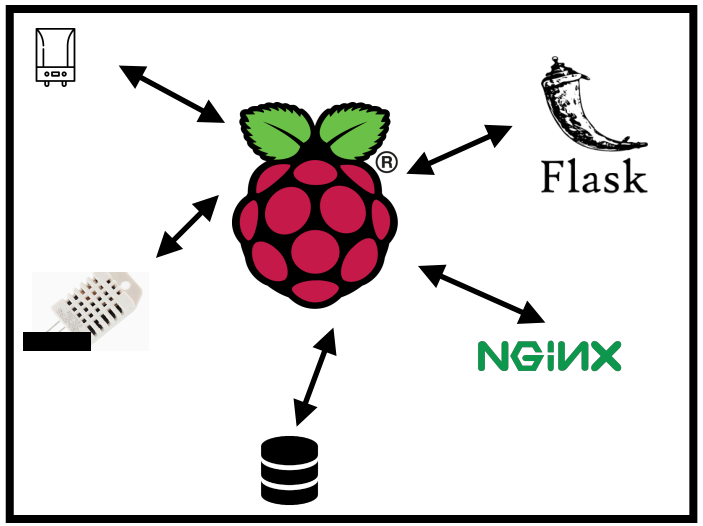
```
[inet_http_server]
port = 127.0.0.1:9001
username = user
password = 123
```

```
from xmlrpc.client import ServerProxy
server = ServerProxy('http://localhost:9001/RPC2')

# Encender la caldera
server.supervisor.startProcess('heating')

# Apagar la caldera
server.supervisor.stopProcess('heating')
```



Acceso remoto y termostato

REST API



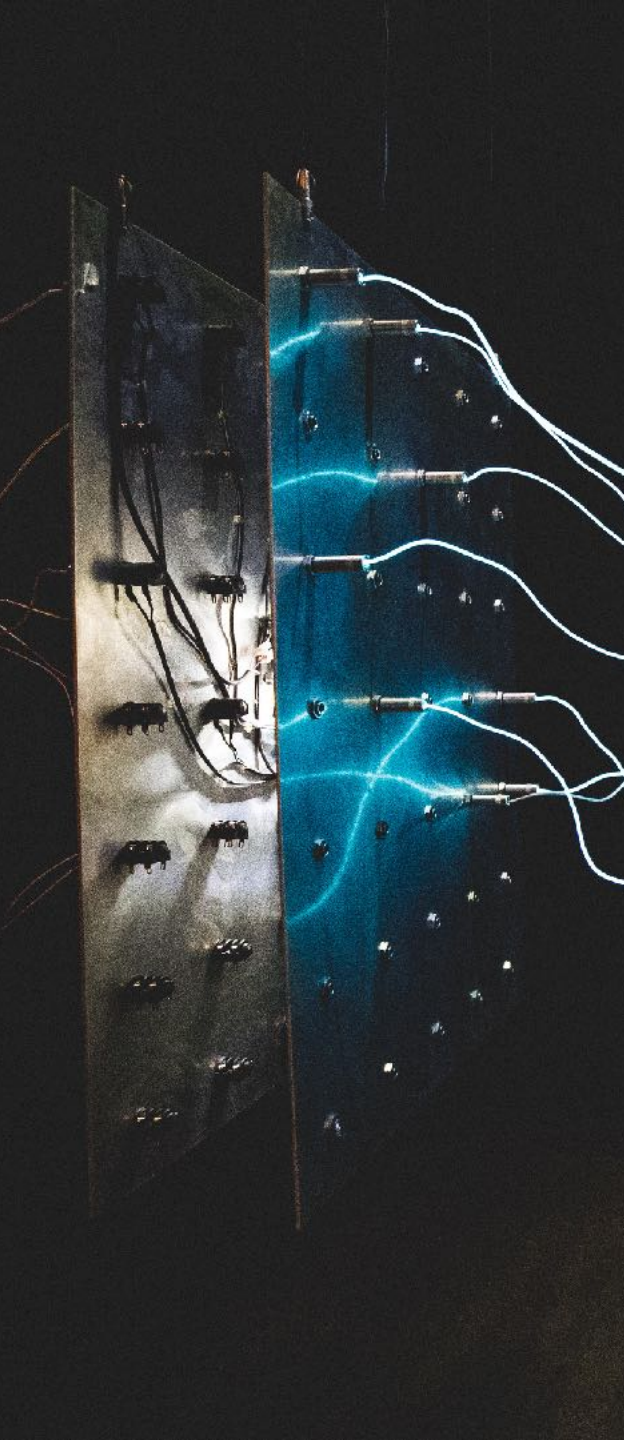
Control de la caldera:

```
from xmlrpc.client import ServerProxy
server = ServerProxy('http://localhost:9001/RPC2')

# Encender la caldera
server.supervisor.startProcess('heating')

# Apagar la caldera
server.supervisor.stopProcess('heating')
```

GitLab

Cliente de supervisor:

https://gitlab.com/hctpbl/heating-api/blob/master/supervisor_client/supervisor_client.py

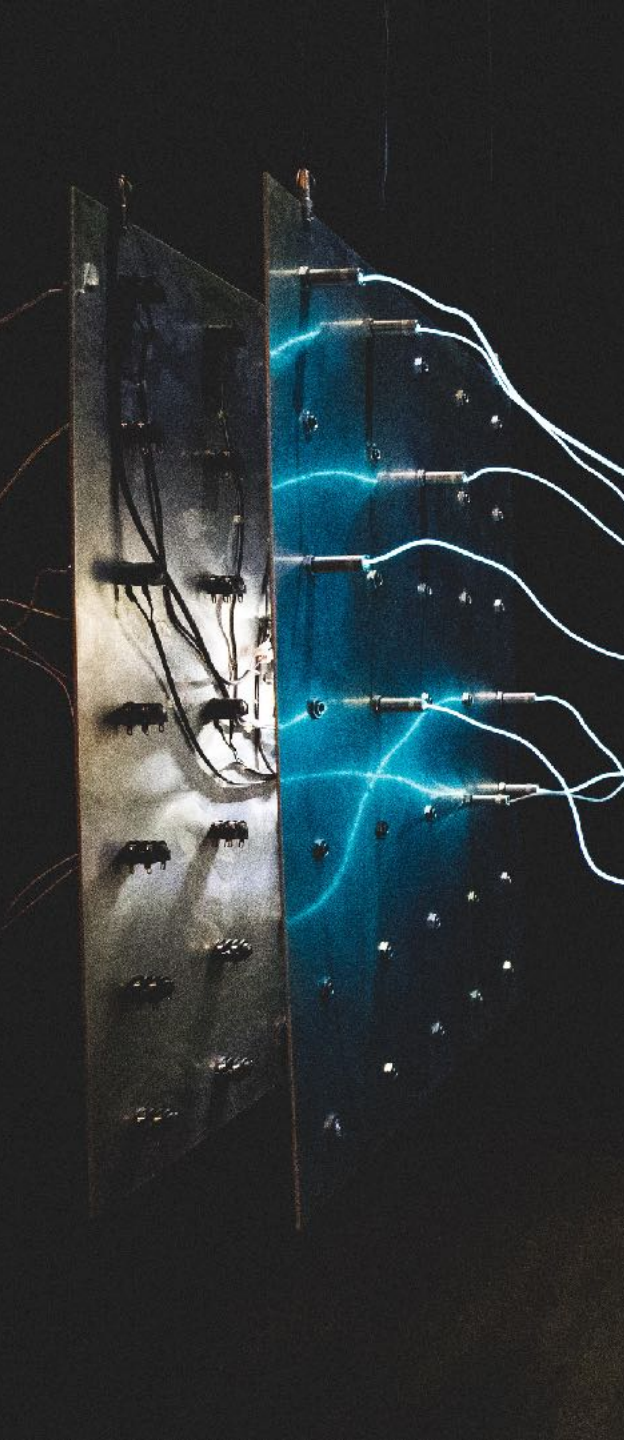


Acceso remoto y termostato

REST API

Control de la caldera:

```
from flask import jsonify, request
from flask_login import login_required

from heating_api import app
from supervisor_client.supervisor_client import SupervisorClient, ProcessStatus


@app.route('/heating', methods=['GET', 'PUT'])
@login_required
def heating_status():
    supervisor_config = app.config['SUPERVISOR']
    supervisor_client = SupervisorClient(
        host=supervisor_config['HOST'],
        port=supervisor_config['PORT'],
        username=supervisor_config['USERNAME'],
        password=supervisor_config['PASSWORD']
    )
    if request.method == 'PUT':
        data = request.get_json()
        if data['heating_on']:
            supervisor_client.start('heating')
        else:
            supervisor_client.stop('heating')
    status = supervisor_client.status('heating')
    is_heating_on = status == ProcessStatus.RUNNING
    return jsonify(heating_on=is_heating_on), 200
```



Acceso remoto y termostato

REST API

Control de la caldera:

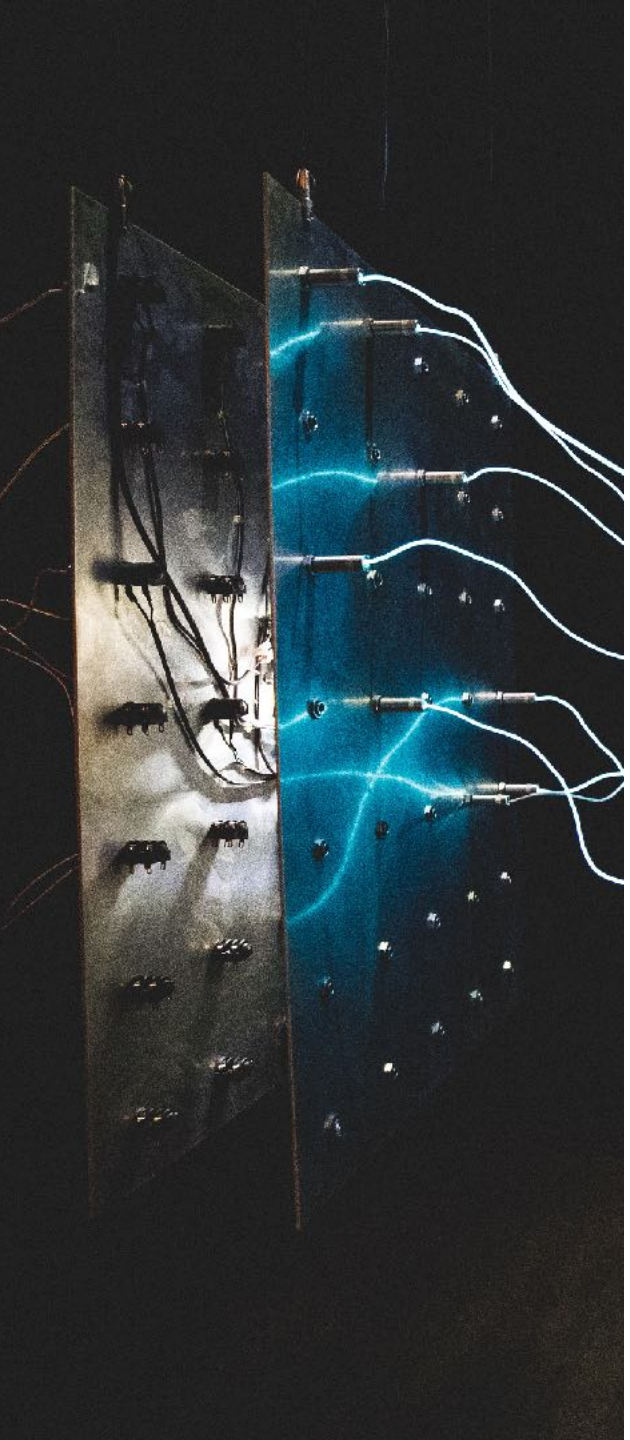
`GET https://nuestraapi.com/heating`

```
{
   "heating_on": false
}
```

`PUT https://nuestraapi.com/heating, {“heating_on”: true}`

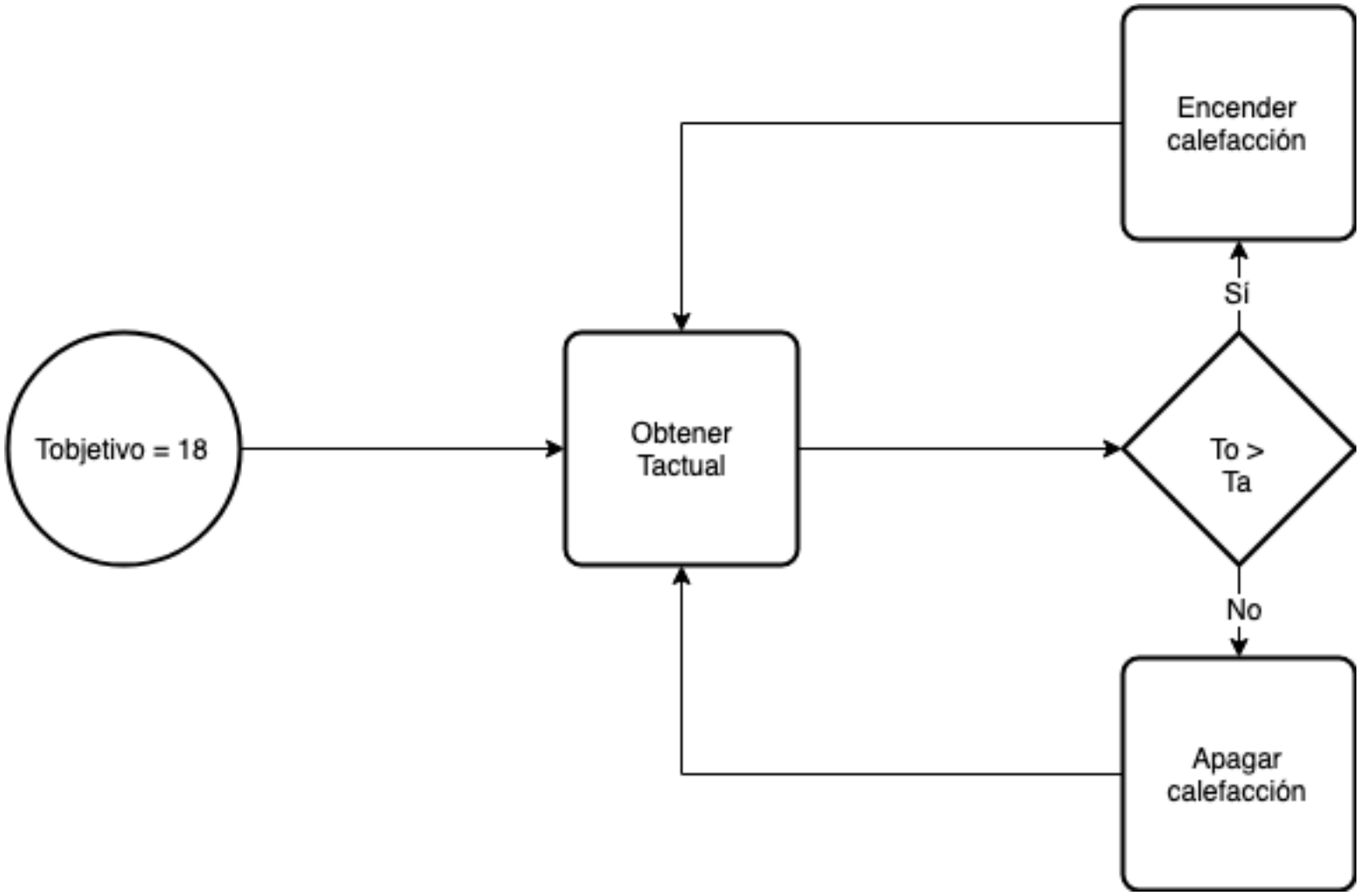
`GET https://nuestraapi.com/heating`

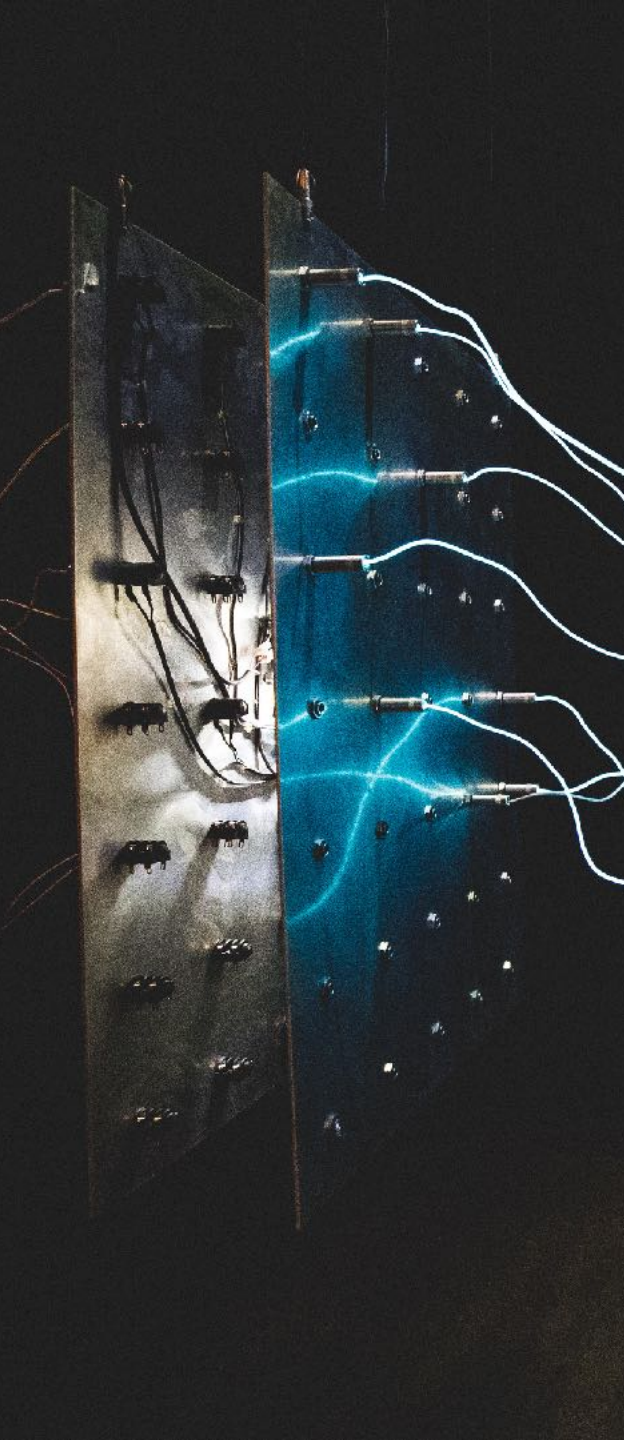
```
{
   "heating_on": true
}
```



Acceso remoto y termostato

TERMOSTATO



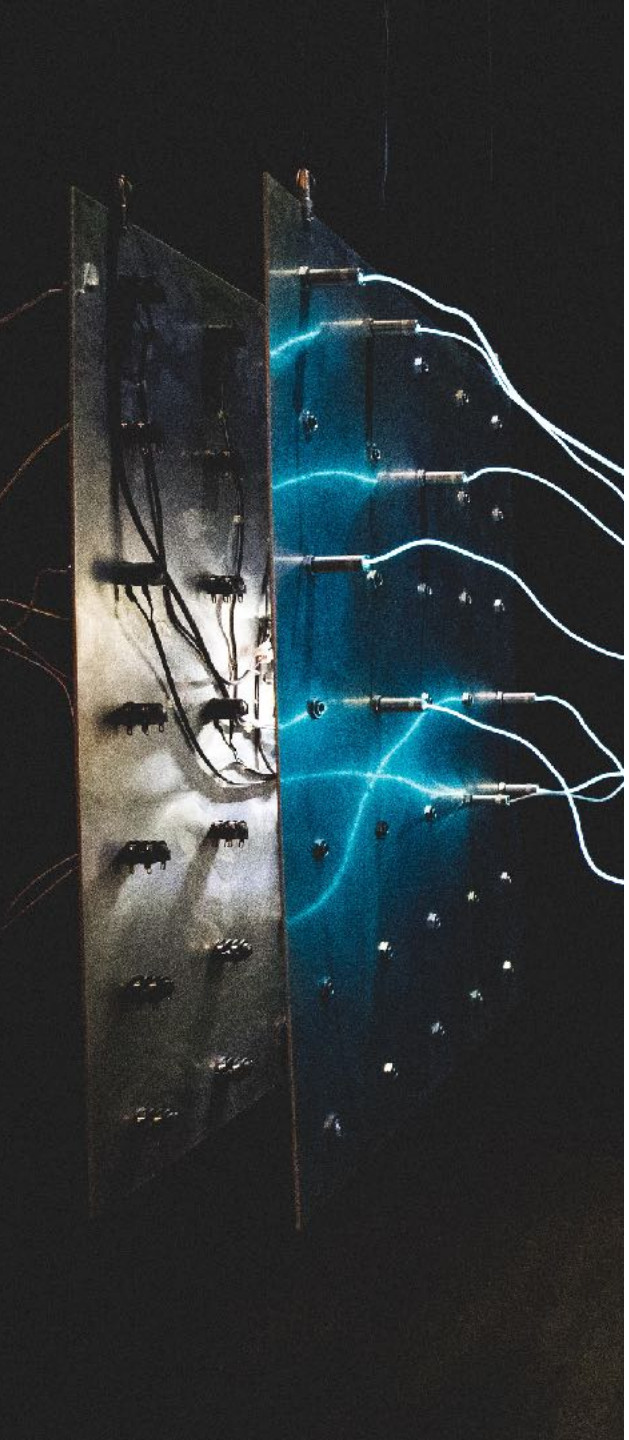


Acceso remoto y termostato

TERMOSTATO

Bucle termostatáto

```
def thermostat_check(target_temperature):
    current_temperature = get_current_temperature()
    is_heating_on = get_is_heating_on()
    if current_temperature < target_temperature and not is_heating_on:
        turn_heating_on()
    elif current_temperature >= target_temperature and is_heating_on:
        turn_heating_off()
```



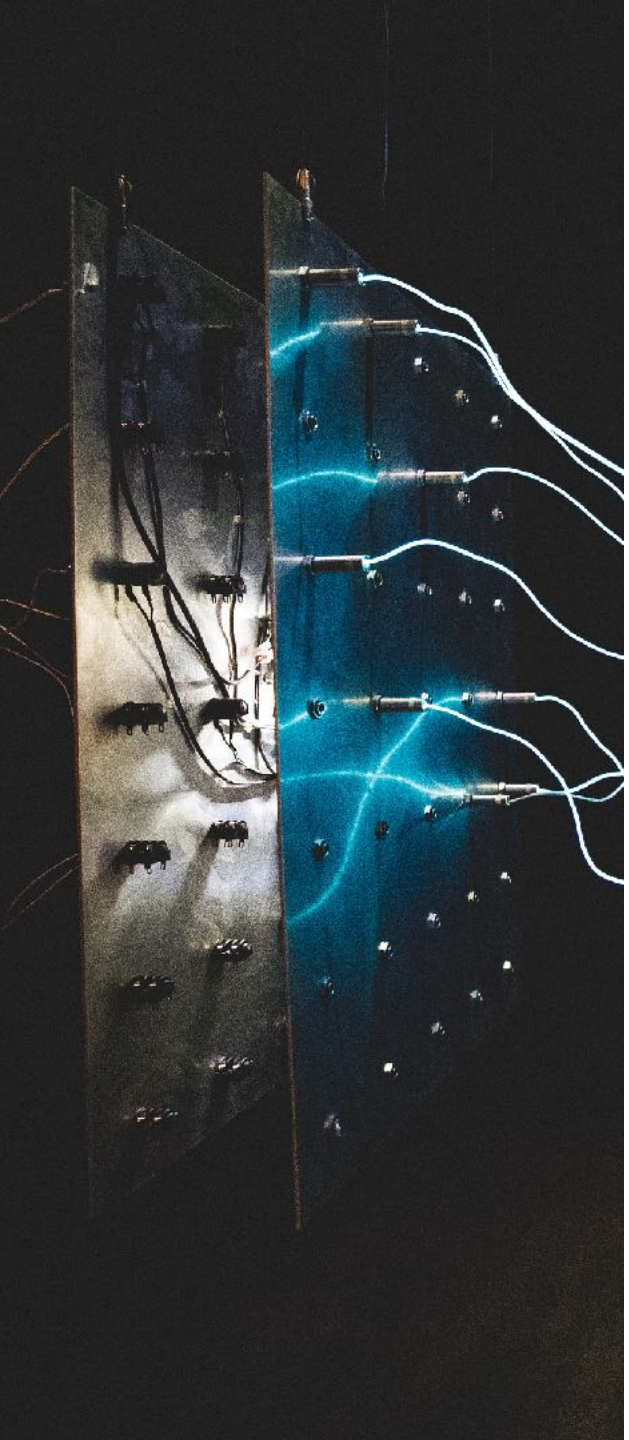
Acceso remoto y termostato

TERMOSTATO

apscheduler: https://apscheduler.readthedocs.io/en/latest/

Inicializar termostato

```
from apscheduler.schedulers.background import BackgroundScheduler
from somewhere import thermostat_check_function


target_temperature = 18
scheduler = BackgroundScheduler({'apscheduler.timezone': 'Europe/Madrid'})
scheduler.start()
scheduler.add_job(
    thermostat_check,
    'interval',
    seconds=60,
    args=[target_temperature],
    id=THERMOSTAT_JOB_ID
)
```



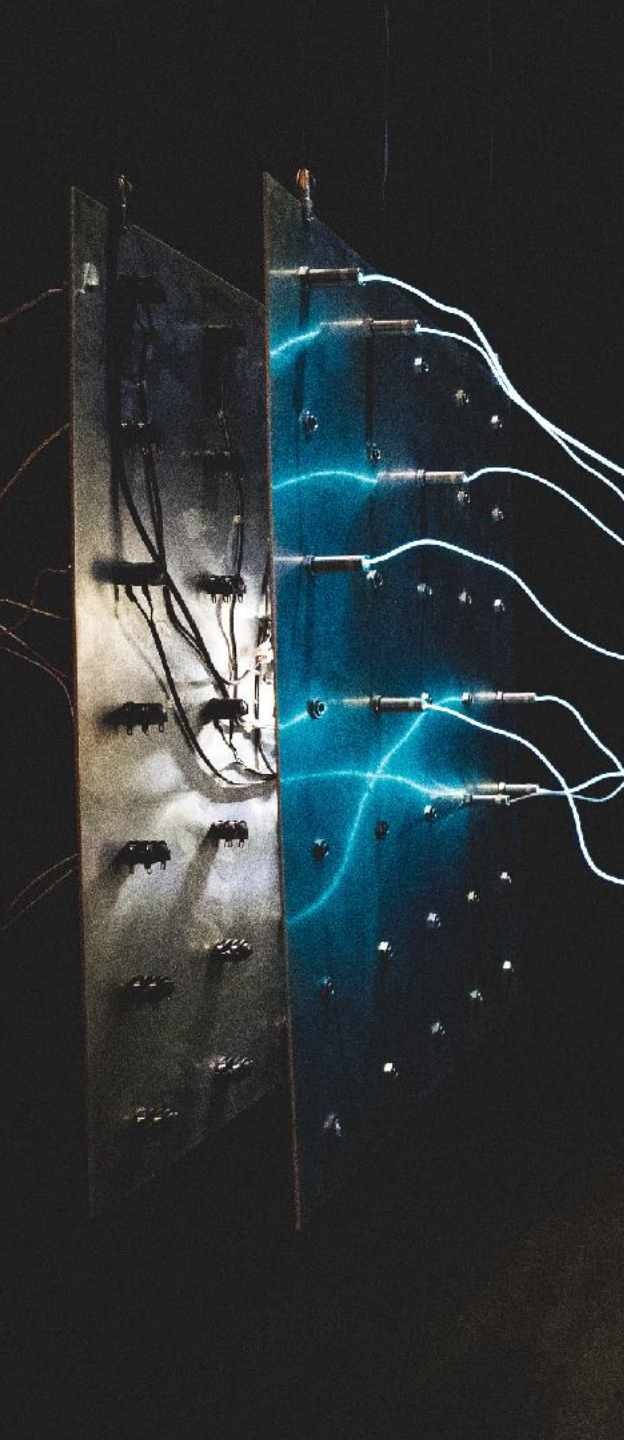
Acceso remoto y termostato

TERMOSTATO

apscheduler: https://apscheduler.readthedocs.io/en/latest/

Modificar termostato

```
current_thermostat_job = scheduler.get_job(THERMOSTAT_JOB_ID)
current_thermostat_job.modify(args=[new_target_temp])
```



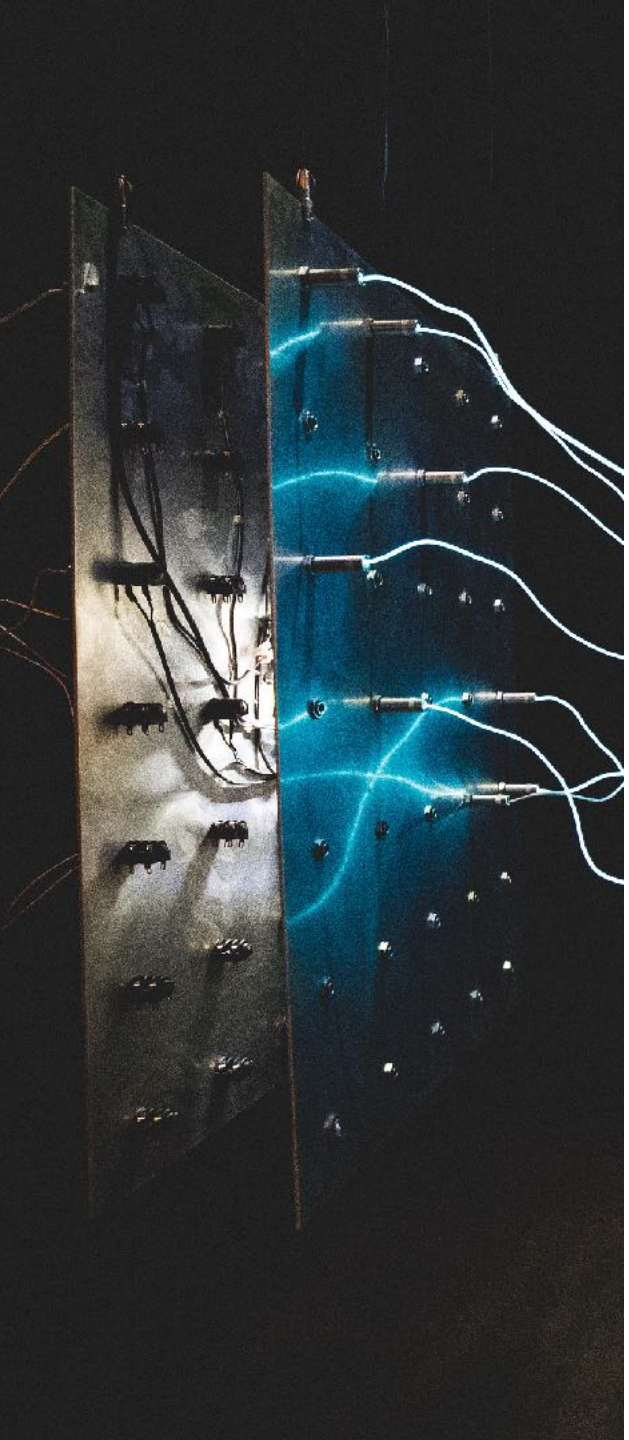
Acceso remoto y termostato

TERMOSTATO

apscheduler: https://apscheduler.readthedocs.io/en/latest/

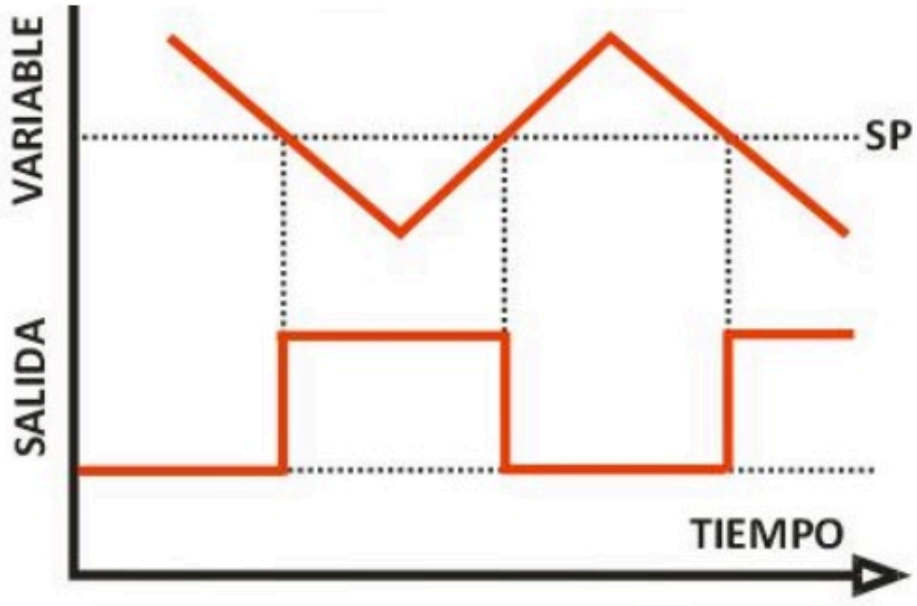
Eliminar termostato (volver a modo manual)

```
current_thermostat_job = scheduler.get_job(THERMOSTAT_JOB_ID)
current_thermostat_job.remove()
```



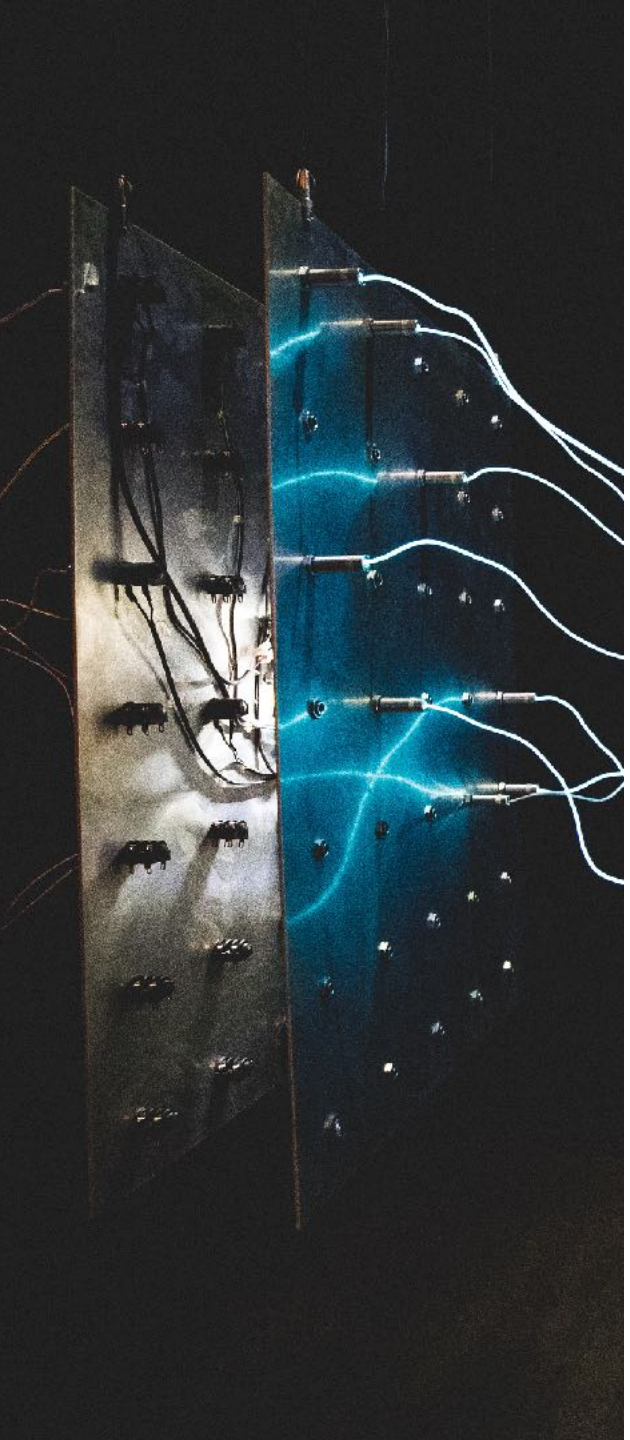
Acceso remoto y termostato

TERMOSTATO



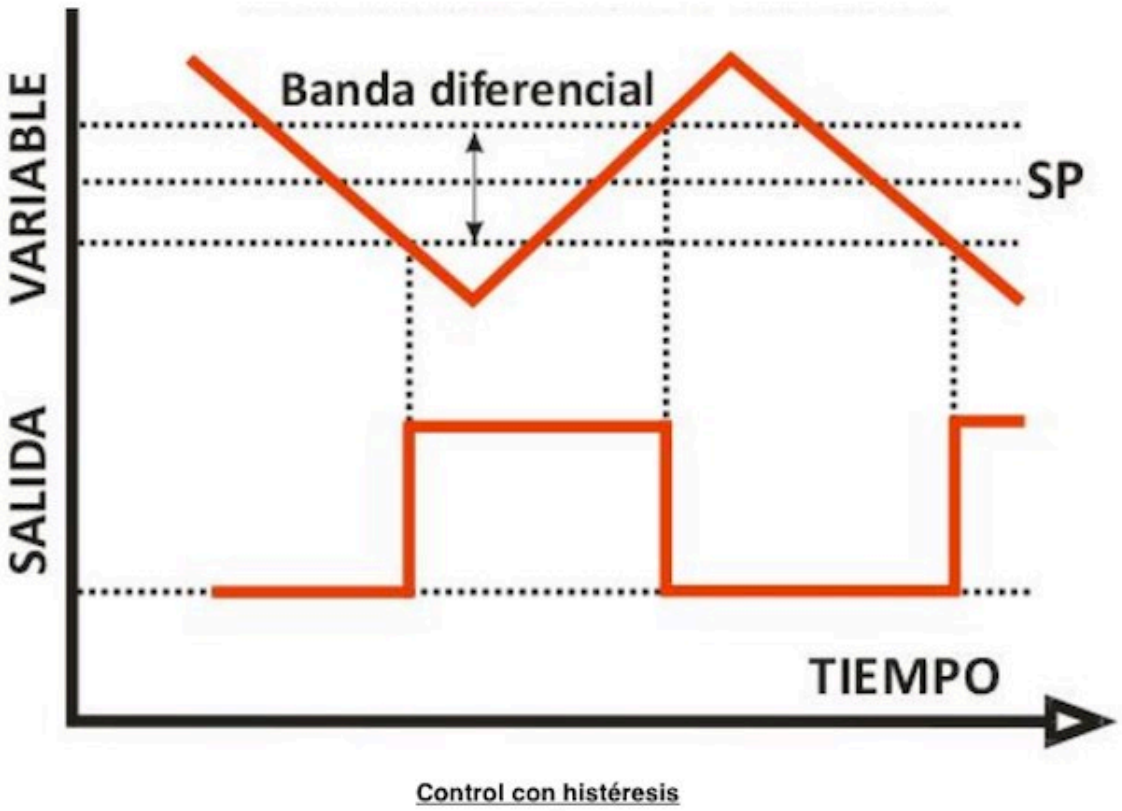
Control todo/nada sin histéresis

FUENTE: http://www.domoticadomestica.com/termostatos-inteligentes-control-pid/



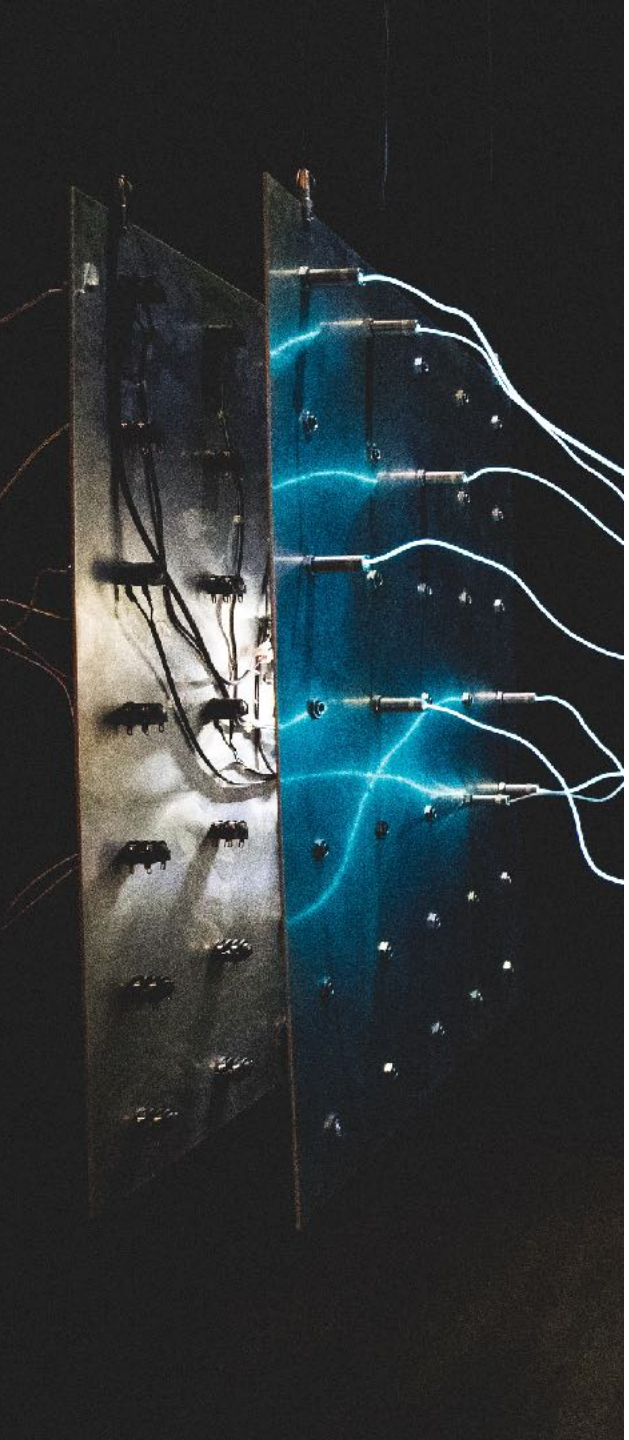
Acceso remoto y termostato

TERMOSTATO



Control con histéresis

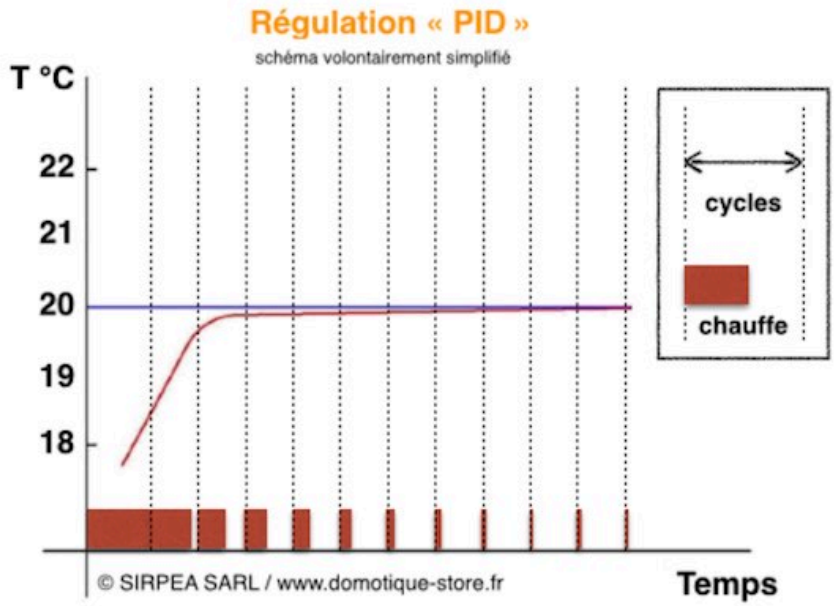
FUENTE: http://www.domoticadomestica.com/termostatos-inteligentes-control-pid/



Acceso remoto y termostato

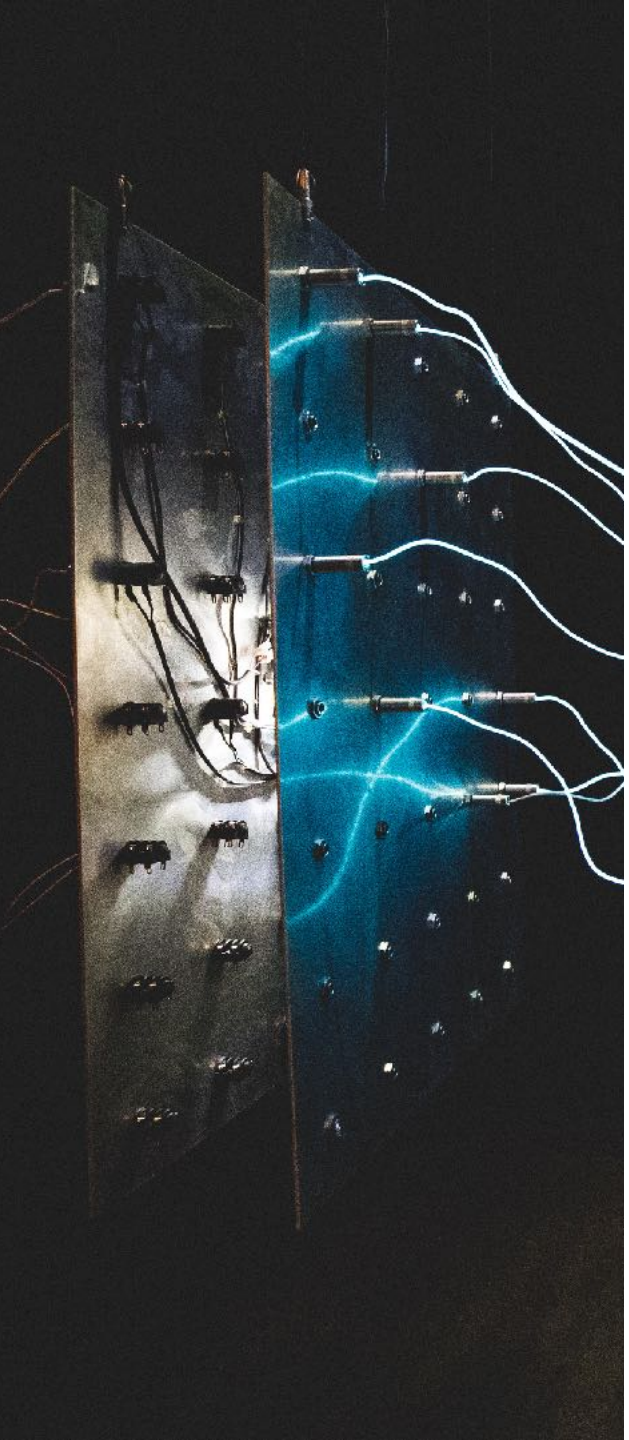
TERMOSTATO

simple-pid: https://pypi.org/project/simple-pid/



Control PID

FUENTE: http://www.domoticadomestica.com/termostatos-inteligentes-control-pid/



Acceso remoto y termostato

TERMOSTATO

Repositorio git: https://gitlab.com/hctpbl/heating-api

Ventajas e Inconvenientes

03

Ventajas

DINERO

El sistema hecho a mano es más barato

APRENDIZAJE

Se resuelven problemas de muy distintos tipos

PRIVACIDAD

Tus datos son tuyos y sólo tuyos

COLABORACIÓN

Puede que hagas algo que beneficie a la comunidad

Inconvenientes

TIEMPO

Hay que... vivir

INTERFAZ

Es muy difícil que logremos una interfaz tan atractiva como la de los sistemas comerciales

CARACTERÍSTICAS AVANZADAS

Algunas de las características son difíciles de imitar

INTEGRACIÓN

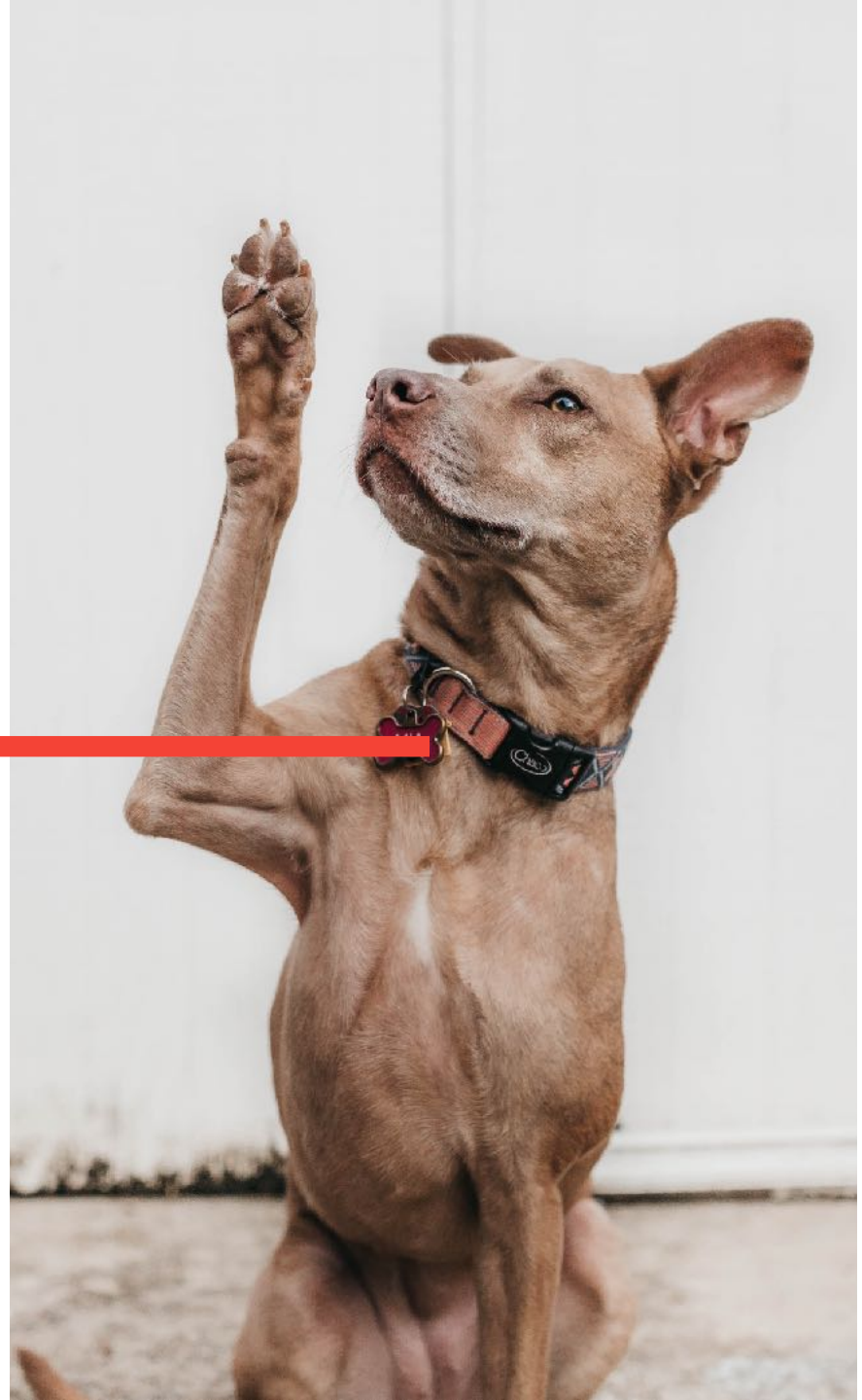
Google Home, Apple HomeKit.. Etc

CONCLUSIONES

Requiere mucho tiempo
Gran aprendizaje
Poder del software libre

Preguntas

04



Atribuciones

- Iconos: https://icofont.com/
- Icono caldera: Icons made by Freepik from www.flaticon.com
- Imagen transparencia 1: Image by Free-Photos from Pixabay
- Imagen transparencia 5: Photo by Moja Msanii on Unsplash
- Imagen transparencia 7: Photo by Matthew Henry on Unsplash
- Imagen transparencia 8: Photo by Louis Reed on Unsplash
- Imagen medición temperatura: Photo on Visualhunt.com
- Imagen almacenamiento temperatura: Photo by jesse orrico on Unsplash
- Imagen control de la caldera: Photo by Achudh Krishna on Unsplash
- Imagen transparencia 46: Image by ndemello from Pixabay
- Imagen transparencia 47: Photo by Mark Riechers on Unsplash
- Imagen acceso remoto y termostato: Photo by israel palacio on Unsplash
- Imagen transparencia 76: Image by Free-Photos from Pixabay
- Imágenes transparencia 77:
  - Dinero: Image by Harry Strauss from Pixabay
  - Aprendizaje: Photo by 🇸🇮 Janko Ferlič - @specialdaddy on Unsplash
  - Privacidad: Photo by Lianhao Qu on Unsplash
  - Colaboración: Photo by Matteo Vistocco on Unsplash
- Imágenes transparencia 78:
  - Tiempo: Photo by Andrik Langfield on Unsplash
  - Interfaz: Photo by Dan LeFebvre on Unsplash
  - Características Avanzadas: Photo by Franck V. on Unsplash
  - Integración: Photo by JJ Ying on Unsplash
- Imagen tranparencia 79: Photo by Kelly Sikkema on Unsplash
- Imagen transparencia 80: Photo by Camylla Battani on Unsplash