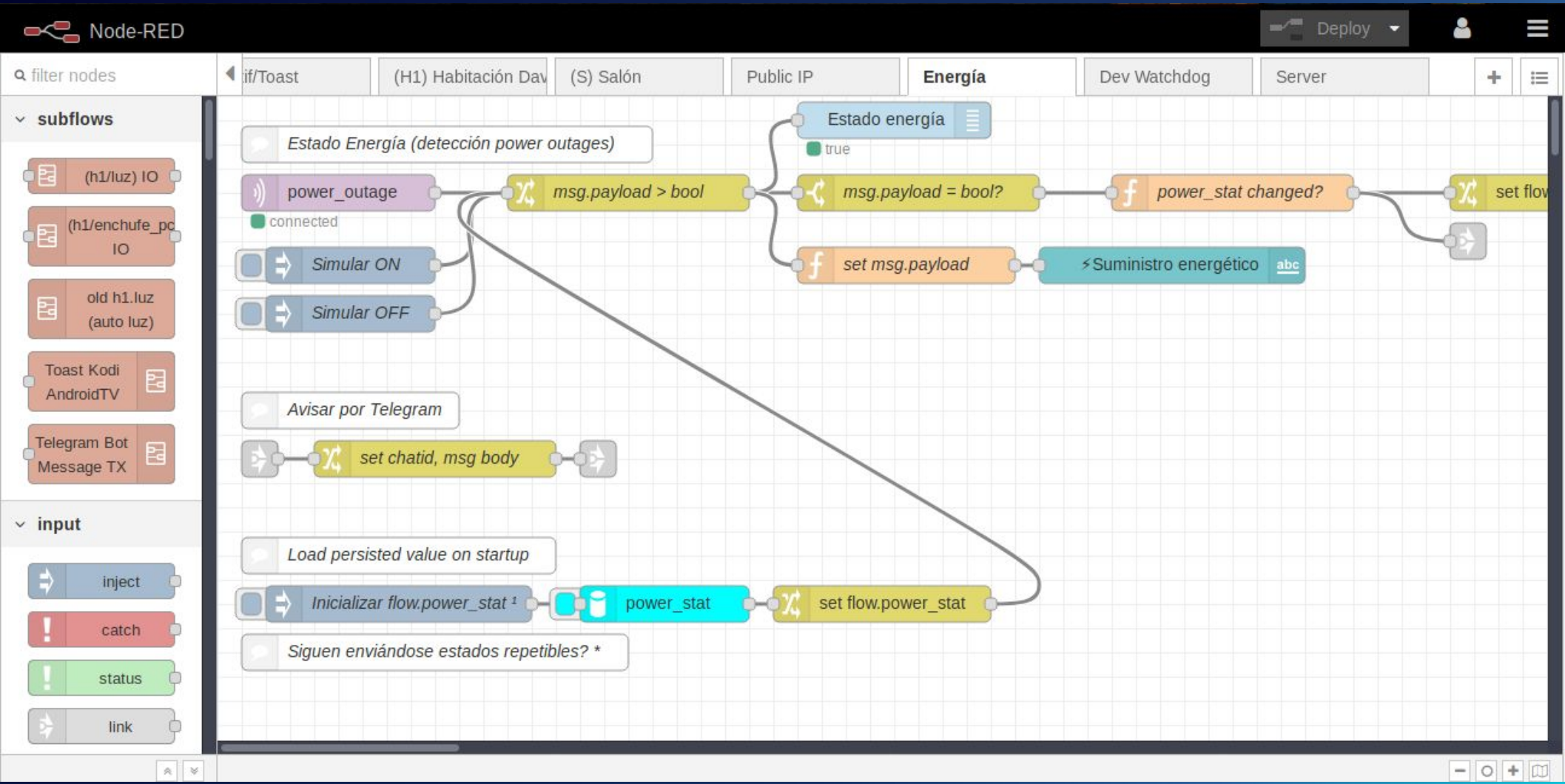
MQTT + Python

with paho-mqtt

Disclaimer

- Me centraré en MQTT para domótica
- Suelo usar Node-RED más que Python para la lógica de la domótica...


Node-RED
Deploy
filter nodes
if/Toast
(H1) Habitación Dav
(S) Salón
Public IP
Energía
Dev Watchdog
Server
subflows
(h1/luz) IO
(h1/enchufe_pc IO
old h1.luz (auto luz)
Toast Kodi AndroidTV
Telegram Bot Message TX
input
inject
catch
status
link
Estado Energía (detección power outages)
Estado energía
true
power_outage
connected
msg.payload > bool
msg.payload = bool?
power_stat changed?
set flow
Simular ON
Simular OFF
set msg.payload
⚡Suministro energético
abc
Avisar por Telegram
set chatid, msg body
Load persisted value on startup
Inicializar flow.power_stat ¹
power_stat
set flow.power_stat
Siguen enviándose estados repetibles? *

¿Qué es MQTT?

- Protocolo de mensajería publish/subscribe
- Especialmente diseñado para IoT
- Ligero, bajo consumo de recursos
- Centralizado

¿Qué es MQTT?

Protocolo de mensajería publish/subscribe

- Los clientes pueden:
  - Enviar mensajes (publicar)
  - Recibir mensajes (suscribir)
- Los mensajes se envían a un *topic*, no a un cliente
- Los mensajes se leen de un *topic*, no de un cliente

¿Qué es MQTT?

Especialmente diseñado para IoT
Ligero, bajo consumo de recursos

- Envío de mensajes en tiempo real, sin colas
- Poco consumo de recursos (hardware y red) en comparación con otros protocolos (p.ej. HTTP)

¿Qué es MQTT?

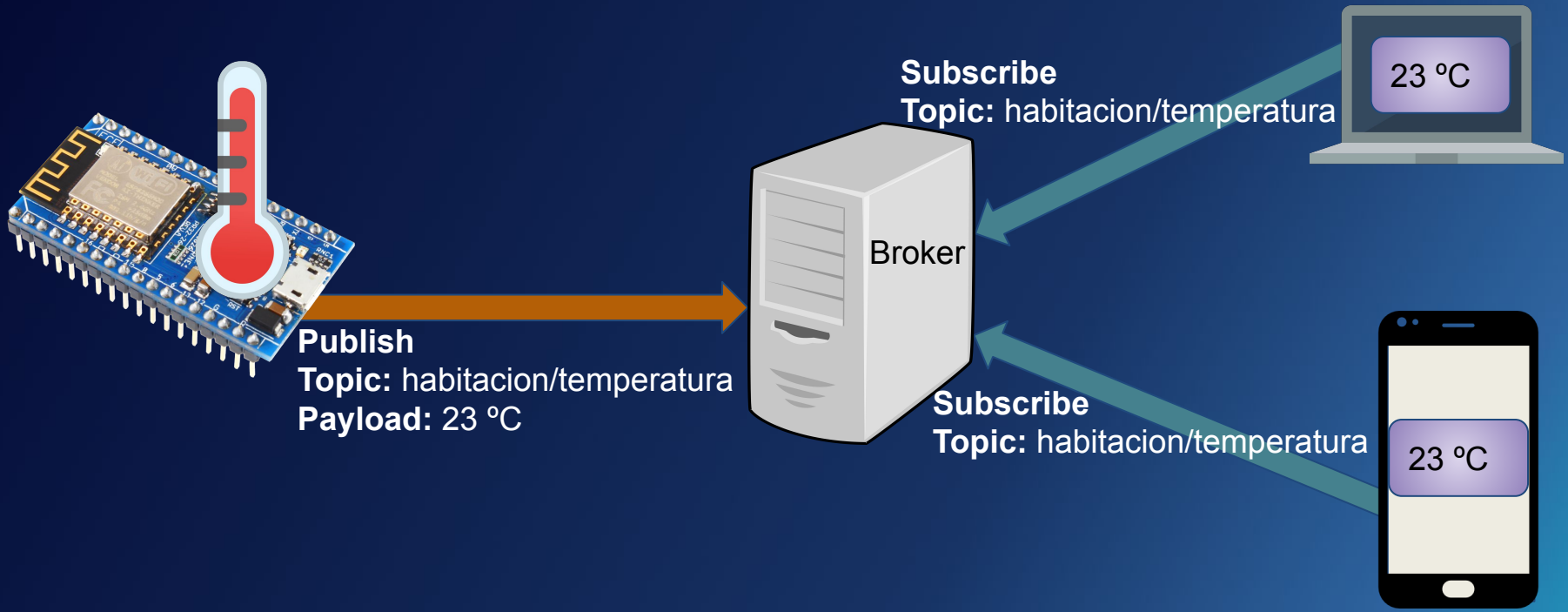
Centralizado

- Todos los mensajes pasan por el *broker* MQTT
- Clientes (publicadores y suscriptores) siempre se comunican con broker, nunca con otros clientes
- No necesitamos saber las direcciones de cada cliente, sólo del broker

- Instalación de broker MQTT (servidor):
  `sudo apt install` **`mosquitto`**
- Instalación de cliente MQTT:
  `sudo apt install` **`mosquitto-clients`**

¿Qué es MQTT?

Ejemplo de comunicación MQTT:



Topics

- Tipo de dato: string
- Límite: 65536 bytes
- Posibilidad de crear jerarquías con barras inclinadas **/**
- Suscriptores pueden usar wildcards:
  - **+** para un nivel
    ej: `casa/+/temperatura`, sustituyendo `+` a las habitaciones de la casa
  - **#** para uno/varios niveles (sólo al final)
    ej: `casa/#`, sustituyendo `#` a cualquier sub-nivel/es
  - Sólo **#** para suscribirnos a todos los topics

Topics

Formato de los topics y sus niveles

- ¿Libre elección?
- Estándar Homie

Mi elección (domótica):

`habitación/dispositivo/orden`
orden = cmd (comando), stat (estado)

Ej: `salon/luz/cmd` <- “ON”; `salon/luz/stat` -> “ON”

Payloads

- Cuerpo del mensaje
- Tipo de dato: String o binario
- Límite: 268,4 Megabytes
- Flag “retained”: mensaje permanece almacenado en broker para su topic (útil para estados)
  Si un cliente se suscribe tras su envío, lo recibirá

LWT (Last Will & Testament)

- Los clientes que se desconectan involuntariamente pueden enviar un mensaje LWT a cierto topic
- Este mensaje se almacena en el broker durante la conexión del cliente
- Cuando el broker detecta que el cliente cae, envía este mensaje LWT al topic designado
- No se enviará si el cliente se desconecta voluntariamente

Otros aspectos de interés...

...que no trataremos hoy

- Autenticación (user/pass)
- SSL
- QoS y clean session
- Websockets

Arquitectura recomendada

- Sensores/Actuadores (DIY)
  - Lógica mínima/nula
  - Principio KISS (Keep It Simple Stupid)
  - Se limitan a enviar lecturas y/o obedecer a comandos, mediante MQTT
- Backend de lógica
  - Programación de reglas sencillas o complejas
  - Interacción con sensores/actuadores/usuarios mediante MQTT
    - Python, NodeJS, Node-RED...
    - OpenHAB, HomeAssistant...

Recomendaciones personales

- Lógica y centralización domótica:
  Node-RED + Dashboard, HomeAssistant, OpenHAB
- Programas y servicios: Python
- Hardware y dispositivos:
  - Relés, interruptores Wifi: Sonoff + Tasmota
  - Dispositivos Zigbee: Xiaomi/Aqara
    Sniffer USB CC2531 + software zigbee2mqtt
  - DIY: ESP8266 (Wifi) + PubSubClient / Micropython

Librería Python: paho-mqtt

```
pip install -U paho-mqtt
```

```
import paho.mqtt.client as mqtt
client = mqtt.Client()

def on_connect(client, userdata, flags, rc):
    client.subscribe("pc/cmd")
    client.publish("pc/stat", "Estoy Online!")

def on_message(client, userdata, msg):
    print("Rx MQTT Message:")
    print("\tTopic:", msg.topic)
    print("\tPayload:", msg.payload.decode())

client.on_connect = on_connect
client.on_message = on_message

client.connect("127.0.0.1", 1883)
client.loop_forever()
```