

# GossetBo



Projecte Domòtic pel control d'accés  
i monitoratge de consums

José Antonio Álvarez Rodríguez  
DAM-2

# Index

## **1 Estudi de viabilitat.**

- [1.1. Establiment de l'abast del sistema.](#)
- [1.2. Estudi de la situació actual.](#)
- [1.3. Definició dels requisits del sistema.](#)
- [1.4. Estudi de les alternatives de solució.](#)
- [1.5 Selecció de la solució](#)

## **2 Anàlisi del sistema**

- [2.1 Definició del sistema.](#)
- [2.2 Establiment de requisits.](#)
- [2.3 Definició d'interfícies d'usuari.](#)
- [2.4 Especificació del pla de proves.](#)

## **3 Disseny del sistema**

- [3.1 Arquitectura del sistema](#)
  - [3.1.1 Definició de nivells d'arquitectura del sistema.](#)
  - [3.1.2 Especificació d'estàndards,normes de disseny i construcció.](#)
  - [3.1.3 Identificació de subsistemes.](#)
- [3.2 Revisió de casos d'ús](#)
  - [3.2.1. Revisió dels subsistemes segons els casos d'ús.](#)
  - [3.2.2. Elecció d'alternatives de components i llicències més adequades.](#)
  - [3.2.3. Especificacions de desenvolupament](#)
  - [3.2.4. Requisits d'implantació](#)
- [3.3 Anàlisi paradigma estructurat / Orientat a Objectes](#)

## 4 Desenvolupament

### 4.0 Planificació d'activitats

4.1 Programa de control de l'estat del pany de la porta.

4.2 Programa de mesurament del consum d'aigua.

4.3 Programa de mesurament del consum de llum.

4.4 Enviament de codi de desactivació des del telefon

    4.4.1 Programa amb Qml

    4.4.2 Programa amb Java i Android Studio.

4.5 Programa de control general principal

    4.5.1 GbGUI

    4.5.2 GbSecretTopic

    4.5.3 GbCamera

    4.5.4 GbMqtt

    4.5.5 GbControl

4.6 Creació del bot de Telegram i Script d'enviament de missatges

4.7 Instal·lació de broker Mqtt a la Raspberry

4.8 Instalacio d'InfluxDB i Grafana a la Raspberry

## 5 Implantació i proves

### 5.1 Implantació

    5.1.1 Implantació del sistema de control d'accés.

    5.1.2 Implantació del sistema de mesurament de consum d'aigua

### 5.2 Proves

    5.2.1 Proves inicials

    5.2.2 Proves reals

## 6 Conclusions i Enllaços

6.1. Conclusions

6.2. Enllaços

# 1

## Estudi de Viabilitat



## **1.1. Establiment de l'abast del sistema**

Aquest sistema vol ser un punt de partida per la meva part en l'aplicació de sensors, microcontroladors i microordinadors en un sistema domòtic.

Està pensat per a ser aplicat en qualsevol recinte del qual es vulgui millorar el control d'accés i el control dels consums d'aigua i electricitat, no per obtindre un sistema de gran fiabilitat, però amb l'anit de familiaritzar-me amb l'ús de llibreries i eines que més endavant pugui utilitzar en altres àmbits. Penso instal·lar-lo al meu habitatge per testejar la seva eficiència i possibilitats de millora.

## **1.2. Estudi de la situació actual**

De vegades quan surto de cassa he pensat que estaria bé tindre un sistema què m'envies un missatge al mòbil en cas d'haver-hi un problema, com un robatori, un incendi o inundació.

Per altre costat, també vaig pensar en què veig els consums d'electricitat i aigua quan arriba el rebut però no sóc conscient del consum en el dia a dia, atès què hauria d'anar mirant el comptador de casa meva i anar-hi calculant l'increment. Per això he pensat què estaria bé disposar d'una aplicació què em facilités aquests monitoratges.

En el mercat es poden trobar moltes càmeres de vigilància i sistemes què poden ser visualitzats pel mòbil, però jo volia implementar el meu sistema amb components barats, de fàcil adquisició, reutilitzables i què puguin oferir usos en paral·lel al control d'accés.

En quan al monitoratge de consums, també hi ha al mercat molts dispositius domòtics amb la possibilitat de ser consultats via mòbil o navegador, però aquests sistemes son caixes negres què necessiten d'un instal·lador especialitzat, a més la seva configuració acostuma a ser incomoda pels usuaris en general per no parlar del seu preu i manteniment.

### **1.3. Definició dels requisits del sistema**

Pel control d'accés vull què el sistema quedi armat quan tanco el pany de la porta amb clau i què sigui necessari enviar un codi des del mòbil per a desarmar el sistema.

En cas de què s'obi el pany de la porta sense haver-hi enviat el codi de desactivació, vull que el sistema m'envii un missatge d'algun tipus al mòbil i activi una càmera que pugui veure igualment pel mòbil.

En el cas dels consums d'electricitat i aigua, l'objectiu es poder monitorar-los en lapses de temps seleccionables i què es proporcionin estadístiques d'aquests consums en aquells lapses.

El cost total del sistema no ha de superar els 100 € i el temps d'implantació no superar els tres dies. A més vull que el sistema no depengui de tipus concrets de sensors ni de plataforma en el cas del software.

També vull que sigui de codi obert i que les seves parts puguin ser utilitzades per separat en altres tipus d'usos sense o amb petites modificacions així com què les funcionalitats del sistema siguin fàcilment extensibles.

### **1.4. Estudi de les alternatives de solució**

Com a possibles solucions he contemplat les següents tres possibilitats:

1. Connectar sensors a un ordinador de tipus torre per mig de cablejat i monitorar la informació emmagatzemada en fitxers remotament.
2. Utilitzar microordinadors dedicats per a cada sensor connectat a ell per cablejat i enviar la informació a una base de dades central a un ordinador convencional per algun mitjà sense fils.
3. Utilitzar plaques controladores connectades per cablejat als sensors i enviar la informació per algun mitjà sense fils a un microordinador central al que pugui accedir remotament.

## **1.5. Selecció de la solució**

Per seleccionar la millor possibilitat entre les anteriors he pensat en els pros i contres de cadascuna d'elles.

### **Solució 1:**

Avantatges:

- El cablejat es fiable i no requereix de cap altre mitjà que consumeixi energia ni altre tipus de recurs.
- La implementació es fàcil, no es requereix de cap interfície ni protocol addicional.
- Requereix menys components.
- La transmissió és generalment més ràpida que en els mitjans sense fils.

Inconvenients:

- La instal·lació es lenta i de difícil modificació.
- Antiestètic i incomode.
- L'ordinador de tipus torre no pot ubicar-se en llocs petits, consumeix molta energia i té un sistema de connexions per cable de difícil accés.
- La gestió dels fitxers i el filtratge de les dades es farà mes i mes complex amb el temps.

### **Solució 2:**

Avantatges:

- Al disposar de sistema operatiu, els microordinadors es poden configurar en una xarxa virtual per formar un cluster.
- Es pot utilitzar qualsevol llenguatge de programació per fer programes de gestió de les dades.
- Se'ls pot connectar una pantalla.

Inconvenients:

- El preu pot arribar a ser considerable.
- El consum d'energia pot arribar a ser alt.
- Malbaratament de recursos de processament i memòria.

### **Solució 3:**

Avantatges:

- Cost baix.
- Consum de energia baix.
- Mida reduïda.

Inconvenients:

- Llenguatges de programació limitats.
- Sense sistema operatiu.
- Llibreries no disponibles segons el model.

Donats els pros i contres, vaig seleccionar la tercera solució, perquè crec que és la que millor s'ajusta a les necessitat del sistema que vull implementar sense malbaratar recursos.

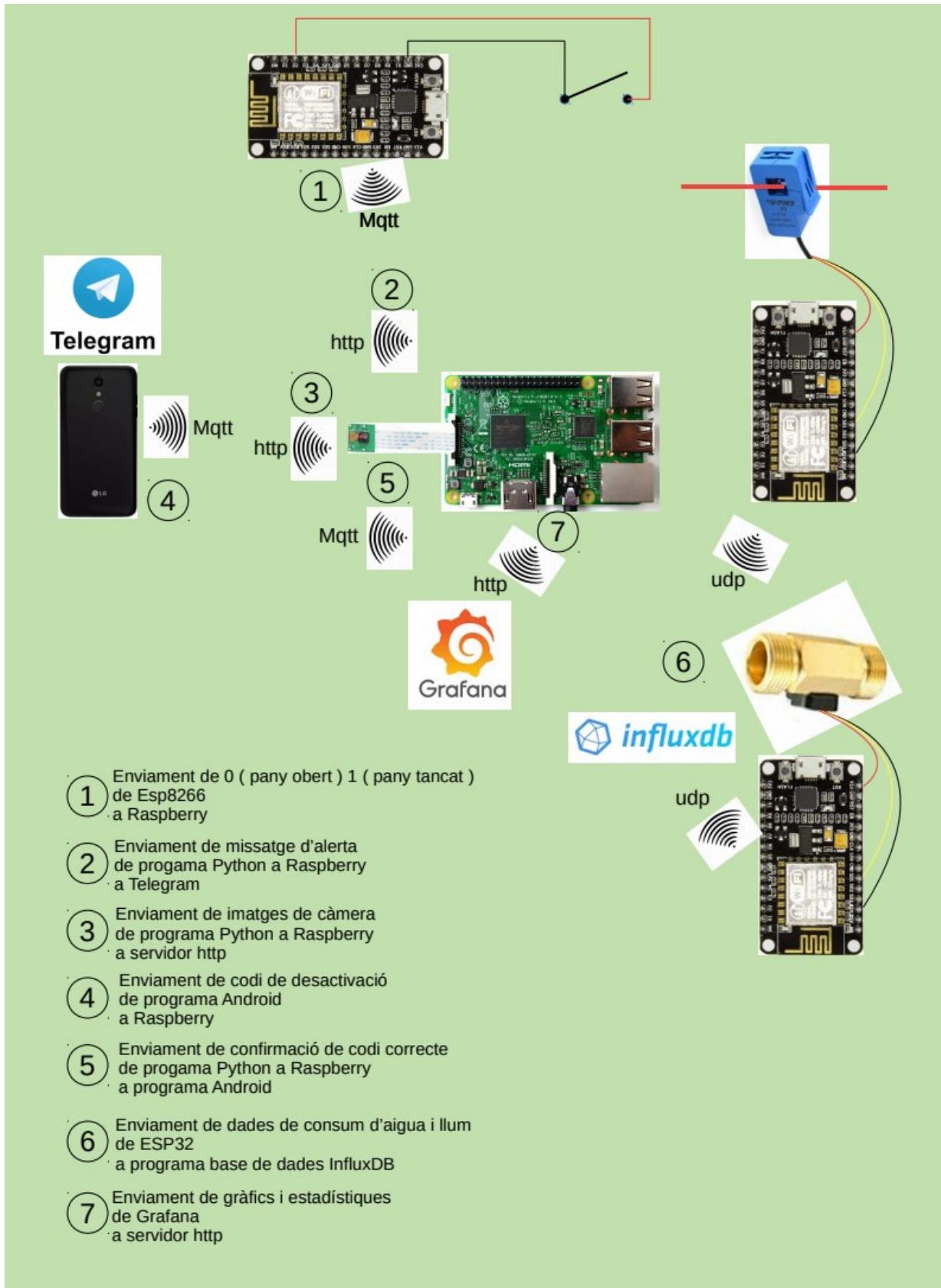
## 2.

# Anàlisi del sistema



## 2.1 Definició del sistema

Esquema del sistema previst



## 2.2 Establiment de requisits

Per assolir les tasques indicades al gràfic anterior necessitaré els següents programes i capacitats.

- Programa C++ per a esp8266 amb les funcions:

(1)

- Lectura del estat d'un botó
- Connexió a la xarxa WiFi
- Connexió a broker Mqtt
- Enviament del estat del botó al broker

(2)

- Creació de Bot de Telegram
- Script de Bash amb les dades del Bot, el missatge a enviar i una sentencia de curl per a enviar el missatge a la URL de Telegram.
- Mòdul de Python què executi el script anterior.

(3)

- Mòdul de Python amb les funcions:
  - Creació de servidor http.
  - Activament i enregistrament d'imatges de càmera.
  - Apertura de finestra de navegador web per a mostrar les imatges.
  - Tancament de la finestra de navegador web.
  - Desactivació de la càmera.

(4)

- Programa Android amb les funcions:
  - Connexió a broker Mqtt.
  - Lectura de codi introduït pel usuari.
  - Enviament de codi a broker.
  - Lectura de missatge de confirmació de codi des del broker.
  - Mostrar resultat al usuari.

(5)

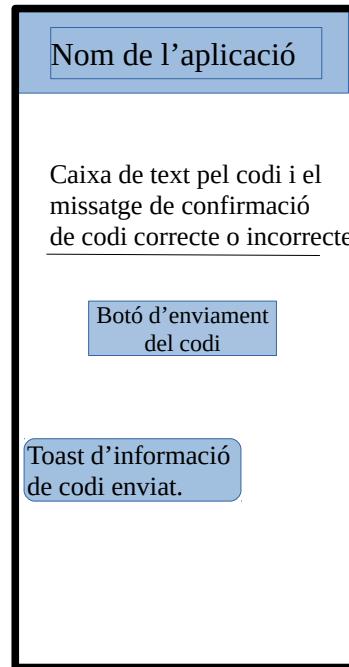
- Mòdul de Python amb les funcions:
  - Lectura del codi de desactivació establert des de arxiu.
  - Comparació del missatge rebut al broker amb l'establert.
  - Enviament al broker del resultat de la comparació.
  - Possibilitar el canvi de codi de desactivació establert.

- Instal·lació del servei influxDB a Raspberry i configurar-lo per rebre paquets UDP.
- Creació de les bases de dades.
- Programa C++ per a ESP32 amb les següents funcions:
  - Connexió a xarxa WIFI
  - Lectura del voltatge al pin analògic de la esp32
  - Conversió del voltatge llegit a Watts
  - Introduir dades al final de una cua de floats
  - Treure dades del principi de la cua de floats
  - Enviament de paquets UDP amb el format correcte per a ser llegit com a registre a la base de dades influxDB
- Programa C++ per a ESP32 semblant al anterior però en lloc de llegir del pin analògic, llegir la quantitat de pulsos enviats pel sensor i convertir-los en litres.

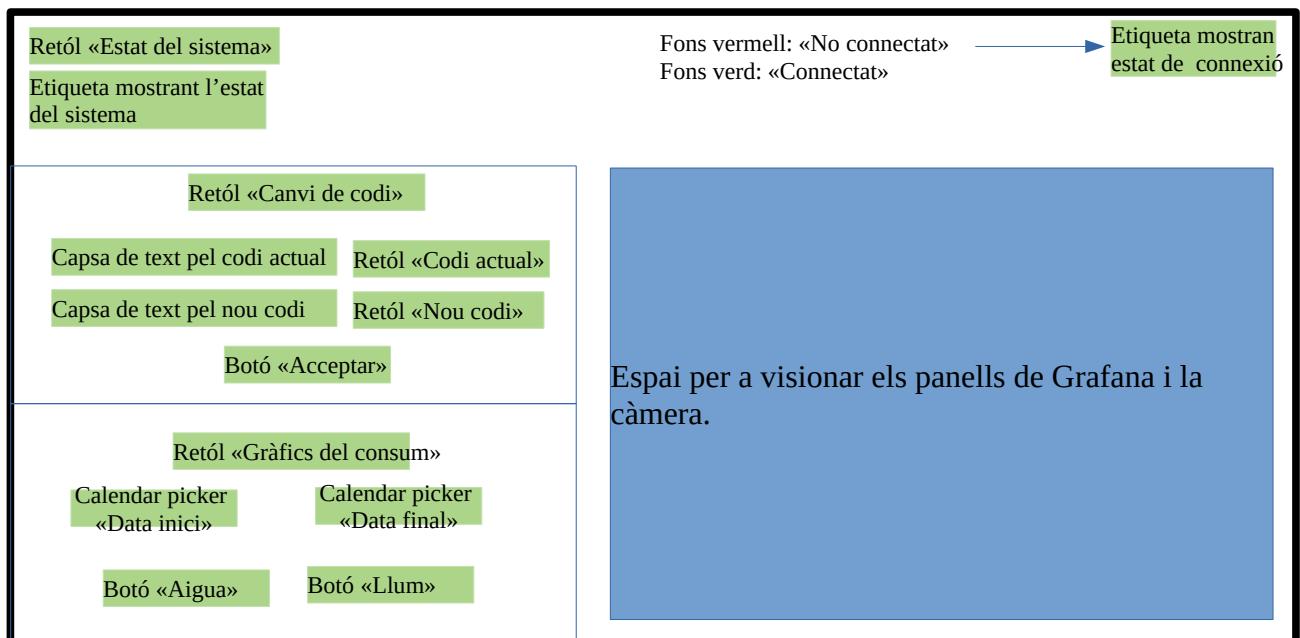
- 
- Instal·lació del servei de Grafana a Raspberry.
  - Connexió de Grafana amb les bases de dades de InfluxDB
  - Creació dels panells pels gràfics i configuració de les dades.
  - Mòdul de Python què obri una finestra de navegador i mostri els panells.

## 2.3 Definició d'interfícies d'usuari

Interfície pel mòbil :



Interfície per Raspberry:



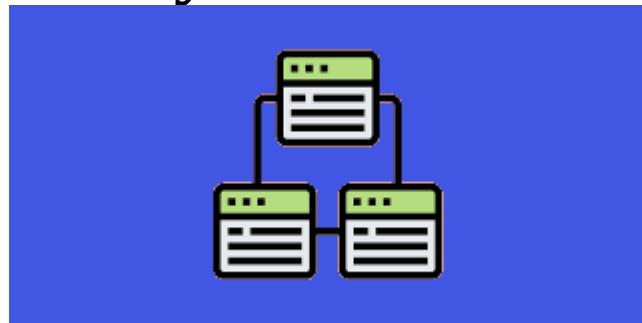
## 2.4 Especificació del pla de proves

El pla de proves està format per a proves de quatre tipus:

- **Proves de funcions i mètodes:** Consistents en provar cada funció o mètode per separat i comprovar que fa el què te que fer, portant els valors dels paràmetres a extrems de màxims, mínims, valors aleatoris i captura de excepcions.
- **Proves de mòduls:** Es provarà cada mòdul per separat fent crides a les seves funcions i mètodes i es farà una revisió dels possibles efectes secundaris no esperats.
- **Proves de programa:** Es comprovarà el funcionament del programa després d'integrar cadascun dels mòduls que es vagin afegint.
- **Proves de calibratge dels sensors:** Per a establir què els valors de les constants per a la conversió de les senyals dels sensor en les lectures finals són el més precises possibles. Per exemple, per traduir nombre de pulsos per litre d'aigua o voltatge en el pin analògic a Watts / hora i intensitat.

# 3.

## Disseny\$ del sistema



## 3.1 Arquitectura del sistema

### 3.1.1 Nivells d'arquitectura

El sistema estarà compost per tres nivells:

- Nivell de client: Programes amb interfície gràfica per a interactuar amb l'usuari i processar i mostrar els resultats.
- Nivell d'adquisició i transmissió de dades: Programes per a la gestió dels sensors i la transmissió de les dades d'ells obtingudes al nivell d'emmagatzematge i processament.
- Nivell d'emmagatzematge: Bases de dades i gestors de fitxers pel us de les capes d'adquisició i client.

### 3.1.2 Especificació d'estàndards i normes

En tots els programes utilitzats en l'aplicació, es seguiran al màxim els principis S.O.L.I.D.:

- **Single responsibility:** Cada classe, mòdul , mètode o funció s'encarregará d'una única funcionalitat.
- **Open-Close:** Serà possible d'estendre els mòduls i classes, per exemple afegint nous atributs per mig d'herència, però les classes i mòduls originals no es modificaran. D'aquesta manera que les possibles evolucions de l'aplicació no implicarà fer canvis en la totalitat i es farà més fàcil.
- **Liskow-Substitution:** En cas de crear classes derivades de les originals, serà possible substituir les classes base per aquestes sense alterar el funcionament correcte de l'aplicació.
- **Interface segregation principle:** Escissió de les interfícies que les classes esposen per a ser usades per d'altres, en interfícies més petites, de manera que les classes què les usen, no més tinguin coneixement dels mètodes i funcions que realment necessiten.
- **Dependency inversion principle:** Els mòduls d'alt nivell no han de dependre dels de baix nivell. Tots dos han de dependre d'abstraccions, com interfícies. Les abstraccions no han de dependre dels detalls d'implementació, sinó els detalls d'implementació dependre de les abstraccions.

### 3.1.3 Identificació de subsistemes

Per els programes de gestió dels sensors, s'identifiquen clarament tres subsistemes:

- Funcions per a la connexió a la xarxa i transmissió de dades per udp a la base de dades influxDB.
- Funcions per a la connexió , subscripció a topics i enviament i rebuda de missatges a broker mqtt.
- Funcions de gestió específica del sensor en concret.

Els dos primers subsistemes es poden utilitzar indistintament per a qualsevol sensor, de manera que val la pena pensar en fer llibreries amb ells.

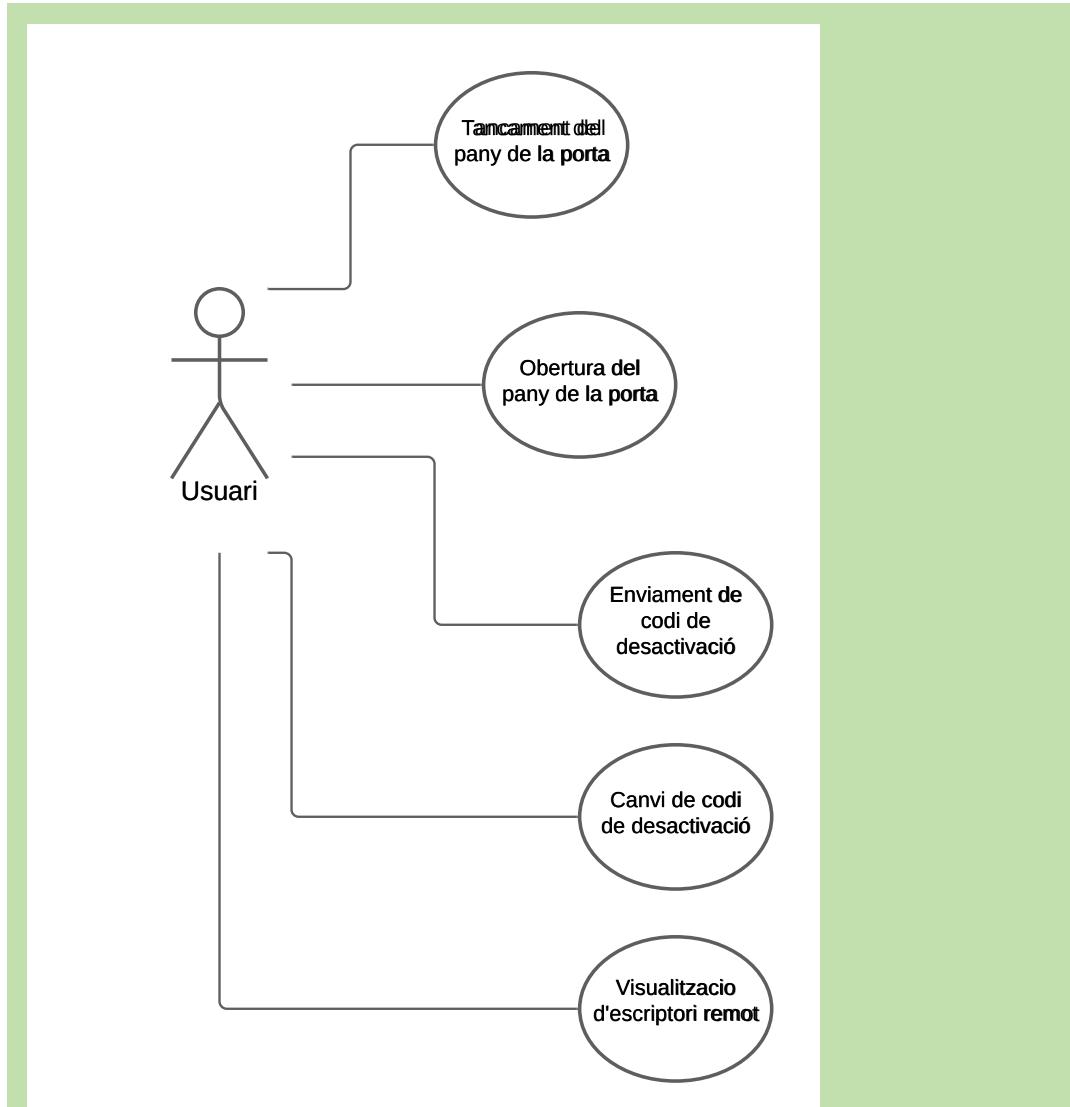
En quan la programa central, també s'identifiquen els dos primers subsistemes anteriors i també:

- Enviament de imatges de la càmera a servidor http.
- Enviament de missatges a Telegram.
- Interfície gràfica d'usuari.
- Gestió de fitxers de configuració.

## 3.2 Revisió de casos d'us

### 3.2.1 Revisió dels subsistemes segons els casos d'ús.

El diagrama de casos d'us és el següent:



Els quatre primers casos d'us, queden coberts pels subsistemes de connexió a la xarxa i connexió a mqtt.

L'últim queda cobert per a la resta de subsistemes.

Segons aquest diagrama, no es necessiten subsistemes addicionals.

### 3.2.2 Elecció d'alternatives de components i llicències més adequades.

#### Transmissió d'esdeveniments entre programes

Vaig pensar a les següents alternatives:

- Protocol TCP
- Protocol UDP
- Radiofreqüència ( LORA )
- Protocol Mqtt

Vaig escollir Mqtt per les següents raons:

- És un protocol lleuger, adequat pel us en dispositius IoT de baixa potència.
- Fàcil d'implementar.
- Es senzill crear un servidor d'us privat.
- No requereix de dispositius addicionals.
- L'hem utilitzat molt aquest curs i em resulta molt familiar.

#### Emmagatzematge de dades

Les alternatives eren:

- SQL ( MySql )
- XML
- Seriació
- Firebase
- InfluxDB

Vaig decantar-me per InfluxDB per:

- Puc transmetre les dades directament des les plaques de desenvolupament.
- Es fàcil d'instal·lar com un servei.
- Les querys SQL no són gaire diferents d'altres bbdd.
- Més ràpid que una bbdd convencional.
- No requereix de codi extra.
- Posa una marca temporal de manera automàtica quan es desa una dada.
- Es comunica amb Grafana fàcilment.

## Representació gràfica de les dades

Vaig escollir Grafana entre:

- Fer un gràfic propi utilitzant un Canvas
- Apache Zeppelin
- Matplotlib

Perquè:

- S'instal·la fàcilment com a servei.
- No requereix codi addicional.
- Es comunica fàcilment amb influxDB i altres fonts de dades.
- Es fàcil d'utilitzar.
- Es pot consultar des de qualsevol lloc per internet.

Tots aquests components són gratuïts i de llicència open source.

En el programa de control, he utilitzat les llibreries següents com a components:

Nom del component	Mòdul d'ús	Llicència	Descripció	Enllaç
os	GbControl	Lliure ( PSFL )	Possibilita l'ús de funcionalitats del sistema operatiu. En aquest cas he utilitzat la funció <b>System</b> per a executar el script de Bash per enviar missatge al bot de Telegram.	<a href="https://docs.python.org/3/library/os.html">https://docs.python.org/3/library/os.html</a>
subprocess	GbControl GbGUI	Lliure ( PSFL )	Permet executar codi en un procés diferent. He utilitzat la funció <b>Popen</b> per a executar el servidor http i el navegador en altres processos.	<a href="https://docs.python.org/3/library/subprocess.html">https://docs.python.org/3/library/subprocess.html</a>

Nom del component	Mòdul d'ús	Llicència	Descripció	Enllaç
Io	GbCamera	Lliure ( PSFL )	Proporciona facilitats pel maneig de diferents tipus de I/O. L'utilitza la classe StreamingOutput de GbCamera amb <b>BytesIO</b> per a crear un buffer de bytes a memòria.	<a href="https://docs.python.org/3/library/io.html">https://docs.python.org/3/library/io.html</a>
picamera	GbCamera	BSD	Exposa un interície per a interactuar amb la càmera de la Raspberry. Es crea un objecte d'aquest tipus i s'utilitzen els seus mètodes: <b>rotation, start_recording i stop_recording.</b>	<a href="https://picamera.readthedocs.io/en/release-1.13/">https://picamera.readthedocs.io/en/release-1.13/</a>
logging	GbCamera	Lliure ( PSFL )	Proporciona esdeveniments de log per tots els mòduls de Python. La classe StreamingHandler utilitza la funció <b>warning</b> per a mostrar un missatge d'avís en cas d'excepció.	<a href="https://docs.python.org/3/library/logging.html">https://docs.python.org/3/library/logging.html</a>

Nom del component	Mòdul d'ús	Llicència	Descripció	Enllaç
socketserver	GbCamera	Lliure ( PSFL )	Simplifica la tasca de creació de servidors de xarxa. L'utilitza la classe StreamingServer com a classe base.	<a href="https://docs.python.org/3/library/socketserver.html">https://docs.python.org/3/library/socketserver.html</a>
threading	GbCamera	Lliure ( PSFL )	Construeix una intereficie per fils d'alt nivell sobre la de baix nivell thread. S'utilitza un objecte de tipus Condition a la classe StreamingOutput amb el seu mètode <b>notify_all</b> per a notificar l'accés a un buffer de bytes a la resta de fils.	<a href="https://docs.python.org/3/library/threading.html">https://docs.python.org/3/library/threading.html</a>
http	GbCamera	Lliure ( PSFL )	Recopila varis mòduls per a treballar amb el protocol http. La classe StreamingHandler utilitza la classe <b>BaseHTTPRequestHandler</b> del mòdul <b>server</b> com a classe base.	<a href="https://docs.python.org/3/library/http.html">https://docs.python.org/3/library/http.html</a>

Nom del component	Mòdul d'ús	Llicència	Descripció	Enllaç
tkinter	GbGUI	Lliure ( PSFL )	És el <b>GUI toolkit</b> estàndard per Python.	<a href="https://docs.python.org/3/library/tkinter.html">https://docs.python.org/3/library/tkinter.html</a>
datetime	GbGUI	Lliure ( PSFL )	Proporciona classes per el maneig de temps i dates. S'utilitza la funció <b>strptime</b> per a passar de dates en format dd-mm-YY a unix time.	<a href="https://docs.python.org/3/library/datetime.html">https://docs.python.org/3/library/datetime.html</a>
paho	GbMqtt	Eclipse public lisence	Proporciona un client per interactuar amb un broker mqtt. L'utilitza GbMqtt com a classe base.	<a href="https://pypi.org/project/paho-mqtt/">https://pypi.org/project/paho-mqtt/</a>
configparser	GbSecretTopic	Lliure ( PSFL )	Proporciona un lenguaje bàsic per a treballar amb arxius de configuració. S'utilitzen els mètodes <b>read</b> , <b>write</b> i <b>set</b> d'un objecte d'aquesta classe per a gestionar l'arxiu de configuració del programa.	<a href="https://docs.python.org/3/library/configparser.html">https://docs.python.org/3/library/configparser.html</a>

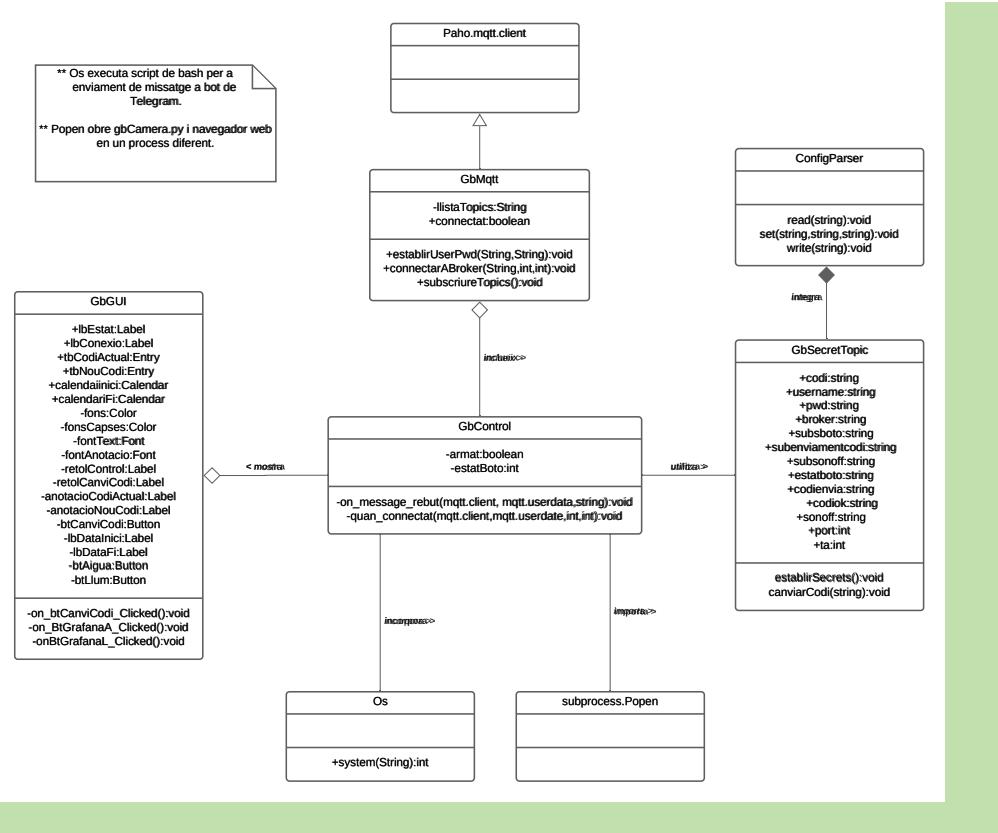
En el programes de control dels sensors, he utilitzat el següents components:

Nom del component	Mòdul d'ús	Llicència	Descripció	Enllaç
SimpleTimer.h	PulsacioBoto.ino	GNU LGPL	Serveix per a llençar accions cada vegada que passa un lapse de temps establert. L'he usat a la esp8266 peque en aquesta placa FreeRTOS requereix de la instal·lació de llibreries addicionals. Utilitzo els mètodes <b>enable</b> , <b>disable</b> , <b>setInterval</b> i <b>run</b> d'un objecte SimpleTimer.	<a href="https://playground.arduino.cc/Code/SimpleTimer/">https://playground.arduino.cc/Code/SimpleTimer/</a>
ESP8266WiFi.h	PulsacioBoto.ino	GNU LGPL	Per a connectar la placa esp8266 a internet. He utilitzat un objecte de la classe WiFiClient per pasar-lo com a argument en la construcció del objecte PubSubClient, i les funcions <b>begin</b> , <b>status</b> i <b>localIP</b> de la classe WiFi	<a href="https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#client">https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#client</a>
PubSubClient.h	PulsacioBoto.ino	MIT lisence	Proveeix un client per a fer publicacions i subscripcions en un servidor que proveeixi MQTT	<a href="https://pubsubclient.knolleary.net/api">https://pubsubclient.knolleary.net/api</a>

Nom del component	Mòdul d'ús	Llicència	Descripció	Enllaç
WiFi.h	MedicioAigua.ino MedicioLlum.ino	GNU	Permet la connexió de la placa ESP32 a internet. Utilitza les funcions <b>begin</b> , <b>status</b> i <b>localIP</b>	<a href="https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h">https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h</a>
WiFiUdp.h	MedicioAigua.ino MedicioLlum.ino	GNU	Servei per a enviar i rebre packets udp a un host en un port determinat. Utilitza els mètodes <b>beginPacket</b> , <b>print</b> i <b>endPacket</b> d'un objecte d'aquesta classe.	<a href="https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiUdp.h">https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiUdp.h</a>

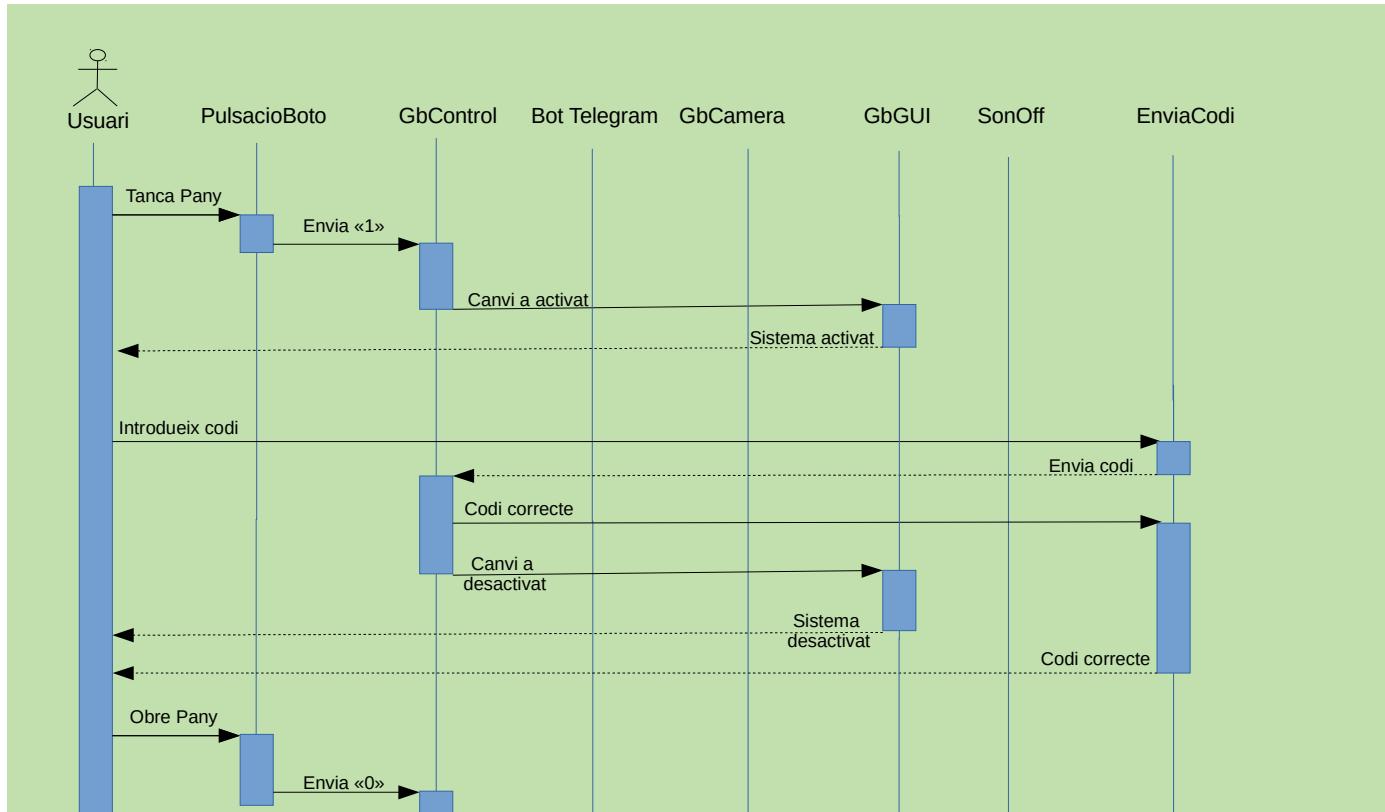
### 3.2.3 Especificacions de desenvolupament

Diagrama UML de GbControl:



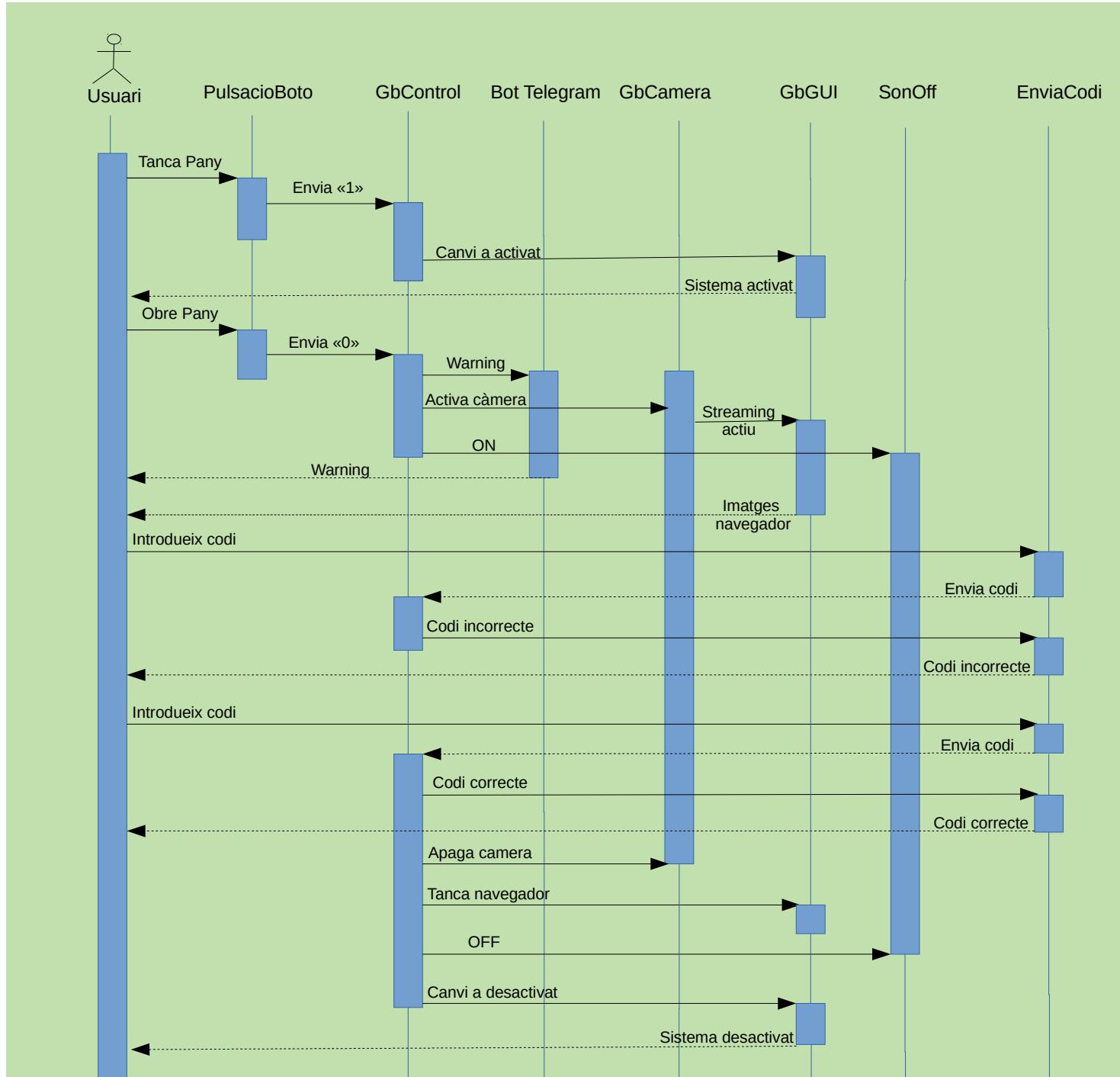
### Diagrama de seqüència quan:

L'usuari tanca el pany de la porta, introduceix el codi de desactivació correcte i obre el pany de la porta.



### Diagrama de seqüència quan:

L'usuari tanca el pany de la porta, obre de nou sense introduir el codi , introduceix un codi de desactivació erroni i finalment introduceix el codi correcte.



### 3.2.4 Requisits d'implantació

Hardware:

- 1 Raspberry Pi 3 o superior.
- 1 Alimentador per a Raspberry
- 1 Raspberry camera.
- 1 NodeMCU esp8266
- 2 NodeMCU esp32
- 3 Protoboard
- 3 Alimentadors 5V
- 1 Sensor de corrent SCT-013 30A
- 1 Sensor de caudal.
- Telefon mobil.
- 1 Sonoff

Sofware:

- Raspberry os.
- Python 3
- Broker Mqtt
- InfluxDb
- Grafana

Altres:

- Compte a ZeroTyer
- Compte i Bot a Telegram

### 3.3 Anàlisi paradigma estructurat / Orientat a Objectes

Pel programa principal he escollit el llenguatge de programació Python, el qual permet l'ús dels paradigmes Orientat a objectes i estructurat.

A continuació s'indiquen els paradigmes utilitzats en cada mòdul:

**GbControl:** OO. Aquesta classe, utilitzarà un objecte mqtt client i definirà la seva funció on\_message, per això, al tindre que contenir objectes com a atributs, he utilitzat aquest paradigma.

**GbCamera:** OO. Aquesta classe utilitza varies subclasses que fan us de la herència, per això es necessari utilitzar aquest paradigma.

**GbGUI:** OO. Aquesta classe usa widgets de la classe Tkinter. Aquests widgets són objectes d'aquest paradigma.

**GbMqtt:** OO. Aquesta classe hereta de paho.mqtt.Client la qual treballa amb aquest paradigma.

**GbSecretTopic:** Estructurat. Com el programa no més farà us d'un arxiu de configuració, utilitzaré mètodes del mòdul al estil estructurat.

**MedicioAigua i MedicioLlum:** OO i Estructurat: Aquests programes utilitzaran objectes al fitxer .ino i fitxers escrits en C, el qual utilitza el paradigma estructurat.

**PulsacióBoto:** OO. Aquest programa no utilitzarà mòduls de C, pel que no més s'utilitzarà el paradigma principal de C++.

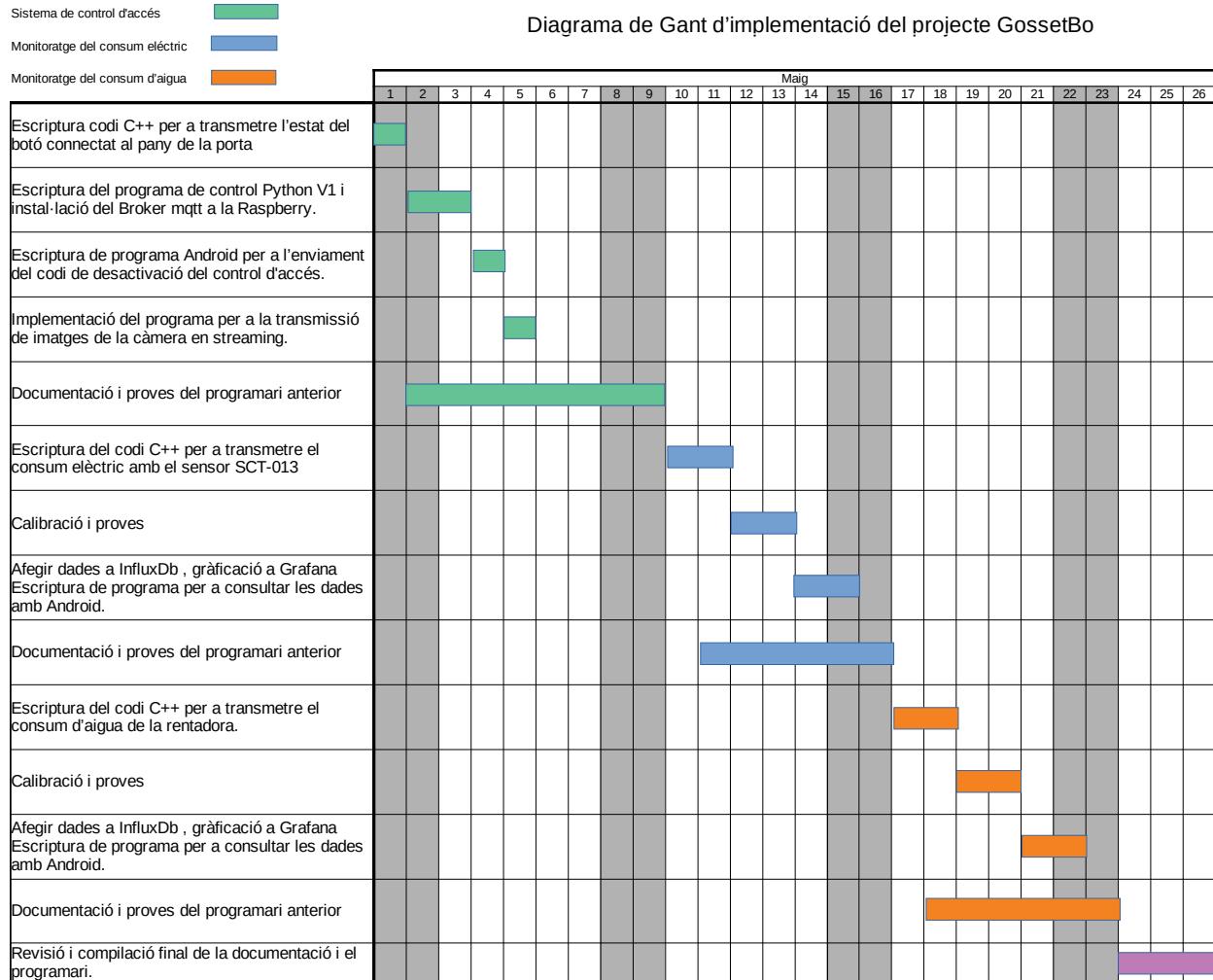
# 4

## Desenvolupament.



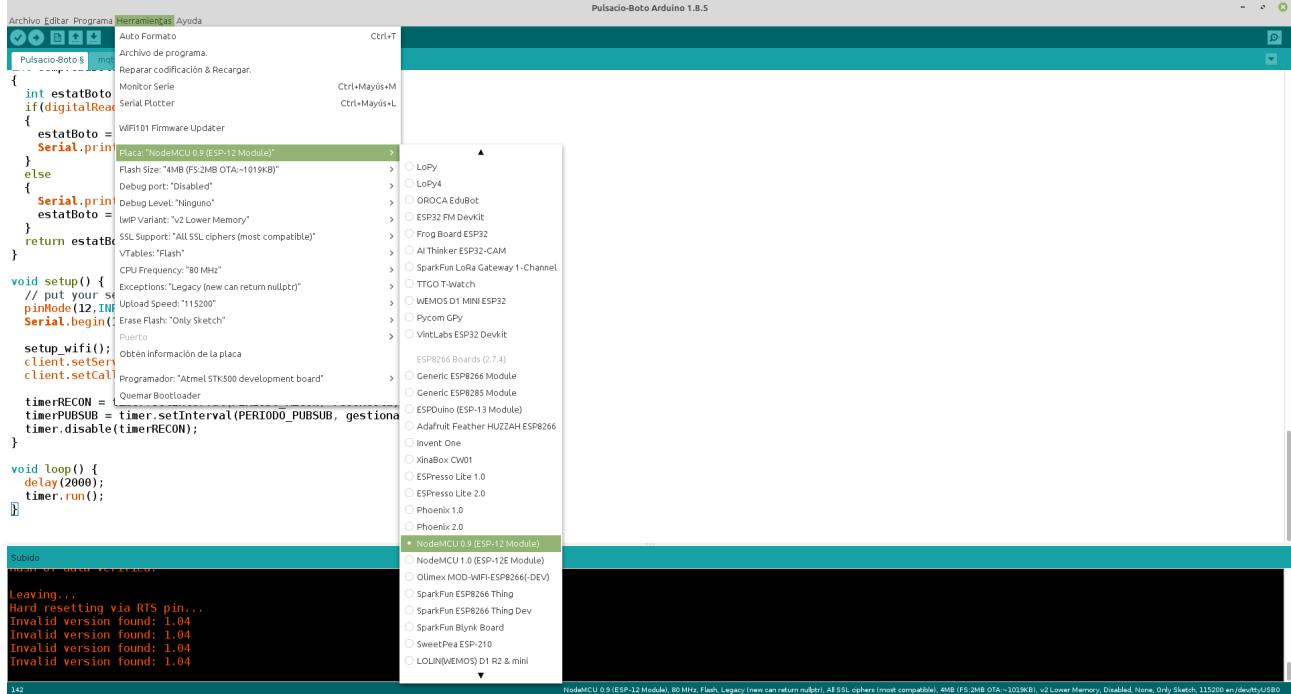
## 4.0 Planificació d'activitats

Per a veure com anava progresant en el desenvolupament de l'aplicació, les proves i documentació, he creat aquest diagrama de Gant



## 4.1 Controlar estat del pany de la porta (Programa Pulsacio-Boto)

He configurat el IDE d'Arduino per a la placa esp8266 seleccionant a, Eines → Placa, la placa NodeMCU 0.9



El fitxer **Pulsacio\_const.h** conté totes les constants utilitzades al programa



El programa comença amb la inclusió de les llibreries necessàries pel seu funcionament i la declaració dels objectes: WiFiClient per a la connexió a la xarxa, PubSubClient per al broker mqtt i el Timer per cridar a les tasques de manera periòdica.

També hi ha tres variables int, dues per a d'identificació de les tasques i un altre per monitoratge del estat del botó.

```
#include "Pulsacio_const.h"
#include <SimpleTimer.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Declaració d'objectes WiFi
WiFiClient espClient;
PubSubClient client(espClient);
SimpleTimer timer; //Timer per executar les tasques periòdiques

int timerPUBSUB, timerRECON; //Id's de les tasques periòdiques del timer
int estatBoto = 0; //Per a monitorar el estat del botó
```

La funció setup, com és habitual , fa la feina d'inicialització. En primer lloc, declara el pin 12 com de lectura em mode pull up , inicialitza els objectes WiFiClient i PubSubClient i crida a la funció **setup\_wifi()**

```
///////////
// Inicialització del pin de lectura, el wifi , el client mqtt
// i el Timer
///////////
// void setup() {
// Asignació del pin 12 com de lectura amb mode pull up
pinMode(12, INPUT_PULLUP);
// Inicialització del wifi i del client mqtt
Serial.begin(115200);
setup_wifi();
client.setServer(mqtt_server, 1883);
client.setCallback(incomingSub);

//Asignació de les tasques periódiques i els seus temps al timer
timerRECON = timer.setInterval(PERIODO_RECON, reconecta);
timerPUBSUB = timer.setInterval(PERIODO_PUBSUB, gestionaPubSub);
//Desactivar la tasca de reconexió una vegada connectat
timer.disable(timerRECON);
}
```

La funció setup\_wifi() inicialitza l'objecte WiFiClient i conté un bucle while que mostra una seqüència de punts mentre es realitza la connexió a la xarxa. Una vegada feta la connexió, es mostra un missatge.

```
///////////
// Conexió a la xarxa WiFi
///////////
// void setup_wifi()
{
  delay(10);
  Serial.println();
  Serial.print("Conexió a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  Serial.println("");
  Serial.println("Conectat a WiFi");
  Serial.println("Direcció IP: ");
  Serial.println(WiFi.localIP());
}
// Fi setup_wifi
```

Una vegada s'ha establert la connexió a la xarxa wifi, la funció setup inicialitza l'objecte PubSubClient amb el broker i el port. També li assigna la funció **incomingSub** com a callback per al event de rebuda d'un missatge. Aquesta funció, no té cap utilitat actualment, però he pensat a implementar-la com a previsió per futures possibles aplicacions.

De moment, es limita a mostrar el topic i el missatge rebut.

```
///////////
// Inicialització del pin de lectura, el wifi , el client mqtt
// i el Timer
//
///////////
//
void setup() {
    // Assignació del pin 12 com de lectura amb mode pull up
    pinMode(12, INPUT_PULLUP);
    // Inicialització del wifi i del client mqtt
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(incomingSub);

    //Assignació de les tasques periòdiques i els seus temps al timer
    timerRECON = timer.setInterval(PERIODO_RECON, reconecta);
    timerPUBSUB = timer.setInterval(PERIODO_PUBSUB, gestionaPubSub);
    //Desactivar la tasca de reconexió una vegada connectat
    timer.disable(timerRECON);
}

///////////
// Funció de callback a l'arribada d'un missatge
//
///////////
//
void incomingSub(char* topic, byte* payload, unsigned int length)
{
    Serial.print("Missatge rebut: [");
    Serial.print(topic);
    Serial.print("]");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}
// Fi callback()
```

Finalment, la funció setup assigna les tasques que el Timer portarà a terme.

La primera es la funció **reconnecta**. Aquesta funció es cridrà quan l'objecte client mqtt es trobi en estat desconectat. Fa un intent d'connexió i si té exit, envia un missatge de OK i desactiva la tasca del timer, en cas contrari, mostra un missatge al port serie i surt, el timer tornarà a cridar aquesta funció passat el temps assignat a la constant.

```
///////////
// Inicialització del pin de lectura, el wifi , el client mqtt
// i el Timer
//
///////////
//
void setup() {
    // Assignació del pin 12 com de lectura amb mode pull up
    pinMode(12, INPUT_PULLUP);
    // Inicialització del wifi i del client mqtt
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(incomingSub);

    //Assignació de les tasques periòdiques i els seus temps al timer
    timerRECON = timer.setInterval(PERIODO_RECON, reconecta);
    timerPUBSUB = timer.setInterval(PERIODO_PUBSUB, gestionaPubSub);
    //Desactivar la tasca de reconexió una vegada connectat
    timer.disable(timerRECON);
}

///////////
// Funció de reconexió després de desconexió del servidor MQTT
//
///////////
//
void reconecta()
{
    if (!client.connected()) {
        Serial.print("Intent de connexió MQTT...");
        // Creació d'un random client ID
        String clientId = "placa";
        clientId += String(random(0xffff), HEX);
        // Intent de connexió amb exit
        if (client.connect(clientId.c_str(), mqtt_user, mqtt_pwd)) {
            Serial.println("connectat");
            // Subscripció al Topic per rebre missatges
            client.subscribe(inTopic);
            // Enviament d'un missatge d'ACK de connexió OK al topic d'ACK
            client.publish(hiTopic, "¡Hola des de la placa!");
            // Desactivar la tasca de reconexió
            timer.disable(timerRECON);
        } else // En cas de fallada del intent de connexió
        {
            Serial.print("fallada, rc=");
            Serial.print(client.state());
            Serial.println(" Nou intent dintre d'uns segons");
        }
    }
}
```

La següent tasca assignada al Timer és **gestionaPubSub**. Aquesta funció comprova si el objecte client està connectat i si és així, crida a la funció **comprobaBoto** que actualitza l'estat del botó i envia un missatge amb aquest estat.

En cas de que el client mqtt estigui desconectat, habilita la tasca del timer per a tornar a connectar-lo i surt.

```
///////////
// Inicialització del pin de lectura, el wifi , el client mqtt
// i el Timer
//
void setup() {
    // Assignació del pin 12 com de lectura amb mode pull up
    pinMode(12, INPUT_PULLUP);
    // Inicialització del wifi i del client mqtt
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(incomingSub);
}

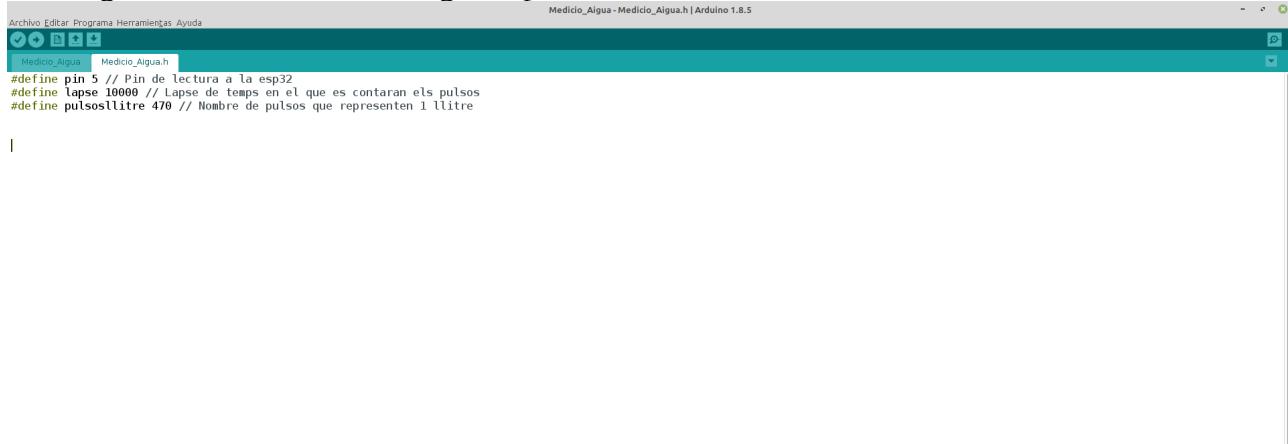
void loop()
{
    timer.run(); // mantenir el timer en marxa
}

///////////
// Comproba la connexió, i si no està connectat, activa la tasca de reconexió.
// Si està connectat, envia l'estat del botó
//
void gestionaPubSub()
{
    if (!client.connected() && !timer.isEnabled(timerRECON))
        timer.enable(timerRECON);
    else
    {
        comprobaBoto();
        char estat[2];
        sprintf(estat, "%d", estatBoto);
        // Enviament del missatge amb l'estat del botó al topic d'estat
        client.publish(EstatTopic, estat);
        client.loop();
    }
} // Fi gestionaPubSub()

///////////
// Comproba l'estat del botó i canvia el valor
// de la variable en conseqüència
//
void comprobaBoto()
{
    if(digitalRead(12) == 1)
    {
        estatBoto = 0;
        Serial.println("Botó no premut");
    }
    else
    {
        Serial.println("Botó premut");
        estatBoto = 1;
    }
}
```

## 4.2 Mesurament del consum d'aigua amb sensor ( Programa MedicioAigua)

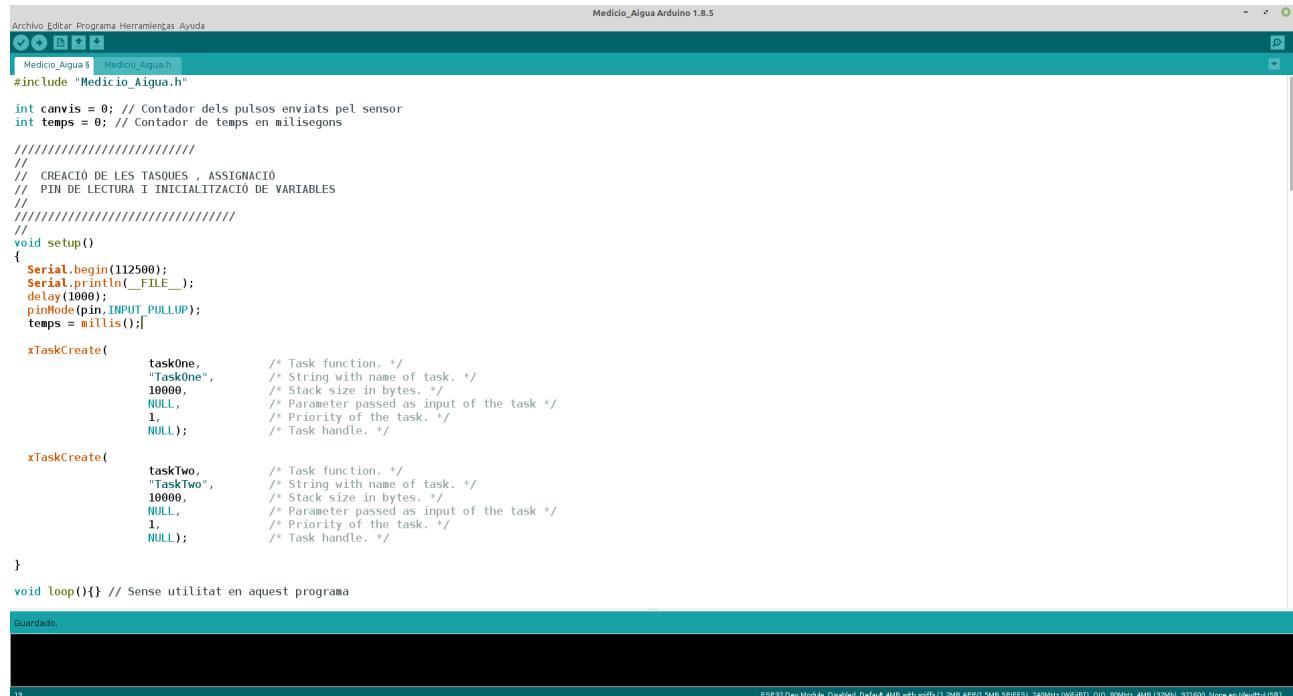
He afegit un fitxer Medicio\_Aigua.h per a definir-hi una serie de constants.



```
#define pin 5 // Pin de lectura a la esp32
#define lapse 10000 // Lapse de temps en el que es contaran els pulsos
#define pulsoslitre 470 // Nombre de pulsos que representen 1 llitre
```

Al fitxer .ino, he inclòs el fitxer .h , he instanciat dues variables pel recompte de canvis de tensió en el pin ( pulsos ) i el temps en millisegons.

A la funció setup, he declarat el pin com de entrada amb pull up, he inicialitzat el temps i he declarat les dues tasques que faré servir. La funció loop què ve per defecte no la faré servir en aquest cas.



```
#include "Medicio_Aigua.h"

int canvis = 0; // Contador dels pulsos enviats pel sensor
int temps = 0; // Contador de temps en millisegons

// CREALCIÓ DE LES TASQUES , ASSIGNACIÓ
// PIN DE LECTURA I INICIALIZACIÓ DE VARIABLES
// 
void setup()
{
    Serial.begin(112500);
    Serial.println(__FILE__);
    delay(1000);
    pinMode(pin, INPUT_PULLUP);
    temps = millis();
}

xTaskCreate(
    taskOne,           /* Task function. */
    "TaskOne",         /* String with name of task. */
    10000,             /* Stack size in bytes. */
    NULL,              /* Parameter passed as input of the task */
    1,                 /* Priority of the task. */
    NULL);            /* Task handle. */

xTaskCreate(
    taskTwo,           /* Task function. */
    "TaskTwo",         /* String with name of task. */
    10000,             /* Stack size in bytes. */
    NULL,              /* Parameter passed as input of the task */
    1,                 /* Priority of the task. */
    NULL);            /* Task handle. */

}

void loop(){} // Sense utilitat en aquest programa
```

La primera tasca, estableix una variable **bPinTeTensio** de tipus boolean per a enregistrar el darrer estat en el que es troava el pin, amb o sense tensió. Després s'entra en un bucle infinit que fa us de la funció btensio que retorna false si hi ha tensió al pin o true si no hi ha. La lectura queda enregistrada en una variable **bEstatActualPin**. Després compara el estat actual del pin amb el darrer estat, i si aquest es diferent, incrementa la variable global canvis. Per últim, s'assigna el estat actual del pin a la variable bPinTeTensio per a fer la comparació a la següent iteració.

```

Archivo Editar Programa Herramientas Ayuda
Medicio_Aigua.h Medicio_Aigua.ino
///////////////////////
// RETORNA TRUE SI HI HA TENSIÓ AL PIN
// DE NOMBRE PASAT COM A ARGUMENT I FALSE
// EN CAS CONTRARI
///////////////////////
boolean btensio(int npin)
{
    if (digitalRead(npin))
        return false;
    return true;
}
///////////////////////
// COMPROVA SI HA HAGUT UN CANVI DE ESTAT
// AL PIN ON EL DISPOSITIU ENVIA PULSOS I
// AUGMENTA EL COMPTADOR EN CAS AFIRMATIU
///////////////////////
void taskOne( void * parameter )
{
    static boolean bPinTeTensio = false;
    while(true)
    {
        boolean bEstatActualPin = btensio(pin);
        if(bEstatActualPin != bPinTeTensio)
        {
            canvis++;
            Serial.println("Canvi");
        }
        bPinTeTensio = bEstatActualPin;
        vTaskDelay(10);
    }
    vTaskDelete( NULL );
}

Guardado.

```

La segona tasca va comprovant si ha transcorregut el lapse de temps establert a la constant lapse i, si es així, fa la divisió entre la quantitat de pulsos i el paràmetre de pulsos per litre i la mostra al port serie. Finalment reseleta els comptadors de temps i pulsos.

```

///////////////////////
// COMPROVA EL TEMPS I QUAN ARriba AL LAPSE ESTABLERT
// MOSTRA EL VALOR DELS LlITRES QUE HAN PASSAT PEL
// SENSOR EN EL LAPSE ESTABLERT, DESPRÉS REINICIA
// ELS CONTADORS DE TEMPS I PULSOS
//
///////////////////////
void taskTwo( void * parameter)
{
    while(true)
    {
        if((millis() - temps) >= lapse) // 10 segons
        {
            Serial.println("Lapse");
            float llitres = (float)canvis / (float)pulsosllitre;
            char cadenacanvis[16];
            sprintf(cadenacanvis, "%d", canvis);
            char cadenalapsee[16];
            sprintf(cadenalapsee, "%d", lapse/1000);
            char cadenallitres[16];
            sprintf(cadenallitres, "%f", llitres);
            String a="Pulsos: " + String(cadenacanvis) + " en: " + String(cadenalapsee) + " segons" + " Llitres: " + cadenallitres;
            Serial.println(a);
            temps = millis();
            canvis = 0;
        }
        vTaskDelay(10);
    }
    vTaskDelete( NULL );
}

```

Els litres d'aigua consumits s'han de desar a una base de dades de InfluxDB. Per portar a terme això, la funció anterior afegirà les dades a una cua i un altre las anirà agafant per davant de la cua i les anirà pujant a la base de dades. La idea es què les dues funcions treballin de manera independent sense interferir-se.

Els fitxers Qua.cpp i Qua.h contenen 5 funcions per a la gestió d'una cua de floats.

```

Archivo Editar Programa Herramientas Ayuda
Medio_Aigua -> Medio_Aigua.h -> Qua.cpp -> Qua.h
Medio_Aigua - Qua.h | Arduino 1.8.5
+-----+
/* CONTE FUNCIONS PER A GESTIONAR UNA QUA DE FLOATS */
+-----+
/* Afegeix el float passat com a argument al darrera de la qua i incrementa la quantitat d'elements */
void afegirQua(float x, float *qua, int *nelements, int llarg);

/* Retorna el primer element de la qua, l'elimina i decrementa la quantitat d'elements */
float treuredeQua(float *qua, int *nelements, int llarg);

/* retorna 1 si la qua es buida i 0 si no ho està */
int quaBuida(int nelements);

/* retorna 1 si la qua es plena i 0 si no ho està */
int quaPlena(int nelements, int llarg);

/* retorna l'element del index pasat com a argument sense eliminar-lo de la qua */
float obtenirElement(int index, float *qua, int nelements);

|
```

```

Archivo Editar Programa Herramientas Ayuda
Medio_Aigua -> Medio_Aigua.h -> Qua.cpp -> Qua.h
Medio_Aigua - Qua.cpp | Arduino 1.8.5
#include <stdio.h>
#include "Qua.h"

/* CONTE FUNCIONS PER A GESTIONAR UNA QUA DE FLOATS */
+-----+
/* Afegeix el float passat com a argument al darrera de la qua i incrementa la quantitat d'elements */
void afegirQua(float x, float *qua, int *nelements, int llarg)
{
    if(quaPlena(*nelements, llarg))
        return;
    quallarg>(*nelements)-1 = x;
    (*nelements)++;
}

/* Retorna el primer element de la qua, l'elimina i decrementa la quantitat d'elements */
float treuredeQua(float *qua, int *nelements, int llarg)
{
    if(quaBuida(*nelements))
        return -1;
    float n = qua[llarg-1];
    (*nelements)--;
    return n;
}

/* retorna 1 si la qua es buida i 0 si no ho està */
int quaBuida(int nelements)
{
    return (nelements)?0:1;
}

/* retorna 1 si la qua es plena i 0 si no ho està */
int quaPlena(int nelements, int llarg)
{
    return (nelements == llarg)?1:0;
}

/* retorna l'element del index pasat com a argument sense eliminar-lo de la qua */
float obtenirElement(int index, float *qua, int nelements )
{
    if(quaBuida(nelements) || index > nelements-1 || index < 0)
        return -1;
    return qua[nelements-index];
}
```

Guardado

Invalid version found: 1.04  
Invalid version found: 1.04  
Invalid version found: 1.04

Al fitxer .ino, he definit el llarg de la cua amb una constant, he afegit un array dinamic de floats **qua** i una variable **nelements** per enregistrar la quantitat d'elements a la cua. A la funció setup, he reservat l'espai de memòria per a la cua i la he inicialitzat amb valor -1 atès que els valors negatius no tenen sentit com a quantitat de litres.

```

Archivo Editar Programa Herramientas Ayuda
Medicio_Aigua.ino Medicio_Aigua.h Qua.cpp Qua.h
#include "Medicio_Aigua.h"

#define llarg 100

int canvis = 0; // Contador dels pulsos enviats pel sensor
int temps = 0; // Contador de temps en milisegons

float * qua;
int nelements = 0;

/////////////////////////////
// CREADÓ DE LES TASQUES , ASSIGNACIÓ
// PIN DE LECTURA I INICIALITZACIÓ DE VARIABLES
// ///////////////////////////////
void setup()
{
    Serial.begin(112500);
    Serial.println(__FILE__);
    delay(1000);
    pinMode(pin, INPUT_PULLUP);
    temps = millis();

    qua = (float*)malloc(sizeof(float)*llarg);
    for(int i = 0; i < llarg; i++)
        qua[i] = -1.0f;

    xTaskCreate(
        taskOne,           /* Task function. */
        "TaskOne",          /* String with name of task. */
        10000,             /* Stack size in bytes. */
        NULL,              /* Parameter passed as input of the task */
        1,                 /* Priority of the task. */
        NULL);             /* Task handle. */

    xTaskCreate(
        taskTwo,           /* Task function. */
        "TaskTwo",          /* String with name of task. */
        10000,             /* Stack size in bytes. */
        NULL,              /* Parameter passed as input of the task */
        1,                 /* Priority of the task. */
        NULL);             /* Task handle. */

    xTaskCreate(
        taskThree,          /* Task function. */
        "TaskThree",         /* String with name of task. */
        10000,             /* Stack size in bytes. */
        NULL,              /* Parameter passed as input of the task */
        1,                 /* Priority of the task. */
        NULL);             /* Task handle. */
}

Guardado.
Invalid version found: 1.04
Invalid version found: 1.04
Invalid version found: 1.04

```

He declarat una nova tasca a la funció setup. Aquesta nova tasca s'encarregarà de treure els valors del davant de la cua i pujar-los a la base de dades.

```

Archivo Editar Programa Herramientas Ayuda
Medicio_Aigua.ino Medicio_Aigua.h Qua.cpp Qua.h
/////////////////////////////
// CREADÓ DE LES TASQUES , ASSIGNACIÓ
// PIN DE LECTURA I INICIALITZACIÓ DE VARIABLES
// ///////////////////////////////
void setup()
{
    Serial.begin(112500);
    Serial.println(__FILE__);
    delay(1000);
    pinMode(pin, INPUT_PULLUP);
    temps = millis();

    qua = (float*)malloc(sizeof(float)*llarg);
    for(int i = 0; i < llarg; i++)
        qua[i] = -1.0f;

    xTaskCreate(
        taskOne,           /* Task function. */
        "TaskOne",          /* String with name of task. */
        10000,             /* Stack size in bytes. */
        NULL,              /* Parameter passed as input of the task */
        1,                 /* Priority of the task. */
        NULL);             /* Task handle. */

    xTaskCreate(
        taskTwo,           /* Task function. */
        "TaskTwo",          /* String with name of task. */
        10000,             /* Stack size in bytes. */
        NULL,              /* Parameter passed as input of the task */
        1,                 /* Priority of the task. */
        NULL);             /* Task handle. */

    xTaskCreate(
        taskThree,          /* Task function. */
        "TaskThree",         /* String with name of task. */
        10000,             /* Stack size in bytes. */
        NULL,              /* Parameter passed as input of the task */
        1,                 /* Priority of the task. */
        NULL);             /* Task handle. */
}

Guardado.
Invalid version found: 1.04

```

A la segona tasca, he afegit la desa dels litres a la cua i , per depuració, he fet que es mostri el contingut de la cua cada vegada que s'afegeix una dada.

```

Archivo Editar Programa Herramientas Ayuda
Medio_Aigua.h Medio_Aigua.h Quia.cpp Quia.h
Medio_Aigua Arduino 1.8.5

///////////////////////////////
// COMPROBA EL TEMPS I QUAN ARRIBA AL LAPSE ESTABLERT
// FICA A LA CUA EL VALOR DELS LLITRES QUE HAN PASSAT PEL
// SENSOR EN EL LAPSE ESTABLERT, DESPRES REINICIA
// ELS CONTADORS DE TEMPS I PULSOS
// ///////////////////////
// void taskTwo( void * parameter)
{
  while(true)
  {
    if((millis() - temps) >= lapse) // 10 segons
    {
      Serial.println("Lapse");
      float llitres = (float)canvis / (float)pulsosllitre;
      afegeirAqua(llitres,qua,&elements,llarg);
      char cadenaanvis[16];
      sprintf(cadenaanvis,"%d", canvis);
      char cadenalapsee[16];
      sprintf(cadenalapse,"%d",lapse/1000);
      char cadenallitres[16];
      sprintf(cadenallitres,"%f",llitres);
      String a="Pulso: " + String(cadenaanvis) + " en: " + String(cadenalapse) + " segons" + " Llitres: " + cadenallitres;
      Serial.println(a);
      for(int i = llarg-1; i > llarg-elements-1; i--)
      {
        char cadenaqua[16];
        sprintf(cadenaqua,"%f",qua[i]);
        Serial.println(cadenaqua);
      }
      temps = millis();
      canvis = 0;
    }
    vTaskDelay(10);
  }
  vTaskDelete( NULL );
}
Guardado.

```

Per pujar les dades necessito connectar la placa a la xarxa WiFi. Per això he afegit les llibreries adients, un nou fitxer WiFi\_Credentials.h amb les dades de la connexió, i he declarat un objecte WiFiUDP.

```

Archivo Editar Programa Herramientas Ayuda
Medio_Aigua_Jo Medio_Aigua_Jo.h Quia.cpp Quia.h WiFi_Credentials.h
Medio_Aigua_Jo Arduino 1.8.5

#include <WiFiServer.h>
#include <WiFiU.h>
#include <WiFiUDP.h>
#include <WiFiClient.h>

#include "Medio_Aigua_Jo.h"
#include "Quia.h"
#include "WiFi_Credentials.h"

#define llarg 100

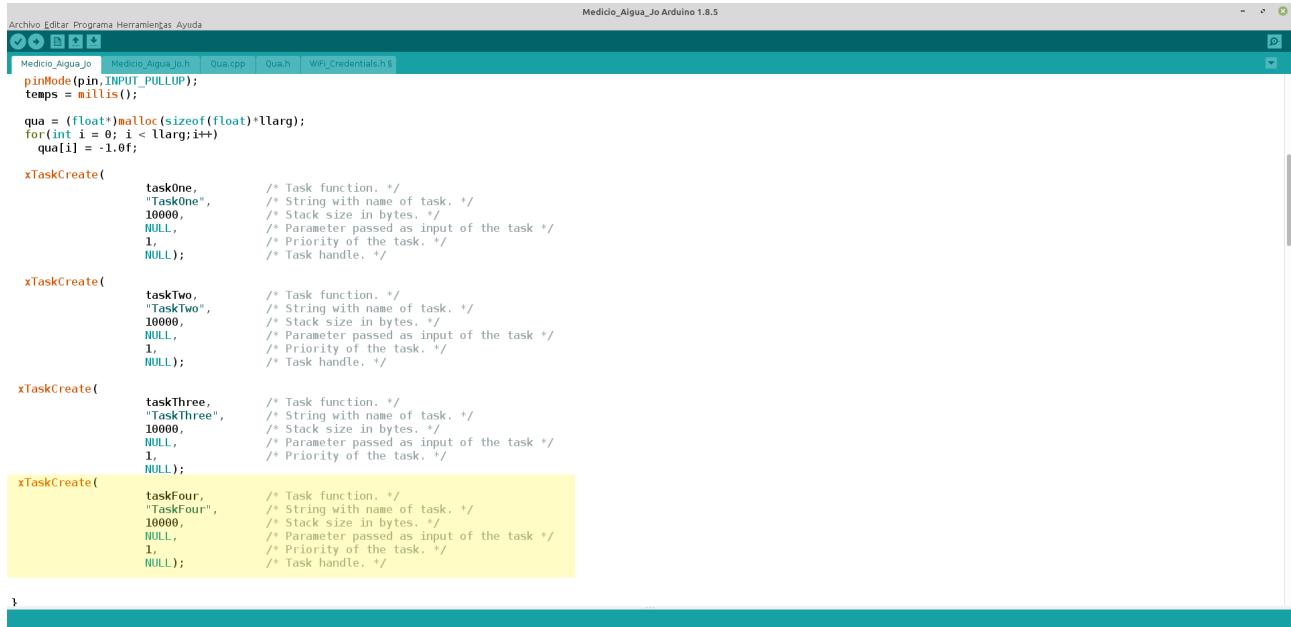
int canvis = 0; // Contador dels pulsos enviats pel sensor
int temps = 0; // Contador de temps en milisegons

float * qua;
int nelements = 0;

WiFiUDP udp;

```

He afegit una nova tasca per a realitzar la connexió.



The screenshot shows the Arduino IDE interface with the file 'Medio\_Aigua\_Jo.ino' open. The code is as follows:

```

Archivo Editar Programa Herramientas Ayuda
Medio_Aigua_Jo.ino [Medio_Aigua_Jo.h Quia.cpp Quia.h WiFi_Credentials.h]
pinMode(pin, INPUT_PULLUP);
temp = millis();

qua = (float*)malloc(sizeof(float)*llarg);
for(int i = 0; i < llarg; i++)
    qua[i] = -1.0f;

xTaskCreate(
    taskOne,
    "TaskOne",
    10000,
    NULL,
    1,
    NULL);

xTaskCreate(
    taskTwo,
    "TaskTwo",
    10000,
    NULL,
    1,
    NULL);

xTaskCreate(
    taskThree,
    "TaskThree",
    10000,
    NULL,
    1,
    NULL);

xTaskCreate(
    taskFour,
    "TaskFour",
    10000,
    NULL,
    1,
    NULL);

}

// Conexió a la xarxa WiFi
// 
void taskFour( void * parameter)
{
    delay(10);
    Serial.println();
    Serial.print("Conexió a ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        vTaskDelay(10);
    }

    randomSeed(micros());
    Serial.println("");
    Serial.println("Conectat a WiFi");
    Serial.println("Direcció IP: ");
    Serial.println(WiFi.localIP());
    vTaskDelete( NULL );
}

```

He implementat aquesta quarta tasca per a la connexió a la xarxa.

```

///////////
//
// Conexió a la xarxa WiFi
//
///////////
//
void taskFour( void * parameter)
{
    delay(10);
    Serial.println();
    Serial.print("Conexió a ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        vTaskDelay(10);
    }

    randomSeed(micros());
    Serial.println("");
    Serial.println("Conectat a WiFi");
    Serial.println("Direcció IP: ");
    Serial.println(WiFi.localIP());
    vTaskDelete( NULL );
}

```

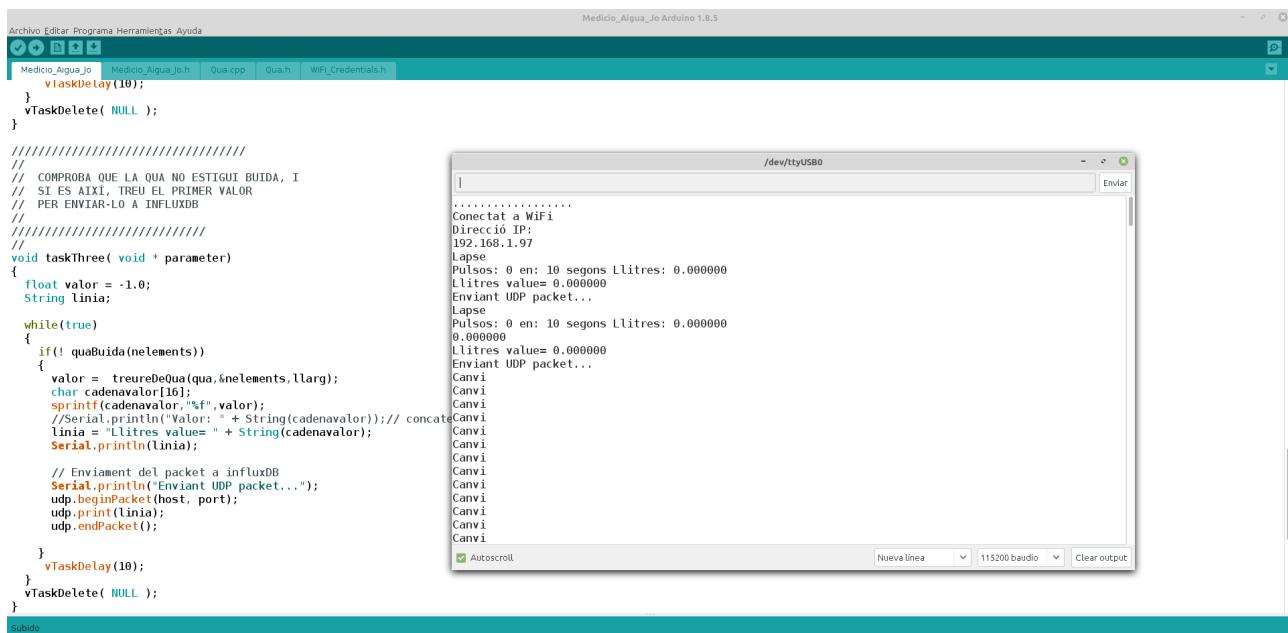
He modificat la tasca 3 perquè envii el mesurament dels litres per UDP a la base de dades influxDB. També comprova si hi ha connexió a la xarxa i, en cas contrari, posa en marxa la taca 4 de nou.

```
///////////
// COMPROBA QUE LA QUA NO ESTIGUI BUIDA, I
// SI ES AIXÍ, TREU EL PRIMER VALOR
// PER ENVIAR-LO A INFLUXDB
//
void taskThree( void * parameter)
{
    float valor = -1.0;
    String linia;

    while(true)
    {
        if(WiFi.status() != WL_CONNECTED) { // En cas de no estar connectat a la xarxa
            taskFour(NULL);
        }
        if(! quaBuida(nelements))
        {
            valor = treureDeQua(qua,&nelements,llarg);
            char cadenavalor[16];
            sprintf(cadenavalor,"%f",valor);
            linia = "consum_litres=" + String(cadenavalor); //Taula (consum), camp (litres) i valor
            Serial.println(linia);

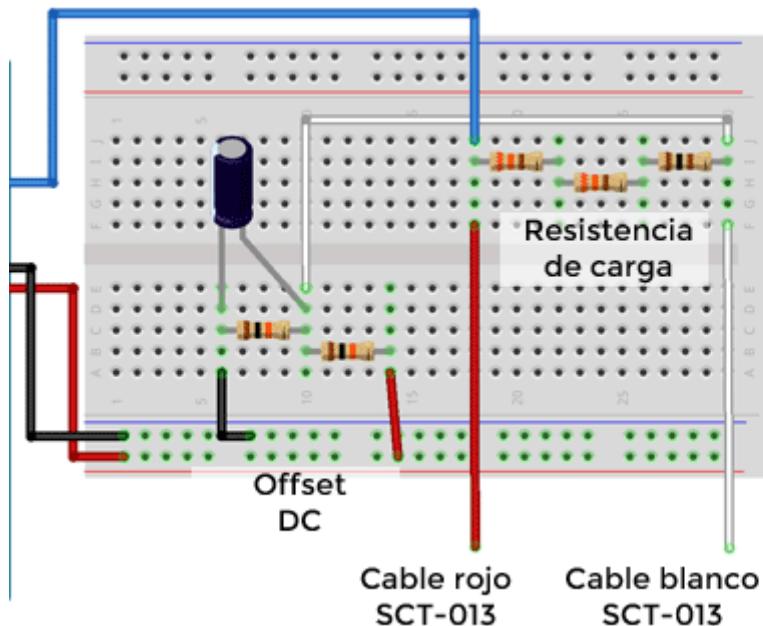
            // Enviament del packet a influxDB
            Serial.println("Enviant UDP packet...");
            udp.beginPacket(host, port);
            udp.print(linia);
            udp.endPacket();
        }
        vTaskDelay(10);
    }
    vTaskDelete( NULL );
}
```

He carregat l'aplicació i veig que es connecta correctament i comença a enviar packets



## 4.3 Mesurament del consum de llum (Programa MedicioLlum)

He tingut problemes per mesurar la corrent amb el sensor SCT-013. Vaig fer el muntatge del circuit a classe amb el professor que ha m'ha ajudat amb aquesta pràctica, en Jordi Binefa. Amb el seu ajut, vaig muntar un circuit similar al de la figura.



En Jordi va portar un Oscil·loscopi i vam mesurar el voltatge que el sensor proporcionava pel cable blanc. El resultat sempre era zero.

Al revisar el sensor, en Jordi es va adonar què aquest sensor no més donava 1 volt quan la intensitat que passa pel cable de la pinça és de 100 ampers.



Aquesta intensitat no està al meu abast i no em donava temps a adquirir un nou sensor i començar tot el procés de nou. Per això vaig decidir escriure un programa que simules la pressa de dades del consum d'un frigorífic i d'aquesta manera, provar la resta de funcions del programa general.

Per simular el temps què el frigorífic està ences i apagat i el seu consum en Watts, he escrit el fitxer RandomGenerator.cpp

```

Archivo Editar Programa Herramientas Ayuda
Medicio_Llum Qua.cpp Qua.h RandomGenerator.cpp RandomGenerator.h WiFi_Credentials.h
#include "RandomGenerator.h"
#include<stdlib.h>
#include<time.h>

/* Genera temps d'apagat entre 23 i 27 minuts en segons */
int generarTiempoApagado()
{
    srand(time(NULL));
    return rand()%240000+1380000;//Entre 23 y 27 minutos (n-m)+n = 1620 segons ; m = 1380 segons
}

/* Genera temps ences entre 20 y 24 minuts en segons */
int generarTiempoEncendido()
{
    srand(time(NULL));
    return rand()%240000+1440000;//Entre 20 y 24 minutos (n-m)+n = 1440 segons ; m = 1200 segons
}

/* Genera Watts entre 170 y 190 */
float generarValor()
{
    srand(time(NULL));
    float vrand = rand()%2000+19000;
    return vrand/100.0f;
}

```

El fitxer Medicio\_Llum.ino, utilitza les mateixes llibreries WiFi i WiFiUdp què el programa de mesura d'aigua. També he utilitzat el mòdul Qua.cpp, però en aquest cas, he utilitzat dues cues, una per els Watt i un altre pels WattH.

He instanciat una serie de variables globals per a les mesures.

```

Archivo Editar Programa Herramientas Ayuda
Medicio_Llum Qua.cpp Qua.h RandomGenerator.cpp RandomGenerator.h WiFi_Credentials.h
#define larg 100

float * qua;
int nelements = 0;

float * quawh;
int nelementswh = 0;

int tempsanterior = 0; // Per registrar el temps passat en milisegons
int horas = 0; // Para contar horas

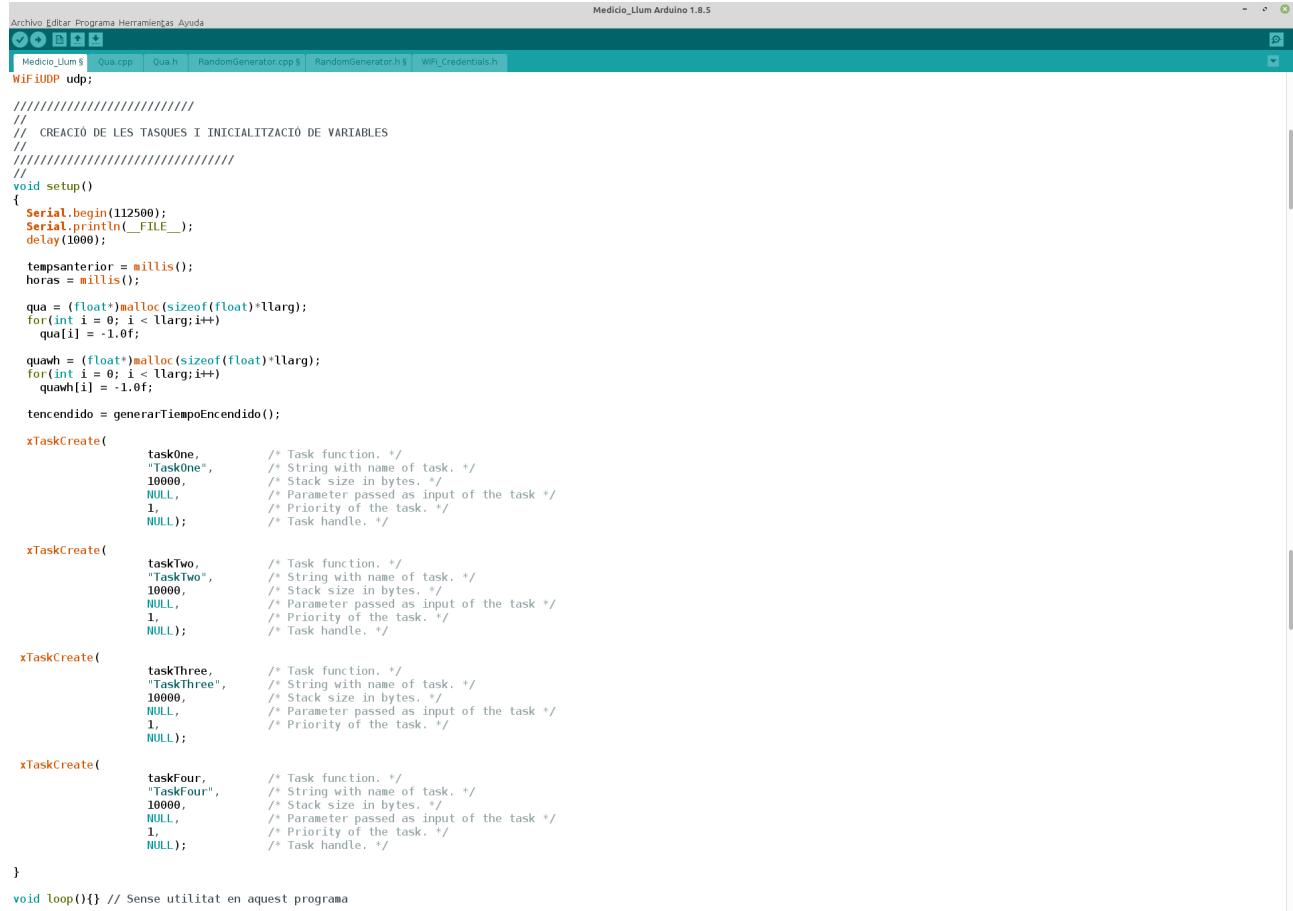
int tapagado = 0; //Temps apagat
int tencendido = 0; //Temps ences

float watts = 0; // Comptador de Watts
float wattsh = 0; // Comptador de Wattsh

WiFiUDP udp;

```

Al mètode setup, he inicialitzat les dues cues i he declarat quatre tasques de FreeRTOS.



```

Archivo Editar Programa Herramientas Ayuda
Medio_Lum Arduino 1.8.5
Medio_Lum.h Qua.cpp Qua.h RandomGenerator.cpp.h RandomGenerator.h WiFi_Credentials.h
WiFiUDP udp;
///////////////////////////////
// CREACIÓ DE LES TASQUES I INICIALITZACIÓ DE VARIABLES
/////////////////////////////
void setup()
{
  Serial.begin(112500);
  Serial.println(__FILE__);
  delay(1000);

  tempsanterior = millis();
  horas = millis();

  qua = (float*)malloc(sizeof(float)*llarg);
  for(int i = 0; i < llarg; i++)
    qua[i] = -1.0f;

  quawh = (float*)malloc(sizeof(float)*llarg);
  for(int i = 0; i < llarg; i++)
    quawh[i] = -1.0f;

  tencendido = generarTiempoEncendido();

  xTaskCreate(
    taskOne,           /* Task function. */
    "TaskOne",         /* String with name of task. */
    10000,             /* Stack size in bytes. */
    NULL,              /* Parameter passed as input of the task */
    1,                 /* Priority of the task. */
    NULL);            /* Task handle. */

  xTaskCreate(
    taskTwo,           /* Task function. */
    "TaskTwo",         /* String with name of task. */
    10000,             /* Stack size in bytes. */
    NULL,              /* Parameter passed as input of the task */
    1,                 /* Priority of the task. */
    NULL);            /* Task handle. */

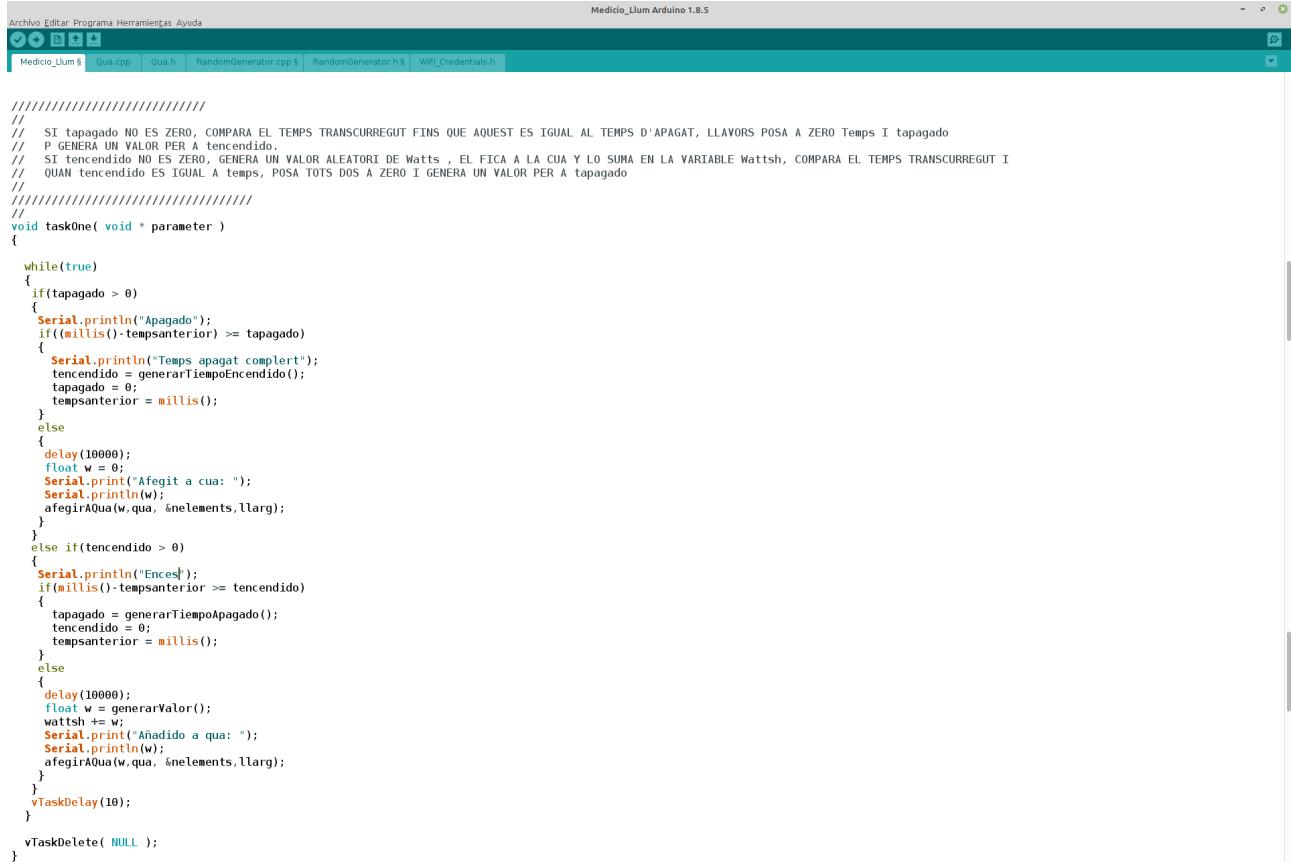
  xTaskCreate(
    taskThree,          /* Task function. */
    "TaskThree",        /* String with name of task. */
    10000,             /* Stack size in bytes. */
    NULL,              /* Parameter passed as input of the task */
    1,                 /* Priority of the task. */
    NULL);            /* Task handle. */

  xTaskCreate(
    taskFour,          /* Task function. */
    "TaskFour",         /* String with name of task. */
    10000,             /* Stack size in bytes. */
    NULL,              /* Parameter passed as input of the task */
    1,                 /* Priority of the task. */
    NULL);            /* Task handle. */
}

void loop(){} // Sense utilitat en aquest programa

```

La primera tasca s'encarrega de alternar la generació dels temps de encès i apagat i durant el temps encès, genera valors aleatoris de Watts i els fica a la cua.



```

Archivo Editar Programa Herramientas Ayuda
Medicio_Lium Arduino 1.8.5
Medicio_Lium.h Qua.cpp Qua.h RandomGenerator.cpp.h RandomGenerator.h.h WiFi_Credentials.h

///////////////////////
// SI tapagado NO ES ZERO, COMPARA EL TEMPS TRANSCURREGUT FINS QUE AQUEST ES IGUAL AL TEMPS D'APAGAT, LLAVORS POSA A ZERO Temps I tapagado
// P GENERA UN VALOR PER A tencendido.
// SI tencendido NO ES ZERO, GENERA UN VALOR ALEATORI DE Watts , EL FICA A LA CUA Y LO SUMA EN LA VARIABLE Wattsh, COMPARA EL TEMPS TRANSCURREGUT I
// QUAN tencendido ES IGUAL A temps, POSA TOTS DOS A ZERO I GENERA UN VALOR PER A tapagado
// ///////////////////
void taskOne( void * parameter )
{
    while(true)
    {
        if(tapagado > 0)
        {
            Serial.println("Apagado");
            if((millis() - tempsanterior) >= tapagado)
            {
                Serial.println("Temps apagat complet");
                tencendido = generarTiempoEncendido();
                tapagado = 0;
                tempsanterior = millis();
            }
            else
            {
                delay(1000);
                float w = 0;
                Serial.print("Afegit a cua: ");
                Serial.println(w);
                afegirAQua(w,qua, &nelements,llarg);
            }
        }
        else if(tencendido > 0)
        {
            Serial.println("Encés");
            if(millis() - tempsanterior >= tencendido)
            {
                tapagado = generarTiempoApagado();
                tencendido = 0;
                tempsanterior = millis();
            }
            else
            {
                delay(10000);
                float w = generarValor();
                wattsh += w;
                Serial.print("Añadido a qua: ");
                Serial.println(w);
                afegirAQua(w,qua, &nelements,llarg);
            }
        }
        vTaskDelay(10);
    }
    vTaskDelete( NULL );
}

```

La segona tasca fica els WattH a la segona cua cada vegada que el mesurament de Watts arriba a una hora.

```

///////////////////////
// COMPROVA SI horas A ARRIVAT A 3600000 MILISEGONS
// I SI ES AIXI, FICA Wattsh EN quawh I POSA horas y wattsh A ZERO
// ///////////////////
void taskTwo( void * parameter )
{
    while(true)
    {
        if((millis() - horas) >= 3600000)
        {
            Serial.println("Hora");
            afegirAQua(wattsh,quawh,&nelementsw, llarg);
            char cadenawattsh[16];
            sprintf(cadenawattsh, "%d", wattsh);
            String a="WattH: " + String(cadenawattsh);
            Serial.println(a);
            horas = millis();
            wattsh = 0;
        }
        vTaskDelay(10);
    }
    vTaskDelete( NULL );
}

```

La tercera tasca, puja els valors a influxDB quan aquests estan disponibles a alguna de les dues cues.

```

Archivo Editar Programa Herramientas Ayuda
Medio_Lium Arduino 1.8.5
Medio_Lium.h Qua.cpp Qua.h RandomGenerator.cpp RandomGenerator.h WiFi_Credentials.h
///////////////////////
// COMPROBA SI ALGUNA DE LES QUAS NO ESTA BUIDA
// I SI ES AIXI, TREU EL PRIMER VALOR Y EL PUJA A INFLUXDB
//
//
void taskThree( void * parameter)
{
    float valor = -1.0;
    String linea;
    while(true)
    {
        delay(1000);
        if(! quaBuida(nelements))
        {
            valor = treureDeQua(qua,&nelements,llarg);
            char cadenavalor[16];
            sprintf(cadenavalor,"%f",valor);
            linea = "consumEum watts=" + String(cadenavalor); //Taula (consumE), camp (watts) i valor
            Serial.println(linea);
            // Enviament del packet a influxDB
            Serial.println("Enviant UDP packet...");
            udp.beginPacket(host, port);
            udp.print(linea);
            udp.endPacket();
            udp.endPacket();
            vTaskDelay(10);
        }
        else if(! quaBuida(nelementswH))
        {
            valor = treureDeQua(quawH,&nelementswH,llarg);
            char cadenavalor[16];
            sprintf(cadenavalor,"%f",valor);
            linea = "consumHum wattsH=" + String(cadenavalor); //Taula (consumE), camp (wattsH) i valor
            Serial.println(linea);
            // Enviament del packet a influxDB
            Serial.println("Enviant UDP packet...");
            udp.beginPacket(host, port);
            udp.print(linea);
            udp.endPacket();
            vTaskDelay(10);
        }
    }
    vTaskDelete( NULL );
}

```

Com al programa anterior, la quarta tasca es l'encarregada d'establir la connexió a la xarxa WiFi.

```

///////////////////////
// Conexió a la xarxa WiFi
//
///////////////////////
void taskFour( void * parameter)
{
    delay(10);
    Serial.println();
    Serial.print("Conexió a ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        vTaskDelay(10);
    }

    randomSeed(micros());
    Serial.println("");
    Serial.println("Conectat a WiFi");
    Serial.println("Direcció IP: ");
    Serial.println(WiFi.localIP());

    vTaskDelete( NULL );
}

```

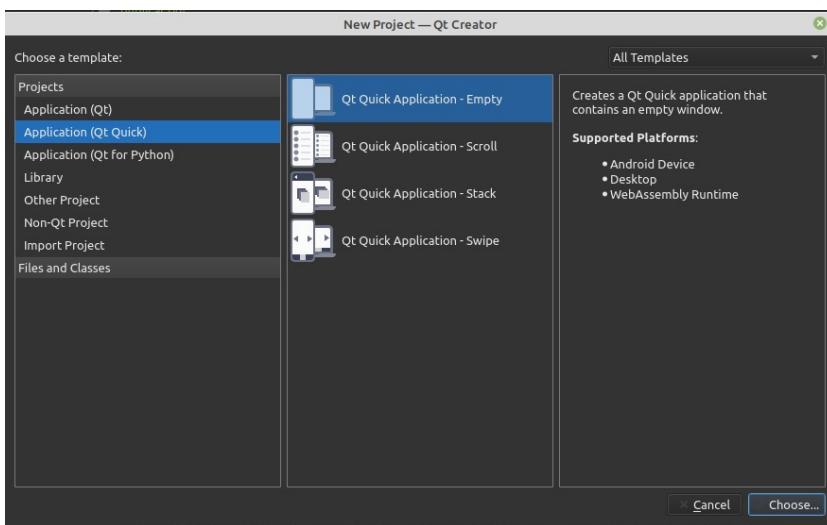
## 4.4 Enviament de codi de desactivació des de telefon ( Programa EnviaCodi)

Tenia dubtes sobre si fer aquest programa amb Qml i compilar-lo per a Android o fer-lo amb Java a Android Studio.

Per comparar els dos sistemes, vaig preparat un programa de cada tipus.

### Programa amb qml i QtCreator:

Vaig obrir un nou projecte qml



Per al rerefons, he afegit un fitxer qmlmqttclient.cpp i qmlmqttclient.h. Al fitxer .h, he inclòs la llibreria QtMqtt/QmqttClient i he declarat una classe **QmlMqttClient** com a derivada de QmqttClient.

En aquesta classe, he declarat el constructor i una funció per a publicar el QString passat com a argument.

48

Al fitxer cpp, he implementat el constructor i la funció per a publicar el missatge.

```

File Edit View Build Debug Analyze Tools Window Help
Projects publicaCodi [main]
  -> publicaCodi.pro
  -> Headers
    -> qmlmqttclient.h
  -> Sources
    -> main.cpp
      -> qmlmqttclient.cpp
      -> Resources
        -> qml.qrc
      -> Line 78, Col 1
      -> Unix (LF)
      -> Line 78, Col 1

qmlmqttclient.cpp @ publicaCodi [main] - Qt Creator
1 //*****
2 //include "qmlmqttclient.h"
3 //include <QDebug>
4 //include <QMqttTopicName>
5
6 ///////////////
7 /// \brief QmlMqttClient::QmlMqttClient
8 /// \param parent
9 /// Constructor
10 ///
11 QmlMqttClient::QmlMqttClient(QObject *parent)
12 : QMqttClient(parent)
13 {
14 }
15
16 ///////////////
17 /// \brief QmlMqttClient::publish
18 /// \param message
19 /// \param qos
20 /// \param retain
21 /// \return
22 /// Publica el QString passat com a primer argument
23 ///
24 int QmlMqttClient::publish( const QString &message, int qos, bool retain)
25 {
26     auto result = QMqttClient::publish(QMqttTopicName("CodiEnviat"), message.toUtf8(), qos, retain);
27     return result;
28 }
29

```

Per a connectar el rerefons amb el frontal, he enregistrat el tipus de la classe anterior al sistema Qml, a a la funció main.

```

File Edit View Build Debug Analyze Tools Window Help
Projects publicaCodi [main]
  -> publicaCodi.pro
  -> Headers
    -> qmlmqttclient.h
  -> Sources
    -> main.cpp
      -> qmlmqttclient.cpp
      -> Resources
        -> qml.qrc
      -> Line 69, Col 23
      -> Unix (LF)
      -> Line 69, Col 23

main.cpp @ publicaCodi [main] - Qt Creator
1 //*****
2 //include "qmlmqttclient.h"
3
4 #include <QGuiApplication>
5 #include <QQmlApplicationEngine>
6 #include <QLoggingCategory>
7
8 int main(int argc, char *argv[])
9 {
10     QGuiApplication app(argc, argv);
11
12     QQmlApplicationEngine engine;
13
14     qmlRegisterType<QmlMqttClient>("MqttClient", 1, 0, "MqttClient"); //--- Enregistrament del tipus QmlMqttClient
15     engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
16
17     if (engine.rootObjects().isEmpty())
18         return -1;
19
20     return app.exec();
21 }
22

```

Al fitxer main.qml, he implementat el frontal. En ell, he importat el tipus enregistrat abans. Després he afegit un objecte Window com a contenidor per a la resta de ginys, un objecte Text per a referenciar el text a enviar, un objecte MqttClient i un Component.onCompleted per a fer la connexió amb el broker quan s'inicii l'aplicació.

Després he afegit una funció per a canviar el contingut del objecte Text i un GridLayout que contindrà un TextField no visible amb el nombre del port ( l'he mantingut per fer-lo servir a la construcció del objecte MqttClient, perquè no accepta literals en aquest paràmetre ) i dos altres GridLayouts.

```
function afegeixChar(nombre)
{
    codi.text = codi.text+qsTr(nombre)
    msgField.text = codi.text
}

GridLayout {
    anchors.fill: parent
    anchors.margins: 10
    columns: 2

    TextField {
        id: portField
        Layout.fillWidth: true
        text: "1883"
        placeholderText: "<Port>"
        inputMethodHints: Qt.ImhDigitsOnly
        enabled: client.state === MqttClient.Disconnected
        visible: false
    }
}
```

Al primer dels GridLayout inclosos dintre del anterior, he afegit l'etiqueta per al rótul, la capsula de text per al codi a enviar i el botó que quan es premi publicarà el missatge.

```

GridLayout {
    enabled: client.state === MQTTClient.Connected
    Layout.columnSpan: 2
    Layout.fillWidth: true
    columns: 4

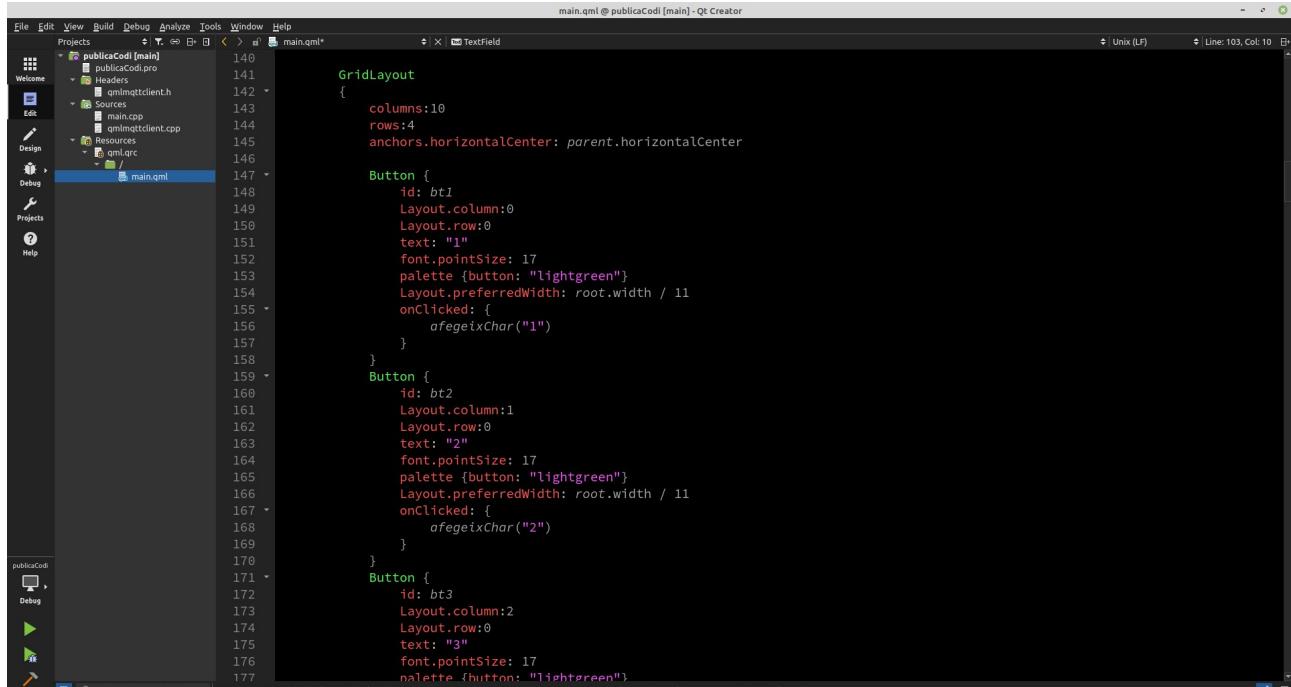
    Label {
        text: "Codi:"
    }

    TextField {
        id: msgField
        Layout.fillWidth: true
        background: Rectangle {
            radius: 2
            implicitWidth: 150
            implicitHeight: 60
            border.color: "#333"
            border.width: 1
            color:"silver"
        }
        font.pointSize: 29
    }

    Button {
        id: pubButton
        Layout.fillWidth: true
        text: "Envia"
        font.pointSize: 17
        palette {button: "chartreuse"}
        onClicked: {
            client.publish(msgField.text,"0",false)
        }
    }
}

```

Finalment en el segon GridLayout, he posat una matriu de botons amb els nombres i les lletres per a escriure el codi.



The screenshot shows the Qt Creator interface with the main.qml file open. The code defines a second GridLayout with 10 columns and 4 rows. It contains three rows of buttons labeled '1', '2', and '3'. The first two rows have 5 buttons each, and the third row has 3 buttons. Each button has a lightgreen background color and a lightgreen text color. The buttons are centered horizontally within their respective grid cells.

```

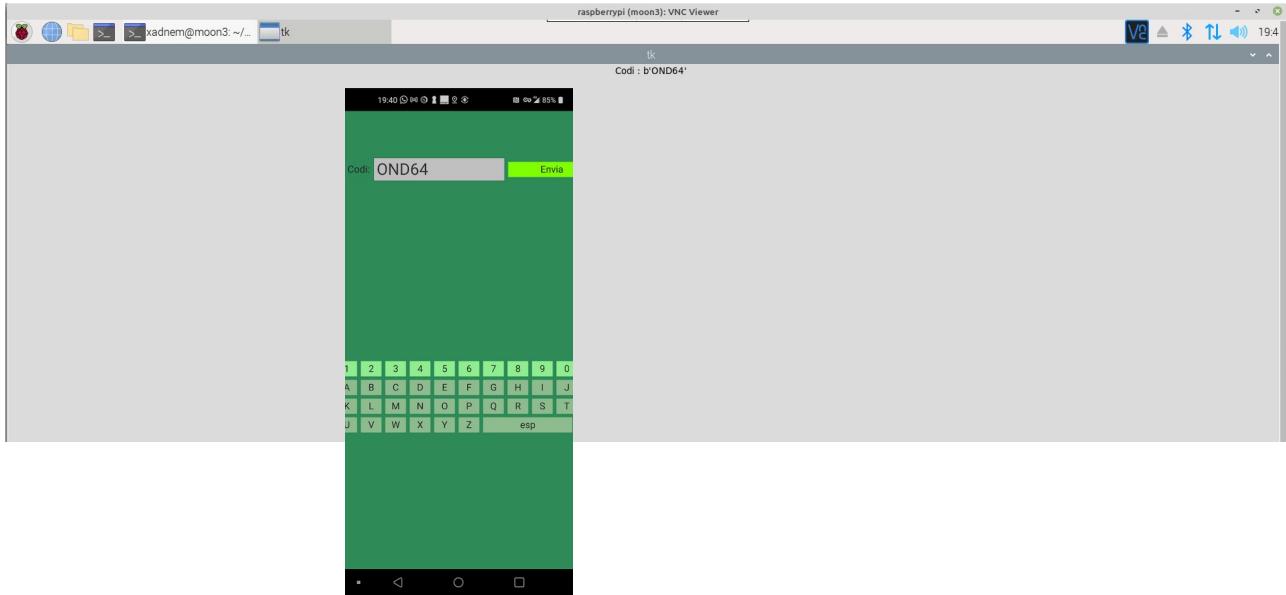
main.qml @ publicaCodi [main] - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
Projects publicaCodi (main) main.qml*
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177

GridLayout {
    columns:10
    rows:4
    anchors.horizontalCenter: parent.horizontalCenter

    Button {
        id: bt1
        Layout.column:0
        Layout.row:0
        text: "1"
        font.pointSize: 17
        palette {button: "lightgreen"}
        Layout.preferredWidth: root.width / 11
        onClicked: {
            afegeixChar("1")
        }
    }
    Button {
        id: bt2
        Layout.column:1
        Layout.row:0
        text: "2"
        font.pointSize: 17
        palette {button: "lightgreen"}
        Layout.preferredWidth: root.width / 11
        onClicked: {
            afegeixChar("2")
        }
    }
    Button {
        id: bt3
        Layout.column:2
        Layout.row:0
        text: "3"
        font.pointSize: 17
        palette {button: "lightgreen"}
    }
}

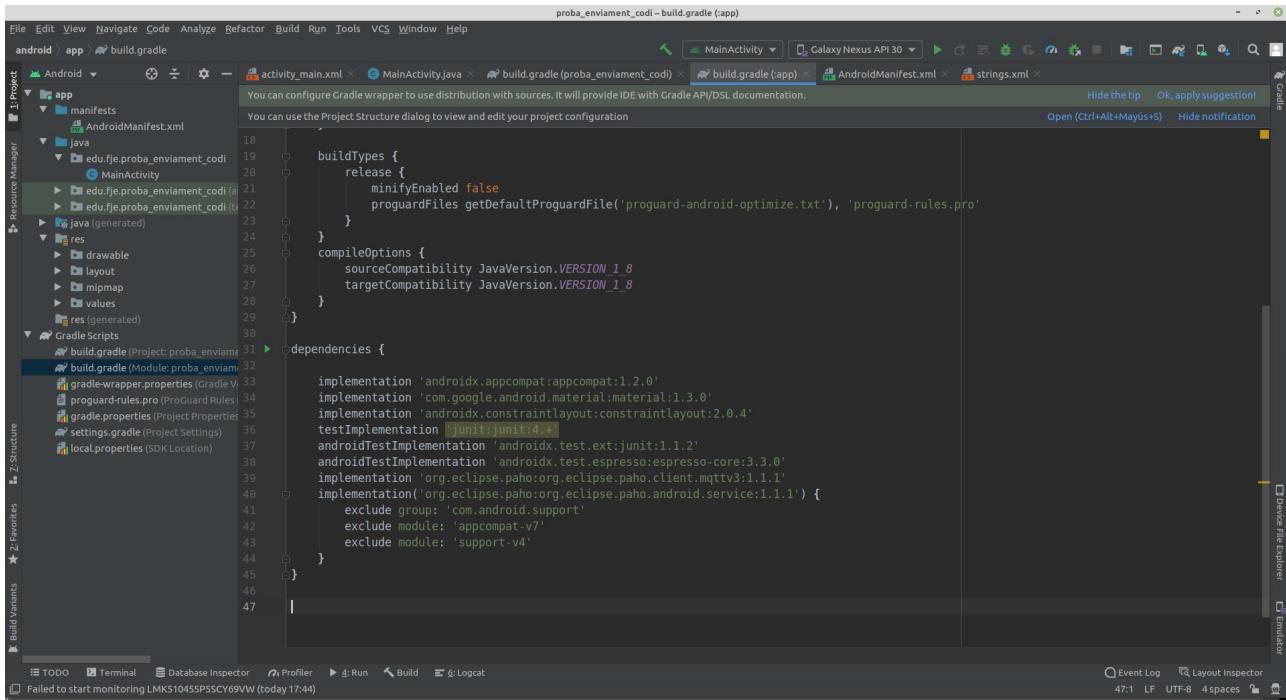
```

L'he compilat per a Android i he provat d'enviar un missatge al broker. Veig que el programa Python de prova a la Raspberry de casa meva a rebut el missatge.



## Programa amb java i Android Studio:

Per començar he afegit algunes dependències necessàries a la carpeta build.gradle(Module...)



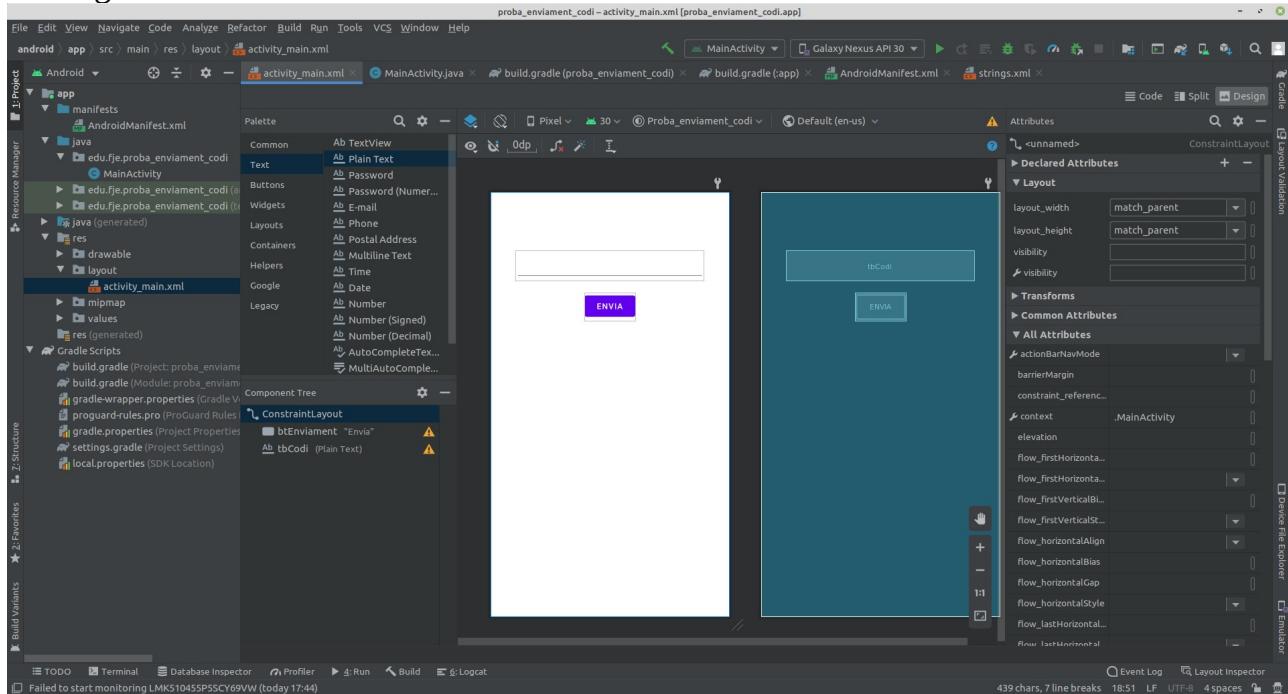
Després he afegit els permisos necessaris i el servei `eclipse.paho` en el manifest.

```

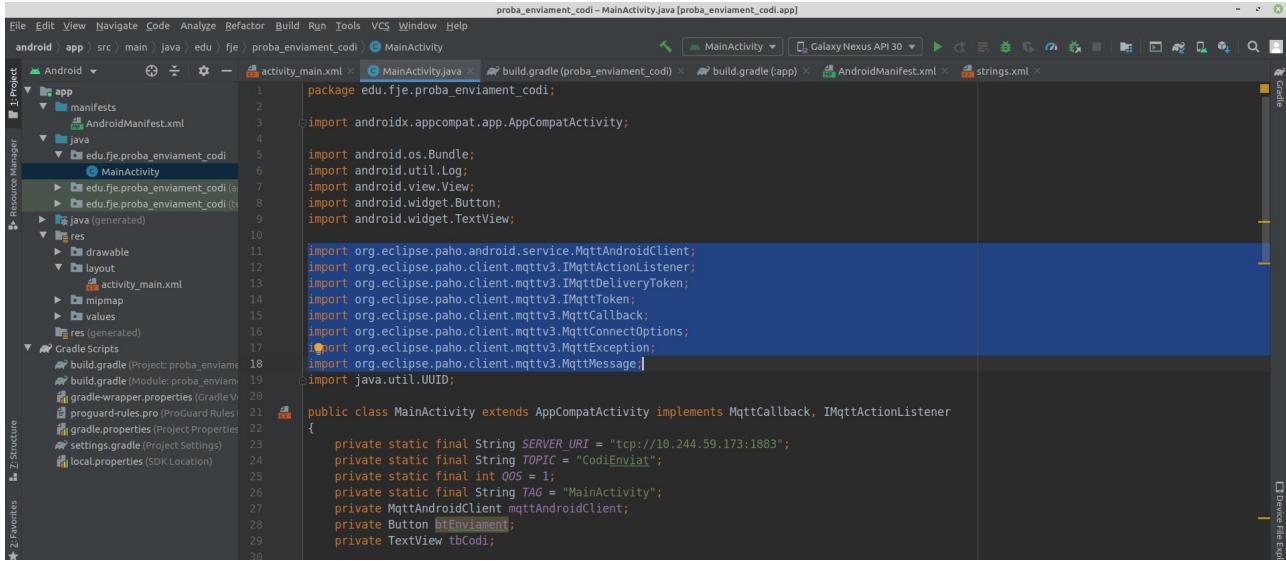
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.fje.proba_enviament_codi">
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        tools:ignore="ScopedStorage" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Proba_enviament_codi">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Mqtt Service -->
        <service android:name="org.eclipse.paho.android.service.MqttService">
        </service>
    </application>
</manifest>

```

Al layout per a la GUI, he afegit simplement una capsa de text i un botó per a publicar el seu contingut.



A la classe principal, he afegit les llibreries de `eclipse.paho` pel objecte `mqtt`, he instanciat les constants amb la ip del broker a la Raspberry, el topic el Qos, un String com a etiqueta i he declarat l'objecte `mqtt` i els widgets.



Al mètode onCreate, he referenciat els widgets , he implementat el onClickListener del botó i he creat i connectat al broker l'objecte mqtt.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Widgets
        btEnviament = findViewById(R.id.btEnviament);
        btEnviament.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String missatge = tbCodi.getText().toString();
                publishMessage(missatge);
            }
        });
        tbCodi = findViewById(R.id.tbCodi);
        // Objekte Mqtt
        String clientId = UUID.randomUUID().toString();
        Log.d(TAG, msg: "onCreate: clientId: " + clientId);

        mqttAndroidClient = new MqttAndroidClient(getApplicationContext(), SERVER_URI, clientId);
        mqttAndroidClient.setCallback(this);
        MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
        mqttConnectOptions.setCleanSession(true);
        mqttConnectOptions.setUserName(USERNAME);
        mqttConnectOptions.setPassword(PASSWORD.toCharArray());

        try {
            Log.d(TAG, msg: "onCreate: Connecting to " + SERVER_URI);
            mqttAndroidClient.connect(mqttConnectOptions, userContext: null, callback: this);
        } catch (MqttException ex){
            Log.e(TAG, msg: "onCreate: ", ex);
        }
    }
}
```

He afegit alguns mètodes útils per a monitorar el comportament del objecte mqtt , subscriure'l al topic per a la rebuda de missatges i gestionar la mateixa.

```

@Override
public void onSuccess(IMqttToken asyncActionToken) {
    Log.d(TAG, msg: "onSuccess: ");
    try {
        mqttAndroidClient.subscribe(TOPIC, QOS);
        mqttAndroidClient.subscribe(topic: "CodiCorrecte", QOS);
    } catch (Exception e) {
        Log.e(TAG, msg: "Error subscribing to topic", e);
    }
}

@Override
public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
    Log.e(TAG, msg: "Failed to connect to: " + SERVER_URI, exception);
}

@Override
public void connectionLost(Throwable cause) { Log.d(TAG, msg: "connectionLost: ", cause); }

@Override
public void messageArrived(String topic, MqttMessage message) {
    Log.d(TAG, msg: "Incoming message: " + new String(message.getPayload()));
    if(topic.equals("CodiCorrecte"))
        tbCodi.setText(new String(message.getPayload()));
}

@Override
public void deliveryComplete(IMqttDeliveryToken token) { Log.d(TAG, msg: "deliveryComplete: "); }

```

Per terminar el programa, he afegit el mètode per a publicar el missatge amb un Toast per a notificar l'enviament.

```

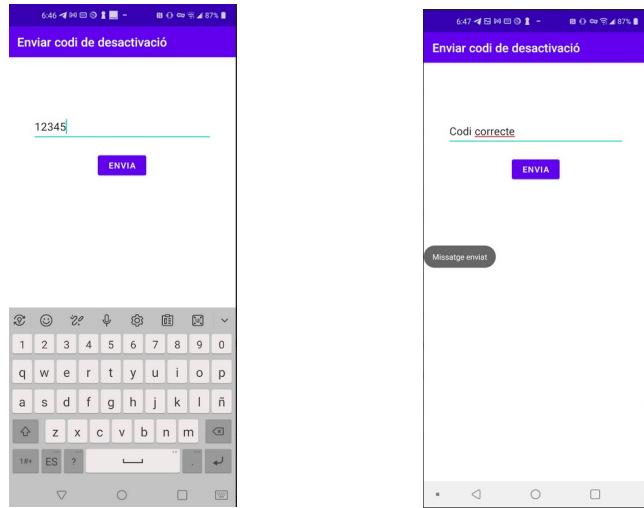
public void publishMessage(String payload) {
    try {
        if (mqttAndroidClient.isConnected() == false)
            mqttAndroidClient.connect();
    }

    MqttMessage message = new MqttMessage();
    message.setPayload(payload.getBytes());
    message.setQos(0);
    mqttAndroidClient.publish(TOPIC, message, userContext: null, new IMqttActionListener() {
        @Override
        public void onSuccess(IMqttToken asyncActionToken) {
            Log.i(TAG, msg: "publish succeed!");
            tbCodi.setText("");
            InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(tbCodi.getWindowToken(), flags: 0);
            Toast notificacio = Toast.makeText(getApplicationContext(), text: "Missatge enviat",
                Toast.LENGTH_LONG);
            notificacio.setGravity(gravity: Gravity.CENTER|Gravity.LEFT, xOffset: 0, yOffset: 0);
            notificacio.show();
        }

        @Override
        public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            Log.i(TAG, msg: "publish failed!");
        }
    });
} catch (MqttException e) {
    Log.e(TAG, e.toString());
    e.printStackTrace();
}
}

```

Finalment vaig escollir la opció amb Android Studio perquè em va semblar un resultat més professional i estètic.

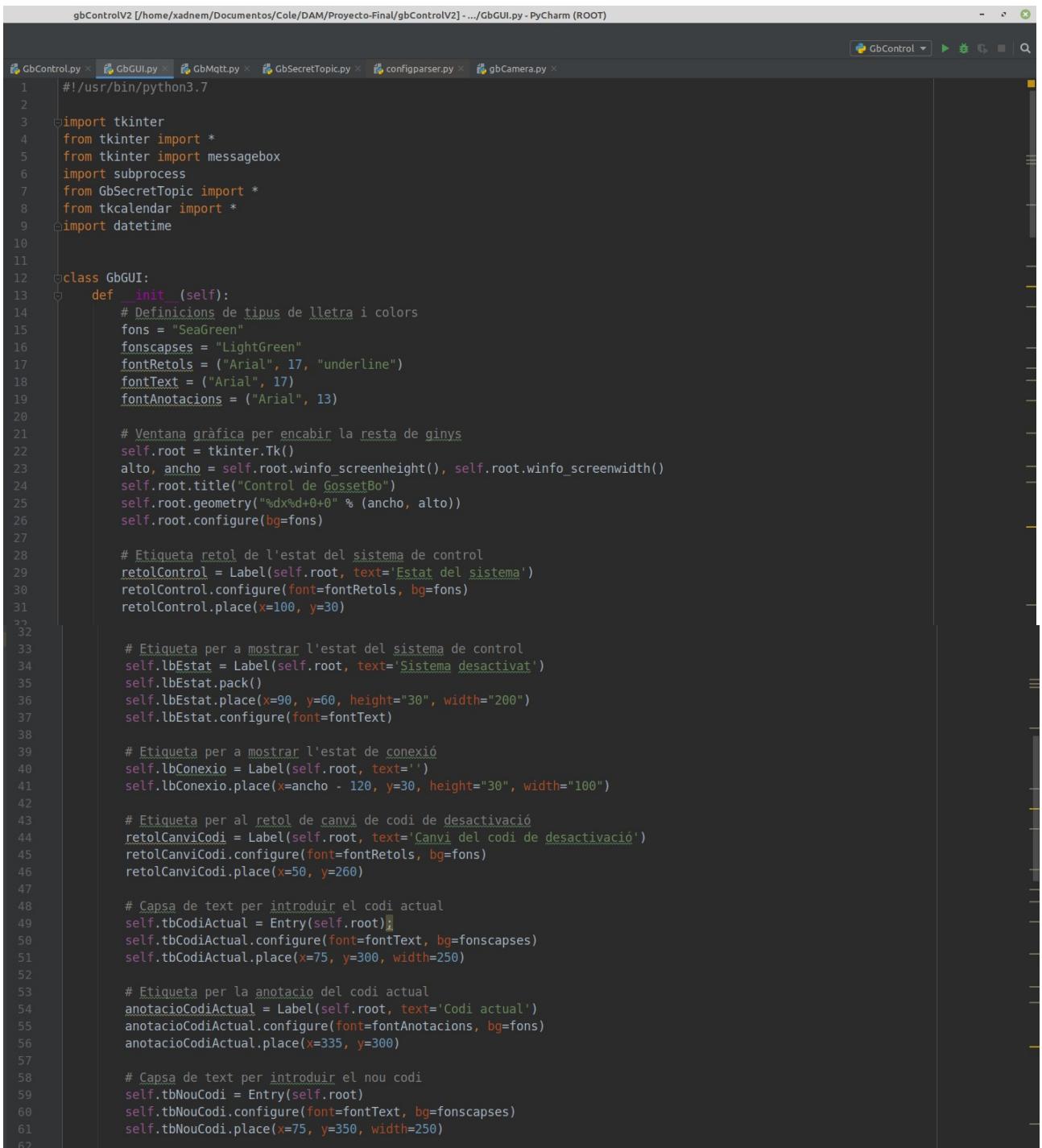


## 4.5 Programa de control general principal ( Programa GbControl)

### 4.5.1 GbGUI

Com el seu nom indica, aquest mòdul és on es defineix la GUI i els seus mètodes.

Al mètode `__init__`, es defineixen tots els widgets de Tkinter utilitzats. Alguns són atributs de la classe, per a poder accedir a ells des de d'altres mòduls quan es creïn objectes d'aquesta classe.



```

gbControlV2 [/home/xadnem/Documentos/Cole/DAM/Proyecto-Final/gbControlV2] -.../GbGUI.py - PyCharm (ROOT)
GbControl.py x GbGUI.py x GbMqtt.py x GbSecretTopic.py x configparser.py x gbCamera.py x

1  #!/usr/bin/python3.7
2
3  import tkinter
4  from tkinter import *
5  from tkinter import messagebox
6  import subprocess
7  from GbSecretTopic import *
8  from tkcalendar import *
9  import datetime
10
11
12 class GbGUI:
13     def __init__(self):
14         # Definicions de tipus de lletra i colors
15         fons = "SeaGreen"
16         fonsCapses = "LightGreen"
17         fontRetols = ("Arial", 17, "underline")
18         fontText = ("Arial", 17)
19         fontAnotacions = ("Arial", 13)
20
21         # Ventana gràfica per encabir la resta de ginys
22         self.root = tkinter.Tk()
23         alto, ancho = self.root.winfo_screenheight(), self.root.winfo_screenwidth()
24         self.root.title("Control de GossetBo")
25         self.root.geometry("%dx%d+0+0" % (ancho, alto))
26         self.root.configure(bg=fons)
27
28         # Etiqueta retol de l'estat del sistema de control
29         self.lbEstat = Label(self.root, text='Estat del sistema')
30         self.lbEstat.configure(font=fontRetols, bg=fons)
31         self.lbEstat.place(x=100, y=30)
32
33         # Etiqueta per a mostrar l'estat del sistema de control
34         self.lbEstat = Label(self.root, text='Sistema desactivat')
35         self.lbEstat.pack()
36         self.lbEstat.place(x=90, y=60, height="30", width="200")
37         self.lbEstat.configure(font=fontText)
38
39         # Etiqueta per a mostrar l'estat de connexió
40         self.lbConexio = Label(self.root, text='')
41         self.lbConexio.place(x=ancho - 120, y=30, height="30", width="100")
42
43         # Etiqueta per al retol de canvi de codi de desactivació
44         self.lbCanviCodi = Label(self.root, text='Canvi del codi de desactivació')
45         self.lbCanviCodi.configure(font=fontRetols, bg=fons)
46         self.lbCanviCodi.place(x=50, y=260)
47
48         # Capsa de text per introduir el codi actual
49         self.tbCodiActual = Entry(self.root)
50         self.tbCodiActual.configure(font=fontText, bg=fonsCapses)
51         self.tbCodiActual.place(x=75, y=300, width=250)
52
53         # Etiqueta per la anotació del codi actual
54         self.anotacioCodiActual = Label(self.root, text='Codi actual')
55         self.anotacioCodiActual.configure(font=fontAnotacions, bg=fons)
56         self.anotacioCodiActual.place(x=335, y=300)
57
58         # Capsa de text per introduir el nou codi
59         self.tbNouCodi = Entry(self.root)
60         self.tbNouCodi.configure(font=fontText, bg=fonsCapses)
61         self.tbNouCodi.place(x=75, y=350, width=250)
62

```

```

63     # Etiqueta per la anotació del nou codi
64     anotacioNouCodi = Label(self.root, text='Nou codi')
65     anotacioNouCodi.configure(font=fontAnotacions, bg=fons)
66     anotacioNouCodi.place(x=335, y=350)
67
68     # Botó per acceptar el nou codi
69     btCanviCodi = Button(self.root, text="Acceptar", bg="GreenYellow", command=self.onBtCanviCodi_Clicked)
70     btCanviCodi.place(x=165, y=400)
71
72     lbRetolGrafana = Label(self.root, text="Gràfics", bg="seagreen", fg="Black",
73                             justify="left", anchor=W, font=fontRetols)
74     lbRetolGrafana.place(x=120, y=520)
75
76     # Etiqueta per retol del calendari de la data d'inici del gràfic
77     lbDataInici = Label(self.root, text="Inici de període", bg="seagreen", fg="Black",
78                          justify="left", anchor=W, font=fontAnotacions)
79     lbDataInici.place(x=250, y=570)
80     # EntryCalendar per a la data d'inici
81     self.calendariInici = DateEntry(self.root, font=("Dejavu", 13))
82     self.calendariInici.place(x=90, y=570)
83
84     # # Etiqueta per retol del calendari de la data de fi del gràfic
85     lbDataFi = Label(self.root, text="Fi de període", bg="seagreen", fg="Black",
86                      justify="left", anchor=W, font=fontAnotacions)
87     lbDataFi.place(x=250, y=630)
88     # Mostrar el EntryCalendar per a la data de fi del període
89     self.calendariFi = DateEntry(self.root, font=("Dejavu", 13))
90     self.calendariFi.place(x=90, y=630)
91
92     # Botó per obrir el gràfic de consum d'aigua a Grafana
93     btAigua = Button(self.root, text="Aigua", bg="LightBlue", command=self.onBtGrafanaA_Clicked)
94     btAigua.place(x=90, y=690)
95
96     # Botó per obrir el gràfic de consum de llum a Grafana
97     btLlum = Button(self.root, text="Llum", bg="RoyalBlue", command=self.onBtGrafanaL_Clicked)
98     btLlum.place(x=180, y=690)
99

```

El mètode `onBtCanviCodi_Clicked` comprova què l'usuari introduceix el codi actual correcte. Si es així, crida a la funció `canviarCodi` del mòdul `GbSecretTopic` passant-li el nou codi com a argument. Si la operació es porta a terme, es mostra un messagebox d'informació. Si l'operació no s'ha realitzat, es mostra un messagebox d'error.

En cas de que es deixi alguna casella buida o el codi actual sigui erroni, es mostra un messagebox d'error.

```

99
100    def onBtCanviCodi_Clicked(self):
101
102        if self.tbCodiActual.get() != GbSecretTopic.codi:
103            messagebox.showerror("Error", "Codi actual erroni")
104        elif len(self.tbNouCodi.get()) == 0:
105            messagebox.showerror("Error", "El nou codi no pot estar buit")
106        else:
107            codi = self.tbNouCodi.get()
108            canviat = GbSecretTopic.canviarCodi(codi)
109            if canviat:
110                messagebox.showinfo("OK", "Codi de desactivació canviat")
111                self.tbCodiActual.delete(0, len(self.tbCodiActual.get()))
112                self.tbNouCodi.delete(0, len(self.tbNouCodi.get()))
113            else:
114                messagebox.showerror("Error", "No s'ha canviat el codi de desactivació." + GbSecretTopic.codi)

```

Els mètodes `onBtGrafanaA_Clicked` i `onBtGrafanaL_Clicked`, agafen les dades als widgets `tkCalendar`, les canvien a unix time i les fan servir a la uri per a obrir el gràfic de Grafana en el rang escollit per l'usuari.

Per obrir la finestra del navegador en un procés diferent, s'utilitza el mètode `Popen` de `subprocess`.

```

115
116     def onBtGrafanaA_Clicked(self):
117         # Datas de inici i final als calendaris
118         inici = self.calendariinici.get_date()
119         fi = self.calendarifl.get_date()
120         # Obtindre els strings de les datas amb barra en lloc de guions
121         inicib = str(inici).replace(".", "/")
122         fib = str(fi).replace(".", "/")
123         # Obtindre el unix time de les datas
124         unixinici = datetime.datetime.strptime(inicib, '%Y/%m/%d').strftime("%s")
125         unixfi = datetime.datetime.strptime(fib, '%Y/%m/%d').strftime("%s")
126         uri = 'http://127.0.0.1:3000/d/qox2hekz/consum-aigua?orgId=1&from=' + unixinici + '999&to=' + unixfi + '000'
127         self.p = subprocess.Popen(['chromium-browser', uri])
128
129     def onBtGrafanaL_Clicked(self):
130         # Datas de inici i final als calendaris
131         inici = self.calendariinici.get_date()
132         fi = self.calendarifl.get_date()
133         # Obtindre els strings de les datas amb barra en lloc de guions
134         inicib = str(inici).replace(".", "/")
135         fib = str(fi).replace(".", "/")
136         # Obtindre el unix time de les datas
137         unixinici = datetime.datetime.strptime(inicib, '%Y/%m/%d').strftime("%s")
138         unixfi = datetime.datetime.strptime(fib, '%Y/%m/%d').strftime("%s")
139         uri = 'http://localhost:3000/d/rjqt0ciRk/consum-llum?from=' + unixinici + '999&to=' + unixfi + '000'
140         self.p = subprocess.Popen(['chromium-browser', uri])
141
142

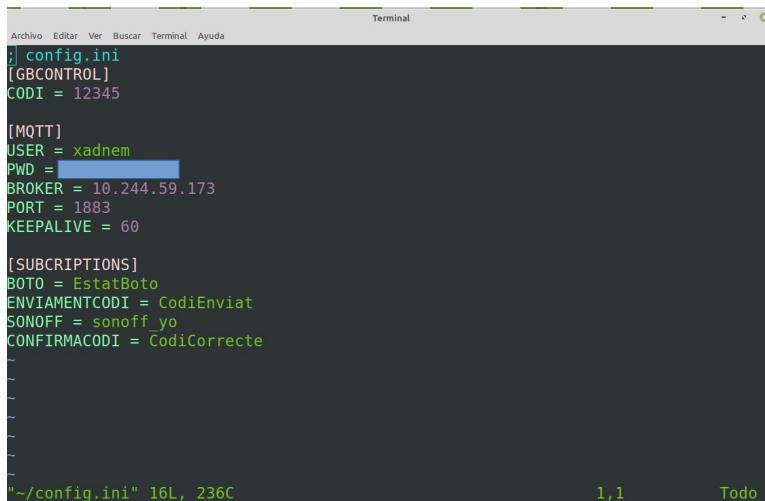
```

#### 4.5.2 GbSecretTopic

El propòsit d'aquest mòdul, es la de llegir les variables de configuració del programa emmagatzemades a un fitxer.

Fa us de la classe **configparser**, la qual proporciona facilitats per al maneig d'aquest tipus de fitxers.

Primer he preparat el fitxer de text amb les dades:



```

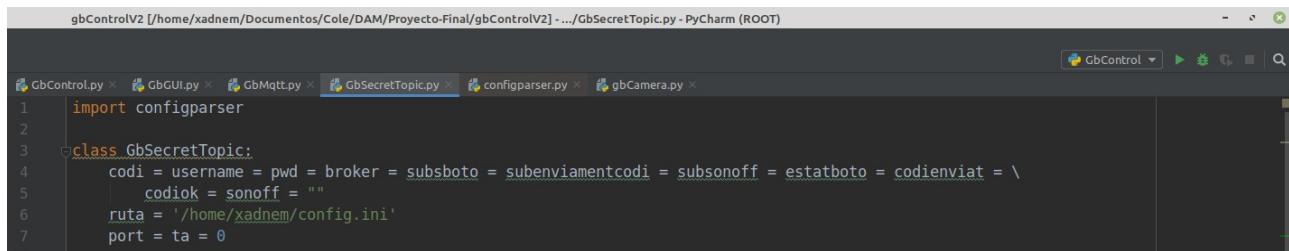
[config.ini]
[GBCONTROL]
CODI = 12345

[MQTT]
USER = xadnem
PWD =
BROKER = 10.244.59.173
PORT = 1883
KEEPALIVE = 60

[SUBSCRIPTIONS]
BOTO = EstatBoto
ENVIAMENTCODI = CodiEnviat
SONOFF = sonoff_yo
CONFIRMACODI = CodiCorrecte
-
```

Es pot veure que el fitxer consta d'unes capseleres ( GBCONTROL , MQTT, SUBSCRIPTIONS ) les quals són els apartats on es clasifiquen les dades. A cada apartat, hi ha una sèrie de parells clau / valor. Aquest és el format que la classe configparser entenc.

A la classe GbSecretTopic he instanciat una sèrie d'atributs de la classe, els quals corresponen a les claus del fitxer de configuració i a la ruta d'aquest.



```
gbControlV2 [/home/xadnem/Dокументos/Cole/DAM/Proyecto-Final/gbControlV2] .../GbSecretTopic.py - PyCharm (ROOT)
GbControl.py x GbUI.py x GbMqt.py x GbSecretTopic.py x configparser.py x gbCamera.py x
1 import configparser
2
3 class GbSecretTopic:
4     codi = username = pwd = broker = subsboto = subenviamentcodi = subsonoff = estatboto = codienviat = \
5         codiok = sonoff = ""
6     ruta = '/home/xadnem/config.ini'
7     port = ta = 0
```

La classe té dos mètodes de classe, el primer és una mena de constructor, què llegeix el fitxer i assigna els valors als atributs de la classe.



```
9 @classmethod
10 def establecerSecrets(cls, ):
11     config = configparser.ConfigParser()
12     config.read(GbSecretTopic.ruta)
13     GbSecretTopic.codi = config['GBCONTROL']['CODI']
14     GbSecretTopic.username = config['MQTT']['USER']
15     GbSecretTopic.pwd = config['MQTT']['PWD']
16     GbSecretTopic.broker = config['MQTT']['BROKER']
17     GbSecretTopic.port = int(config['MQTT']['PORT'])
18     GbSecretTopic.ta = int(config['MQTT']['KEEPALIVE'])
19     GbSecretTopic.subsboto = config['SUBSCRIPTIONS']['BOTO']
20     GbSecretTopic.subenviamentcodi = config['SUBSCRIPTIONS']['ENVIAMENTCODI']
21     GbSecretTopic.subsonoff = config['SUBSCRIPTIONS']['SONOFF']
22     GbSecretTopic.estatboto = config['SUBSCRIPTIONS']['BOTO']
23     GbSecretTopic.codienviat = config['SUBSCRIPTIONS']['ENVIAMENTCODI']
24     GbSecretTopic.codiok = config['SUBSCRIPTIONS']['CONFIRMACODI']
25     GbSecretTopic.sonoff = config['SUBSCRIPTIONS']['SONOFF']
```

El segon, rep una cadena i l'assigna al atribut codi i l'escriu al fitxer de configuració. Si la operació es porta a terme, retorna True, en cas contrari, retorna False.



```
26
27     @classmethod
28     def canviarCodi(cls, noucodi):
29         config = configparser.ConfigParser()
30         config.read(GbSecretTopic.ruta)
31         config.set('GBCONTROL', 'CODI', noucodi)
32         try:
33             with open(GbSecretTopic.ruta, 'w') as configFile:
34                 config.write(configFile)
35                 GbSecretTopic.codi = noucodi
36         except Exception as e:
37             print(str(e))
38             return False
39         return True
40
```

### 4.5.3 GbCamera

Aquest mòdul crea un servidor http què transmet les imatges de la càmera de la Raspberry al port 8000 del localhost.

El mòdul es un compendi de classes què s'acaben utilitzant al final del mateix.

Es comença amb unes línies de codi html assignat a una variable PAGE.

Aquest codi s'utilitza per al retul de la pàgina i de la finestra del navegador. La primera classe StreamingOutput, conté tres atributs: el primer per a frames ( quadres de la càmera ), el segon és un buffer de bytes i el tercer un objecte de la classe Condition què servei per a la gestió de fils.

El mètode write d'aquesta classe, copia el buffer passat com a argument en el buffer de la classe i ho notifica a tots els fils d'execució.

```

1 import io
2 import picamera
3 import logging
4 import socketserver
5 from threading import Condition
6 from http import server
7
8 PAGE="""\
9 <html>
10 <head>
11 <title>Camera de GossetBo</title>
12 </head>
13 <body>
14 <center><h1>Camera de GossetBo</h1></center>
15 <center></center>
16 </body>
17 </html>
18 """
19
20 class StreamingOutput(object): # Classe derivada d'objecte per a encapsular un buffer de bytes i un objecte condition per gestionar fils
21     def __init__(self):
22         # Atributs de la classe
23         self.frame = None
24         self.buffer = io.BytesIO() # Buffer de bytes
25         self.condition = Condition() # Condició per a gestionar fils.
26
27     def write(self, buf):
28         if buf.startswith(b'\xff\xd8'):
29             # New frame, copy the existing buffer's content and notify all
30             # clients it's available
31             self.buffer.truncate()
32             with self.condition:
33                 self.frame = self.buffer.getvalue()
34                 self.condition.notify_all()
35             self.buffer.seek(0)
36         return self.buffer.write(buf)

```

La segona classe d'aquest mòdul, es un handler ( gestor ) què defineix les tasques a realitzar quan es rep una petició de transmissió.

La classe deriva de BaseHTTPRequestHandler i sobreescrivia el mètode do\_GET, què és un mètode de SimpleHTTPRequestHandler, la classe base de BaseHTTPRequestHandler.

```

37
38     class StreamingHandler(server.BaseHTTPRequestHandler):# Clase derivada per a definir les tasques a realitzar quan es rep una nova petició
39         def do_GET(self):
40             if self.path == '/':
41                 self.send_response(301)
42                 self.send_header('Location', '/index.html')
43                 self.end_headers()
44             elif self.path == '/index.html':
45                 content = PAGE.encode('utf-8')
46                 self.send_response(200)
47                 self.send_header('Content-Type', 'text/html')
48                 self.send_header('Content-Length', len(content))
49                 self.end_headers()
50                 self.wfile.write(content)
51             elif self.path == '/stream.mjpg':
52                 self.send_response(200)
53                 self.send_header('Age', 0)
54                 self.send_header('Cache-Control', 'no-cache, private')
55                 self.send_header('Pragma', 'no-cache')
56                 self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
57                 self.end_headers()
58                 try:
59                     while True:
60                         with output.condition:
61                             output.condition.wait()
62                             frame = output.frame
63                             self.wfile.write(b'--FRAME\r\n')
64                             self.send_header('Content-Type', 'image/jpeg')
65                             self.send_header('Content-Length', len(frame))
66                             self.end_headers()
67                             self.wfile.write(frame)
68                             self.wfile.write(b'\r\n')
69                 except Exception as e:
70                     logging.warning(
71                         'Removed streaming client %s: %s',
72                         self.client_address, str(e))
73                 else:
74                     self.send_error(404)
75                     self.end_headers()
76

```

L'última classe StreamingServer deriva de HTTPServer i de ThreadingMixIn.

Finalment, es crea un objecte PiCamera que proporciona enquadraments que són escrits per un objecte StreamingOutput en el seu buffer que a la seva vegada es utilitzat per a un objecte StreamingServer quan es produeix un event de petició de transmissió gestionat per StreamingHandler en un bucle infinit que no més es detén quan es rep una petició explícita de finalització

```

77     """ Classe derivada de socketserver.ThreadingMixIn i de server.HTTPServer per encapsular els seus mètodes
78     i atributs """
79     class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
80         allow_reuse_address = True # Atribut de socketserver que permet reutilitzar adreces
81         daemon_threads = True # Atribut de ThreadingMixIn per indicar si el servidor ha d'esperar per a la finalització dels seus fils d'ex
82
83         with picamera.PiCamera(resolution='640x480', framerate=24) as camera: # Crea un objecte PiCamera
84             output = StreamingOutput() # Crea un objecte StreamingOutput
85             # Rotat l'imatge de la càmera
86             camera.rotation = 180
87             camera.start_recording(output,
88                                    format='mpeg') # Comença a gravar video i el desa al objecte StreamingOutput anterior
89         try:
90             address = ('', 8000) # Localhost i port 8000
91             server = StreamingServer(address,
92                                     StreamingHandler) # Crea un objecte StreamingServer amb la direcció passada com a primer argument i el
93             # controlador del event de recepció de peticions com a segon
94             server.serve_forever() # Gestio de peticions de transmissió fins a rebre una petició explícita de finalització
95         finally:
96             camera.stop_recording()
97

```

#### 4.5.4 GbMqtt

Aquesta classe deriva de mqtt.Client i l'ha utilitzat per a afegir-li una llista de topics com a atribut. Els objectes d'aquesta classe es subscriuràn al topic de la llista a través del mètode subscriureTopics. Els altres dos mètodes, són alias de mètodes existents a mqtt.Client per a establir el nom i password del user i connectar el objecte amb el broker.

```

1 import paho.mqtt.client as mqtt
2
3
4 class GbMqtt(mqtt.Client):
5     def __init__(self, llistaTopics):
6         mqtt.Client.__init__(self)
7         self.llistaTopics = llistaTopics
8         self.connectat = False
9
10    def estableixerUserPassword(self, user, pwd):
11        self.username_pw_set(user, pwd)
12
13    def connectarABroker(self, broker, port, timealive):
14        self.connect(broker, port, timealive)
15
16    def subscriureTopics(self):
17        for topic in self.llistaTopics:
18            self.subscribe(topic)
19

```

#### 4.5.5 GbControl

Aquesta és la classe principal del programa.

El seu mètode `__ini__` comença instanciat els atributs de la classe:

- **armat**: de tipus booleà per registrar quan el sistema està armat i quan no.
- **estatboto**: De tipus int què se estableix en funció dels missatges enviats per la placa esp8266 amb PulsacioBoto.
- **p i p2**: Per a registrar els processos paral·lels i poder finalitzar-los quan calgui.
- **gui**: Objecte de la classe GbGUI per accedir als seus widgets de la classe.
- **mqttClient**: Objecte de la classe GbMqtt què es construeix amb una llista dels topics als que vull que es subscrigeixi. També li són assignats els callbacks què es cridaran quan es produeixin els esdeveniments de connexió al broker i de rebuda de missatges.

Finalment es posa l'etiqueta lbConexio de gui del color i amb el text corresponent al estat de connexió del objecte mqttClient i s'inicien els bucles infinitis de mqttClient i de gui, els quals es porten a terme per defecte, en processos separats.

```

1  #!/usr/bin/python3.7
2
3  from GbSecretTopic import *
4  from GbGUI import GbGUI
5  from GbMqtt import GbMqtt
6  from os import system
7  import subprocess
8
9
10 class GbControl:
11     def __init__(self):
12         self.armat = False
13         self.estatboto = 0
14         self.p = None
15         self.p2 = None
16         # Inicialització de la classe GbSecretTopic amb les dades dels topics , codi de desactivació i ip's
17         GbSecretTopic.establirSecrets()
18         # Creació de objecte GbGUI
19         self.gui = GbGUI()
20         # Creació del objecte GbMqtt subscrit als topic de control
21         llistaTopics = [GbSecretTopic.subsboto, GbSecretTopic.subenviamentcodi, GbSecretTopic.subsonoff,
22                         GbSecretTopic.codienviat]
23         self.mqttclient = GbMqtt(llistaTopics)
24         self.mqttclient.establirUserPassword(GbSecretTopic.username, GbSecretTopic.pwd)
25         self.mqttclient.connectarABroker(GbSecretTopic.broker, GbSecretTopic.port, GbSecretTopic.ta)
26         self.mqttclient.on_message = self.on_message_rebut
27         self.mqttclient.on_connect = self.quan_connectat
28         if self.mqttclient.connectat:
29             self.gui.lbConexio.configure(bg="Chartreuse", text="Connectat")
30         else:
31             self.gui.lbConexio.configure(bg="Red", text="Desconnectat")
32         self.mqttclient.loop_start()
33         self.gui.root.mainloop()
34

```

El callback on\_message\_rebut del objecte mqttClient, es receptiu a dos topics:

- **estatboto:**

- Si el sistema no estava armat i rep un 1 com a missatge, posa la variable armat a True i mostra «Sistema activat» a l'etiqueta lbEstat de gui.
- Si el sistema estava armat i rep un 0 com a missatge, mostra el missatge «WARNING» a l'etiqueta lbEstat de gui, executa el script de Bash per al enviament del missatge d'avís al bot de Telegram, i executa gbCamera i obre el navegador en dos subprocessos referenciat als atributs p i p2 de la classe i activa el interruptor SonOff enviant el missatge ON al topic subsonoff.

- **codienviat:**

- Si el missatge rebut es correspon amb el codi de desactivació a GbSecretTopic, posa el missatge «Sistema desactivat» a l'etiqueta lbEstat de gui i publica «Codi correcte» al topic codiok per què sigui vist pel usuari al telèfon.
- També finalitza els processos del servidor http associat a la càmera , el del navegador i publica «OFF» al topic subsonoff per disconnectar el interruptor SonOff.
- Si el missatge rebut no es correspon amb el codi de desactivació a GbSecretTopic, publica «Codi incorrecte» al topic codiok per què sigui vist pel usuari a l'aplicació del telèfon.

```

35
36     def on_message_rebut (self, client, userdata, msg):
37         """ Mètode de gestió del event de rebuda de missatges pel objecte mqtt """
38         print(f"{msg.topic}{msg.payload}")
39         if msg.topic == GbSecretTopic.estatboto:
40             self.estatboto = int(msg.payload)
41             if not self.armat and self.estatboto == 1:
42                 self.armat = True
43                 self.gui.lbEstat.config(text="Sistema activat")
44             elif self.armat and self.estatboto == 0:
45                 self.gui.lbEstat.config(text=";; WARNING !! ")
46                 system("sh probaBot/bot.sh")
47                 self.mqttclient.publish(GbSecretTopic.subsonoff, "ON")
48
49                 # Activar la camera i el servidor http en un altre fil
50                 self.p = subprocess.Popen(['python3', 'gbCamera.py'])
51                 # Obrir el navegador per veure el contingut de la càmera
52                 self.p2 = subprocess.Popen(['chromium-browser', 'http://127.0.0.1:8000'])
53                 #self.p2 = subprocess.Popen(['firefox', 'http://127.0.0.1:8000'])
54                 self.armat = False
55                 # self.etiqueta.config(text = "Estat del botó: " + msg.payload.decode("utf-8")) #decode per a convertir de bytes a string
56             elif msg.topic == GbSecretTopic.codienviat:
57                 if msg.payload.decode("utf-8") == GbSecretTopic.codi:
58                     self.gui.lbEstat.config(text="Sistema desactivat ")
59                     self.mqttclient.publish(GbSecretTopic.codiok, "Codi correcte")
60                     client.publish(GbSecretTopic.sonoff, "OFF")
61                     if self.p is not None:
62                         self.p.kill()
63                     if self.p2 is not None:
64                         self.p2.kill()
65                     self.armat = False
66                 else:
67                     client.publish(GbSecretTopic.codiok, "Codi incorrecte")

```

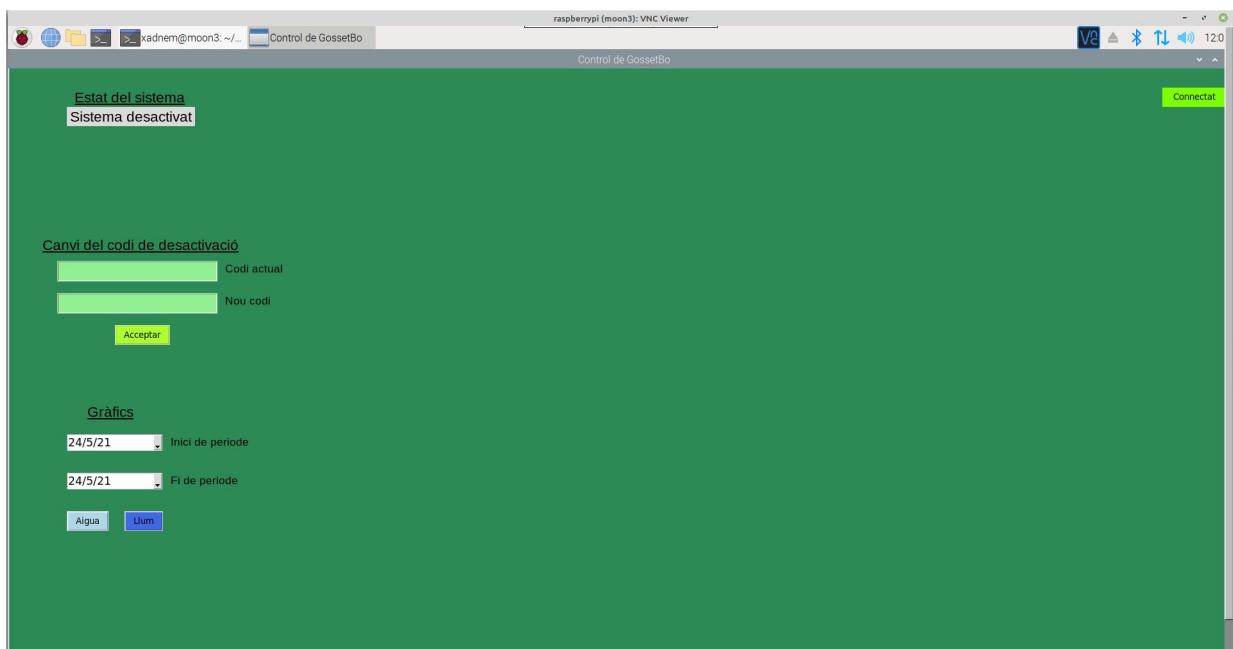
Per últim, el callback quan\_connectat, quan es produeix el esdeveniment de connexió al broker, crida al mètode subscriureTopics de mqttClient per què es subscrigeui a la seva llista de topics i posa l'etiqueta lbConexio de gui en verd amb el missatge «Connectat».

Si es produeix una desconnexió al broker, posa l'etiqueta lbConexio de gui en vermell i mostra «No connectat»

```

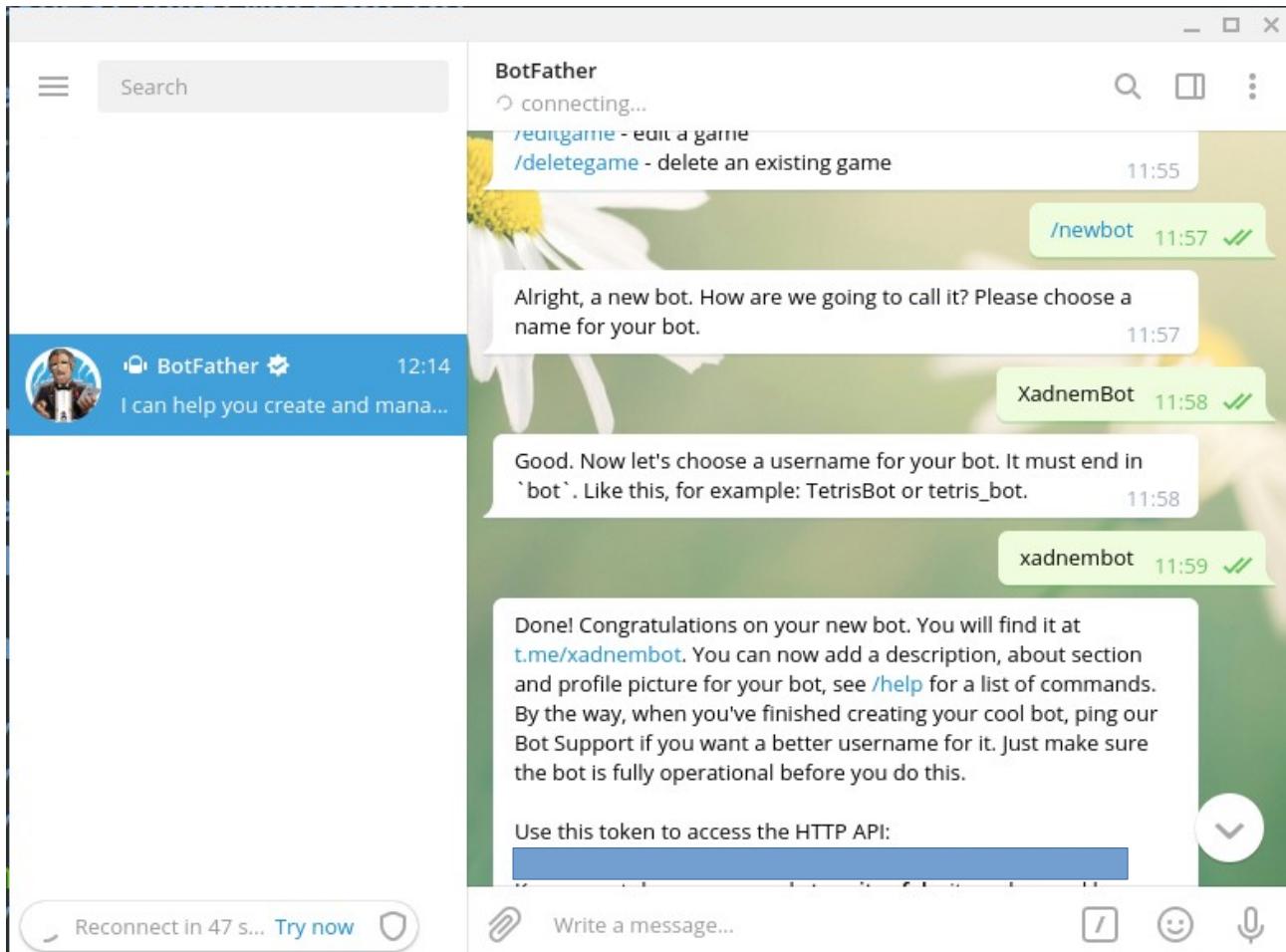
70
71     def quan_connectat(self, client, userdata, flags, rc):
72         # Mètode de connexió al broker mqtt
73         if rc == 0:
74             self.mqttclient.subscriureTopics()
75             self.mqttclient.connectat = True
76             self.gui.lbConexio.configure(bg="Chartreuse", text="Connectat")
77             print("Connectat")
78         else:
79             print(f"(Connexió fallida amb codi: {rc})")
80             self.mqttclient.connectat = False
81             self.gui.lbConexio.configure(bg="Red", text="No connectat")
82

```



## 4.6 Creació del bot de Telegram i Script d'enviament de missatges

Primer he creat un bot a Telegram seguint les instruccions que es proporcionen a tal fi. Al final del procs, he obtingut un token d'accés.



Una vegada creat el bot, he obtingut el meu id. Per aconseguir-lo, s'ha de cercar a Telegram l'usuari @userinfobot i enviar-li el missatge /start. Això fa que envii el nostre id.



Finalment, he escrit amb una serie de variables on he posat el token del bot, el meu id, el missatge a enviar i la url del bot de Telegram. Una sentència de curl fent ús d'aquestes variables s'encarrega de enviar el missatge al bot de Telegram.

```

#!/bin/bash

TOKEN="REDACTED"
ID="1498658345"
MENSAJE="WARNING!! Possible intrusió."
URL="https://api.telegram.org/bot$TOKEN/sendMessage"

curl -s -X POST $URL -d chat_id=$ID -d text="$MENSAJE"

```

## 4.7 Instal·lació de broker Mqtt a la Raspberry

```
xadnem@moon3: /etc/apt/sources.list.d
Archivo Editar Pestañas Ayuda
ectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 50 [application/octet-stream]
Grabando a: "mosquitto-buster.list"

mosquitto-buster.li 100%[=====] 50 --- KB/s en 0s
2021-05-03 09:18:59 (15,5 MB/s) - "mosquitto-buster.list" guardado [50/50]

xadnem@moon3:/etc/apt/sources.list.d $ sudo apt-get update
Des:1 http://download.zerotier.com/debian/buster InRelease [36,8 kB]
Des:2 http://archive.raspberrypi.org/debian buster InRelease [32,9 kB]
Des:3 http://archive.raspberrypi.org/debian buster/main armhf Packages [372 kB]
Des:4 http://raspbian.raspberrypi.org/raspbian buster InRelease [15,0 kB]
Des:5 http://download.zerotier.com/debian/buster/main armhf Packages [3.1
43 B]
Des:6 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13,0
MB]
Des:7 https://repo.mosquitto.org/debian buster InRelease [12,4 kB]
Des:8 https://repo.mosquitto.org/debian buster/main all Packages [3.850 B]
Des:9 https://repo.mosquitto.org/debian buster/main armhf Packages [30,6 kB]
Descargados 13,5 MB en 9s (1.444 kB/s)
Leyendo lista de paquetes... Hecho
xadnem@moon3:/etc/apt/sources.list.d $
```

He modificat l'arxiu /etc/mosquitto/mosquitto.conf per què el broker accepti missatges anònims.

```
moon3 Home (on moon3)
xadnem@moon3: /etc/mosquitto
Archivo Editar Pestañas Ayuda
+xadnem@... xadnem@... xadnem@... xadnem@... xadnem@... xadnem@...
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.gz

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto

log_dest file /var/log/mosquitto/mosquitto.log
#log_dest topic
allow_anonymous true

log_type error
log_type warning
log_type notice
log_type information

connection_messages true
log_timestamp true

listener 1883
#listener 9001
#protocol websockets

include_dir /etc/mosquitto/conf.d
-- VISUAL --
15 25,21 Todo
xadnem@... Node-RED...
```

He inclos la Raspberry a la meva xarxa virtual a Zero Tier. He copiat la ip a aquesta xarxa i l'he posat al programa de Python GbControl per què es connecti a aquest broker i poder accedir a ell des de qualsevol lloc.

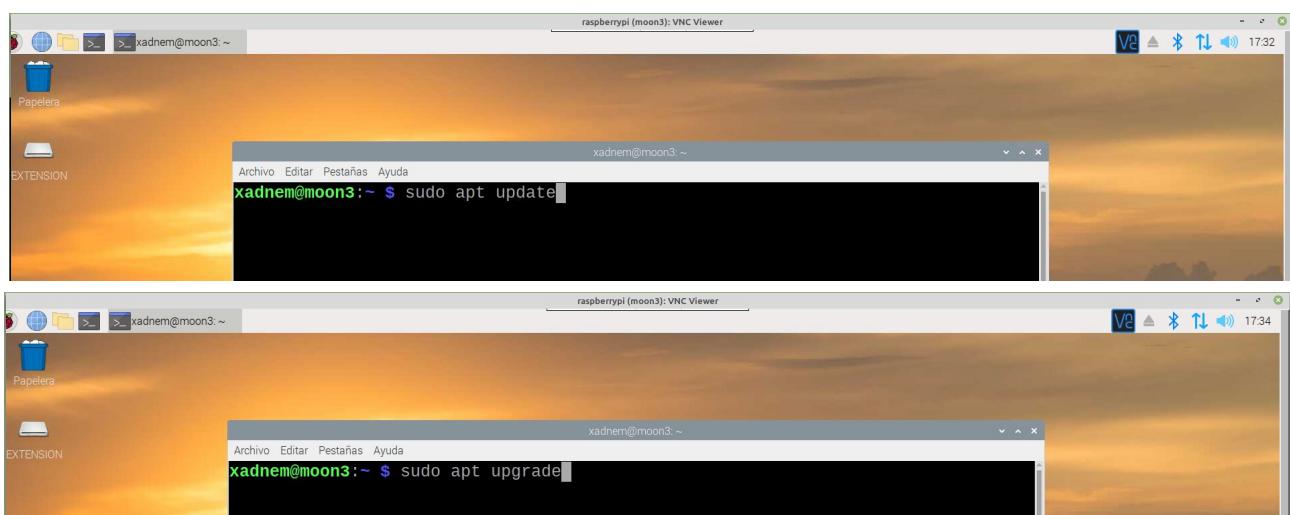
The screenshot shows the GbControl interface with a list of devices:

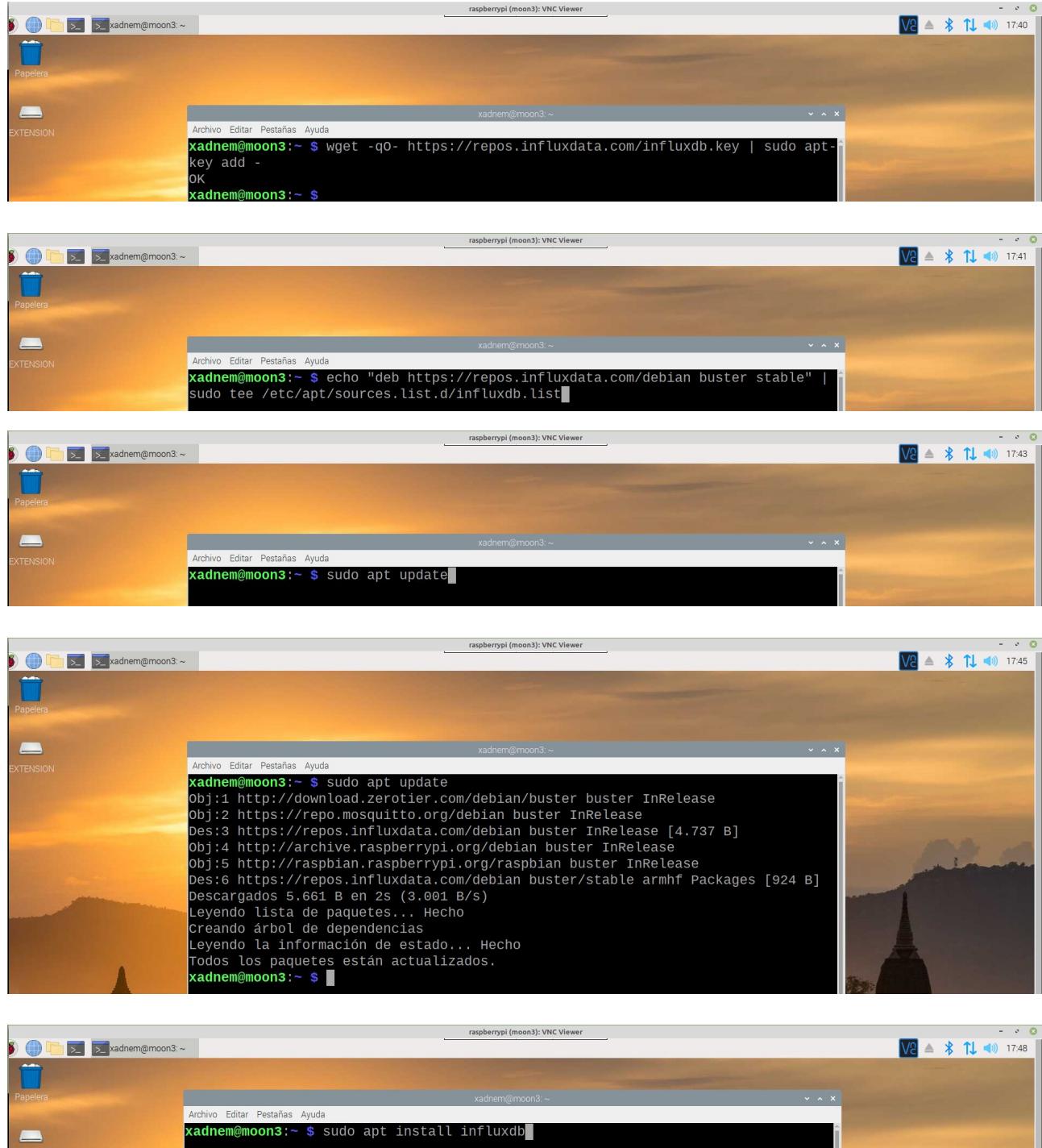
Auth?	Address	Name/Description	Managed IPs	Last Seen	Version	Physical IP
<input checked="" type="checkbox"/>	290383c273 f2:9a:f2:18:dc:e3	laptop El meu portatil	10.244.204.253 + 10.244.0.4	ONLINE	1.6.4	37.14.224.218
<input checked="" type="checkbox"/>	783b0a4d2c f2:c0:ca:91:85:b4	mobil el meu mobil	10.244.254.75 + 10.244.0.5	14D 14H 56M	1.6.4	37.14.224.218
<input checked="" type="checkbox"/>	8b71e4e5fa f2:38:00:7f:2b:60	Raspberry La meva Raspberry	10.244.59.176 + 10.244.0.6	ONLINE	1.6.4	37.14.224.218

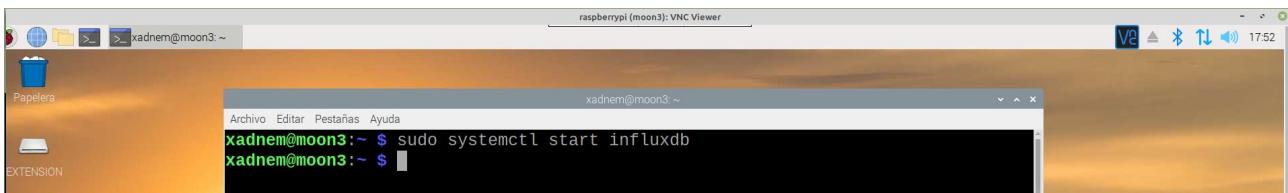
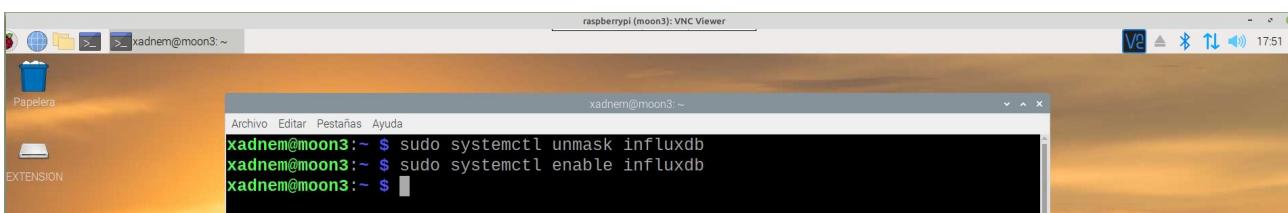
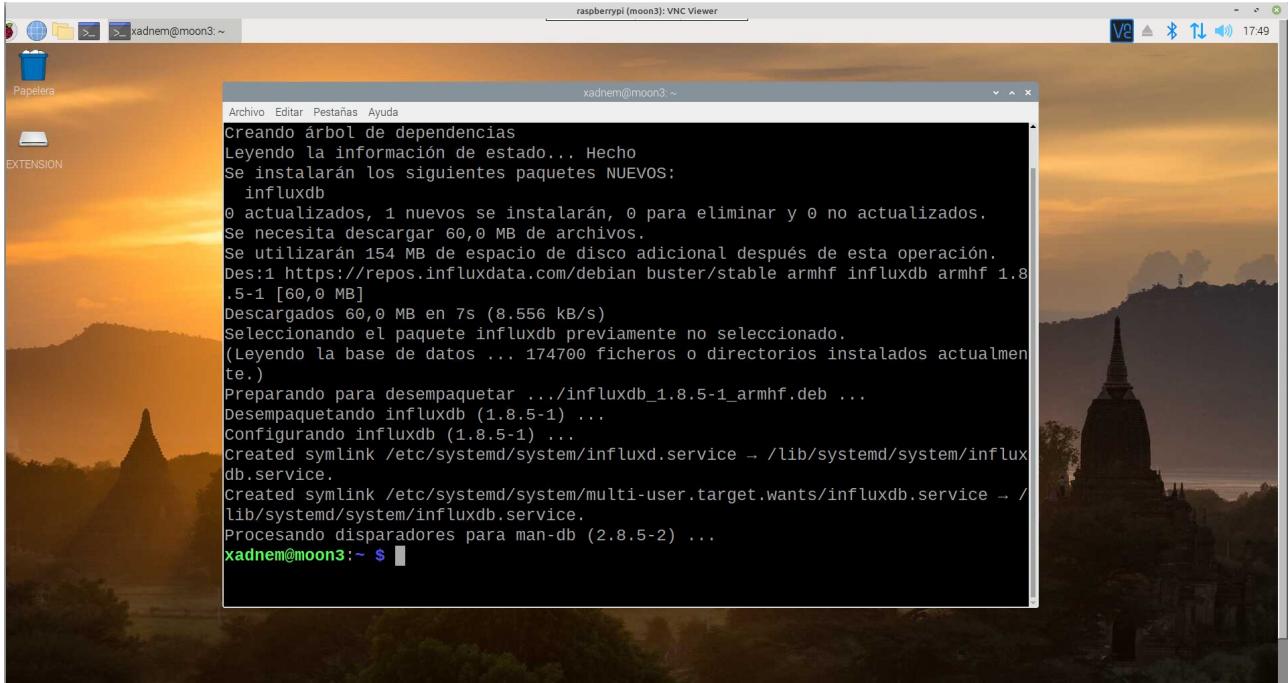
Below the device list, there are sections for "E-MAIL JOIN INSTRUCTIONS" (with an input field for "alice@example.com") and "MANUALLY ADD MEMBER" (with an input field for "#####"). Buttons for "Invite" and "Add New Member" are present. At the bottom left, there is a "Flow Rules" section.

## 4.8 Instal·lació d'InfluxDB i Grafana a la Raspberry

Vaig a emmagatzemar les dades a influxDB, per això tinc què instal·lar-lo a la Raspberry, i que les plaques ESP32 pugin les dades fent servir aquesta base de dades. He començat per a instal·lar influxDB a la Raspberry.

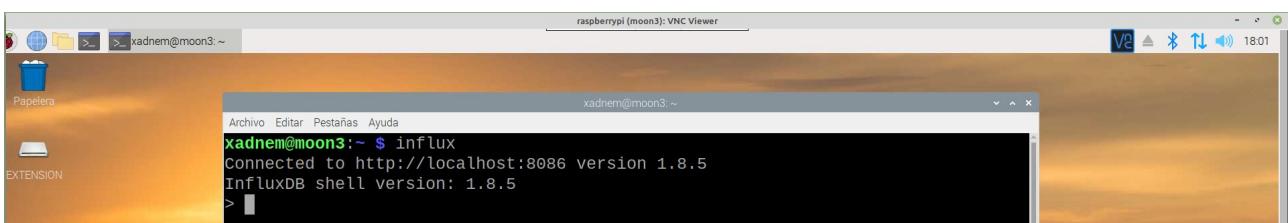




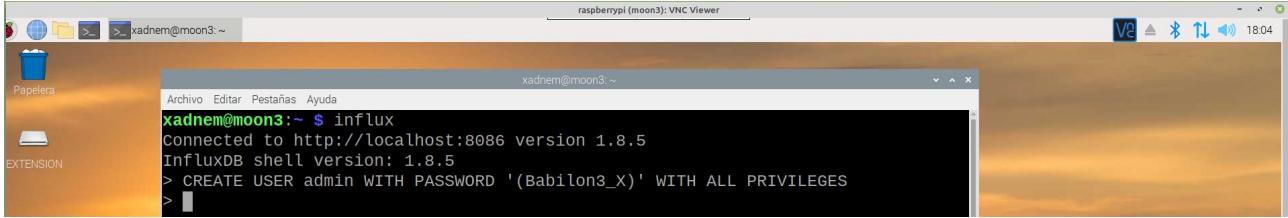


Per defecte, InfluxDB s'instal·la sense usuari ni password, el següent pas va estar configurar-los.

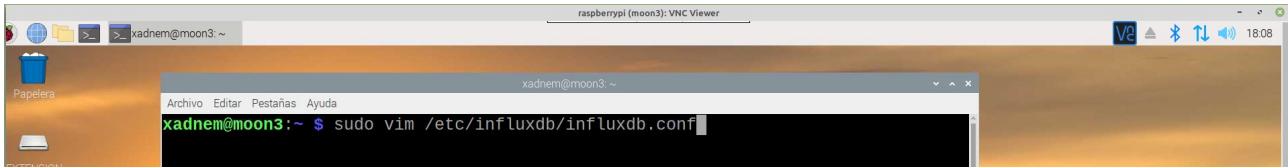
Primer s'ha de carregar l'eina influxdb CLI amb el següent comandament:



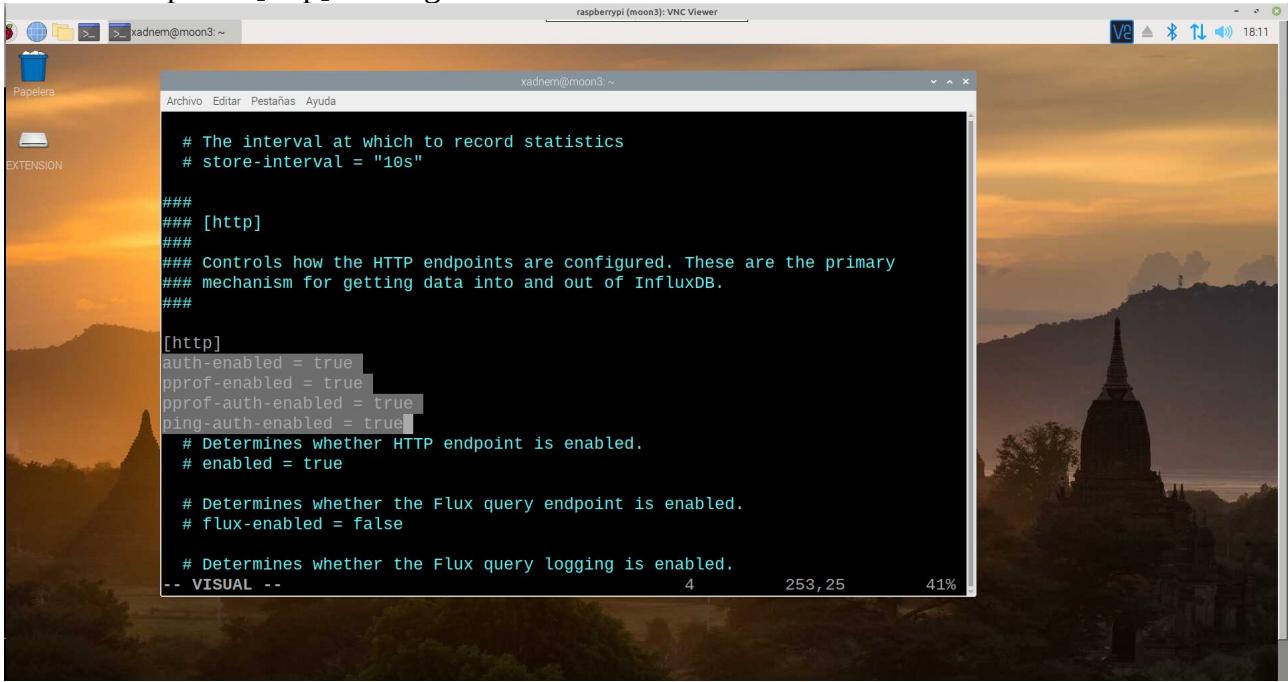
Després s'ha de crear un usuari «admin» amb un password.



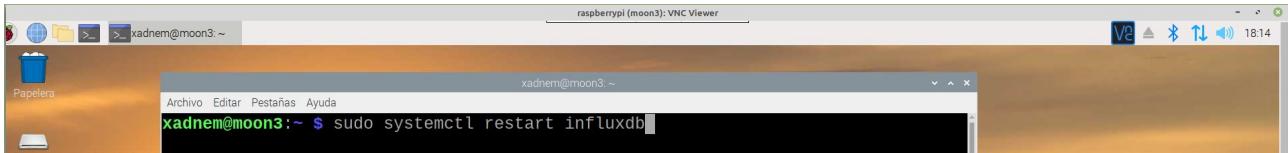
Per sortir de l'eina CLI s'ha de escriure «exit» i prémer enter.  
Després s'ha de modificar el fitxer config per a permetre l'autenticació.



A sota del apartat [http] he afegit:



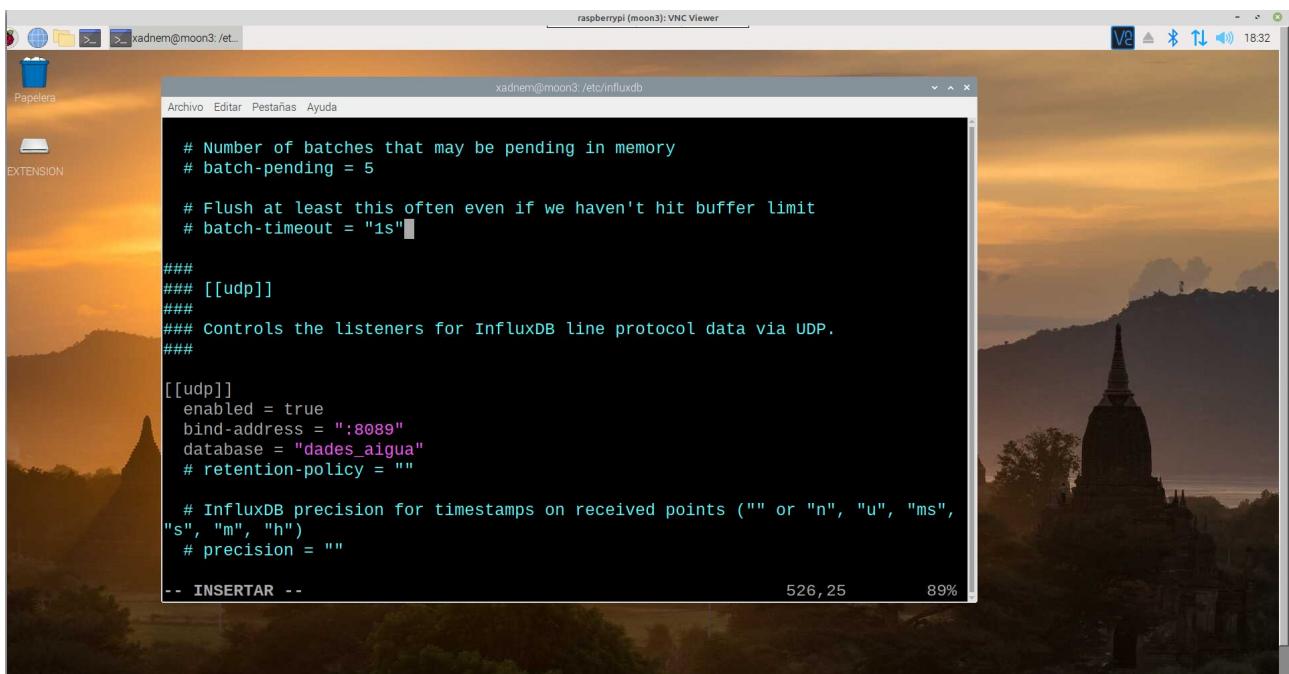
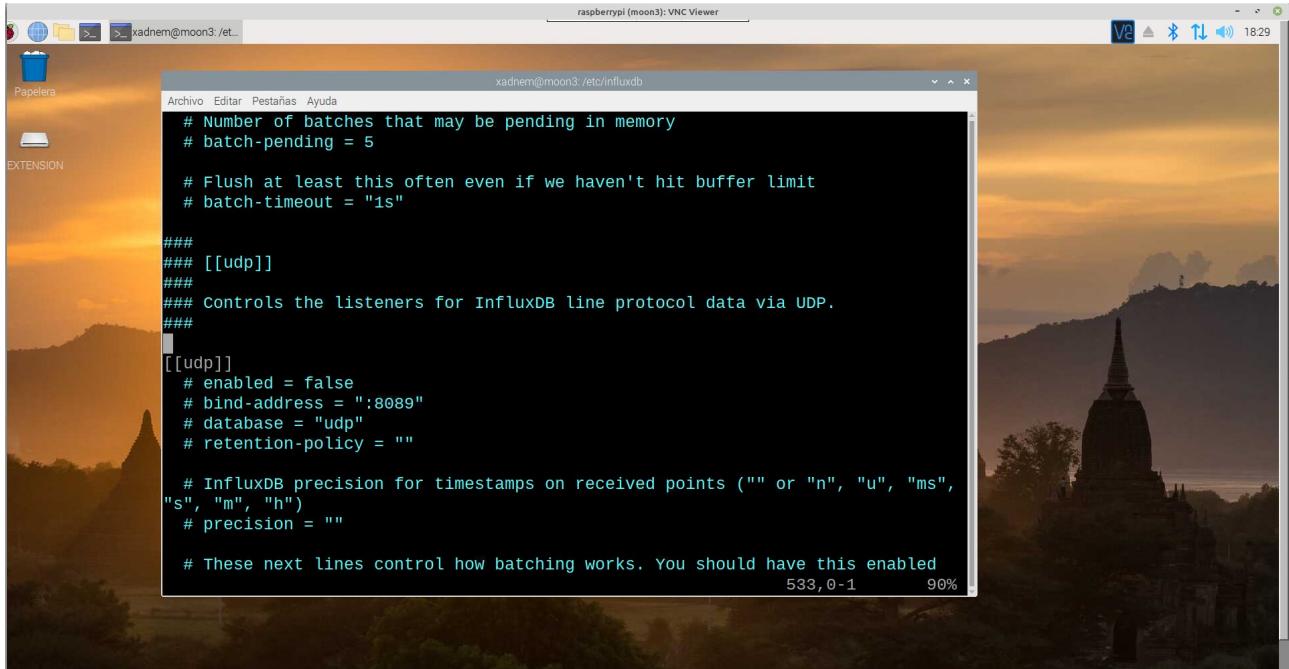
He reiniciat influxDB



Ara per accedir a la eina CLI d'influxDB és necessari posar usuari i password

**influx -username admin -password <password>**

Per a pujar les dades, faré servir el protocol UDP. Aquest protocol està deshabilitat per defecte al fitxer de configuració d'InfluxDB. He editat aquest fitxer i habilitat l'ús de UDP.



Per a provar la base de dades, he afegit alguns registres de prova.

```
xadnem@moon3:/etc/influxdb $ influx -username admin -password ' '
Connected to http://localhost:8086 version 1.8.5
InfluxDB shell version: 1.8.5
> create database dades_aigua
> use dades_aigua
Using database dades_aigua
> insert consum litres=10
> insert consum litres=100
> insert consum litres=30
> select * from consum
name: consum
time          litres
----          -----
1621186041214318193 10
1621186050686512457 100
1621186059961591652 30
> 
```

En aquesta prova, **dades\_aigua** es la base de dades, **consum** seria la taula i **litres** el nom del camp al que se li assigna el valor en SQL.

Al fer el select, veig que hi ha un camp **time** que influxDB afegeix per defecte. En aquest cas, m'ha donat el temps en microsegons des de l'unix time, ho qual no és intel·ligible per els humans.

Per obtindre la data i temps en format llegible pels humans s'ha de posar «precision rfc3339» al CLI.

```
xadnem@moon3:/etc/influxdb $ influx -username admin -password ' '
Connected to http://localhost:8086 version 1.8.5
InfluxDB shell version: 1.8.5
> use dades_aigua
Using database dades_aigua
> precision rfc3339
> select * from consum
name: consum
time          litres
----          -----
2021-05-16T17:27:21.214318193Z 10
2021-05-16T17:27:30.686512457Z 100
2021-05-16T17:27:39.961591652Z 30
> 
```

Encara que no la he utilitzat, he instal·lat la llibreria de python per a influxdb perquè m'ha semblat convenient tindre-la instal·lada ara que estic amb això.

```
xadhem@moon3:/etc/influxdb $ pip3 install influxdb
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting influxdb
  Downloading https://files.pythonhosted.org/packages/01/95/3a72ea5e19df828d27af0a50092f2b24114a5f89922efb4d8a0960bf13ef/influxdb-5.3.1-py2.py3-none-any.whl (77 KB)
    100% |██████████| 81kB 1.7MB/s
Requirement already satisfied: requests>=2.17.0 in /usr/lib/python3/dist-packages (from influxdb) (2.21.0)
Requirement already satisfied: python-dateutil>=2.6.0 in /usr/lib/python3/dist-packages (from influxdb) (2.7.3)
Collecting pytz (from influxdb)
  Downloading https://files.pythonhosted.org/packages/70/94/784178ca5dd892a98f113cdd923372024dc04b8d40abe77ca76b5fb90ca6/pytz-2021.1-py2.py3-none-any.whl (510kB)
    100% |██████████| 512kB 721kB/s
Collecting msgpack (from influxdb)
  Downloading https://www.piwheels.org/simple/msgpack/msgpack-1.0.2-cp37-cp37m-linux_armv7l.whl (267kB)
    100% |██████████| 276kB 769kB/s
Requirement already satisfied: six>=1.10.0 in /usr/lib/python3/dist-packages (from influxdb) (1.12.0)
Installing collected packages: pytz, msgpack, influxdb
Successfully installed influxdb-5.3.1 msgpack-1.0.2 pytz-2021.1
xadhem@moon3:/etc/influxdb $
```

He escrit un petit programa python per a provar la possibilitat d'accendir a les dades.

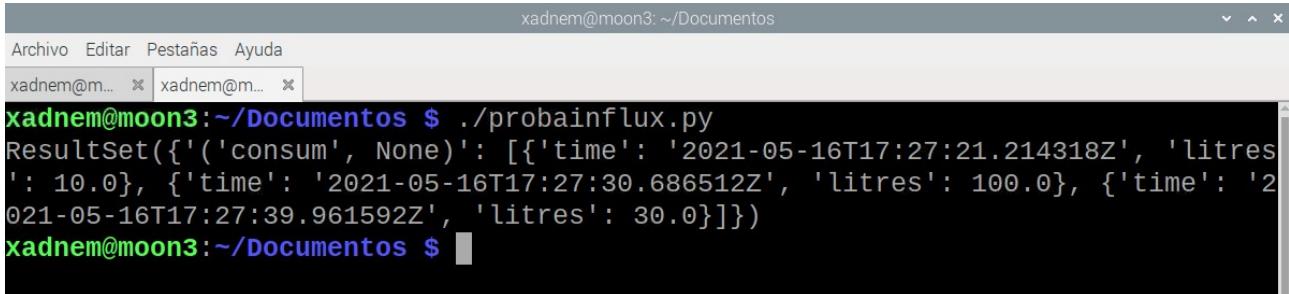
```
#!/usr/bin/python3.7
from influxdb import InfluxDBClient

def main(host='localhost', port=8086):
    """Instantiate a connection to the InfluxDB."""
    user = 'admin'
    password = '_____'
    dbname = 'dades_aigua'
    query = 'select * from consum'
    bind_params = {'host': 'server01'}
    client = InfluxDBClient(host, port, user, password, dbname)
    result = client.query(query)
    print(result)

main()
```

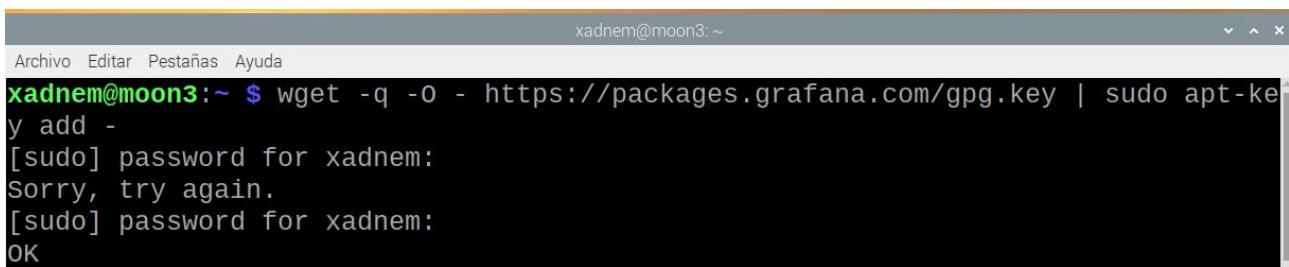
Shell  
Python 3.7.3 (/usr/bin/python3)  
>>>

Al executar-lo veig que obtinc les dades.

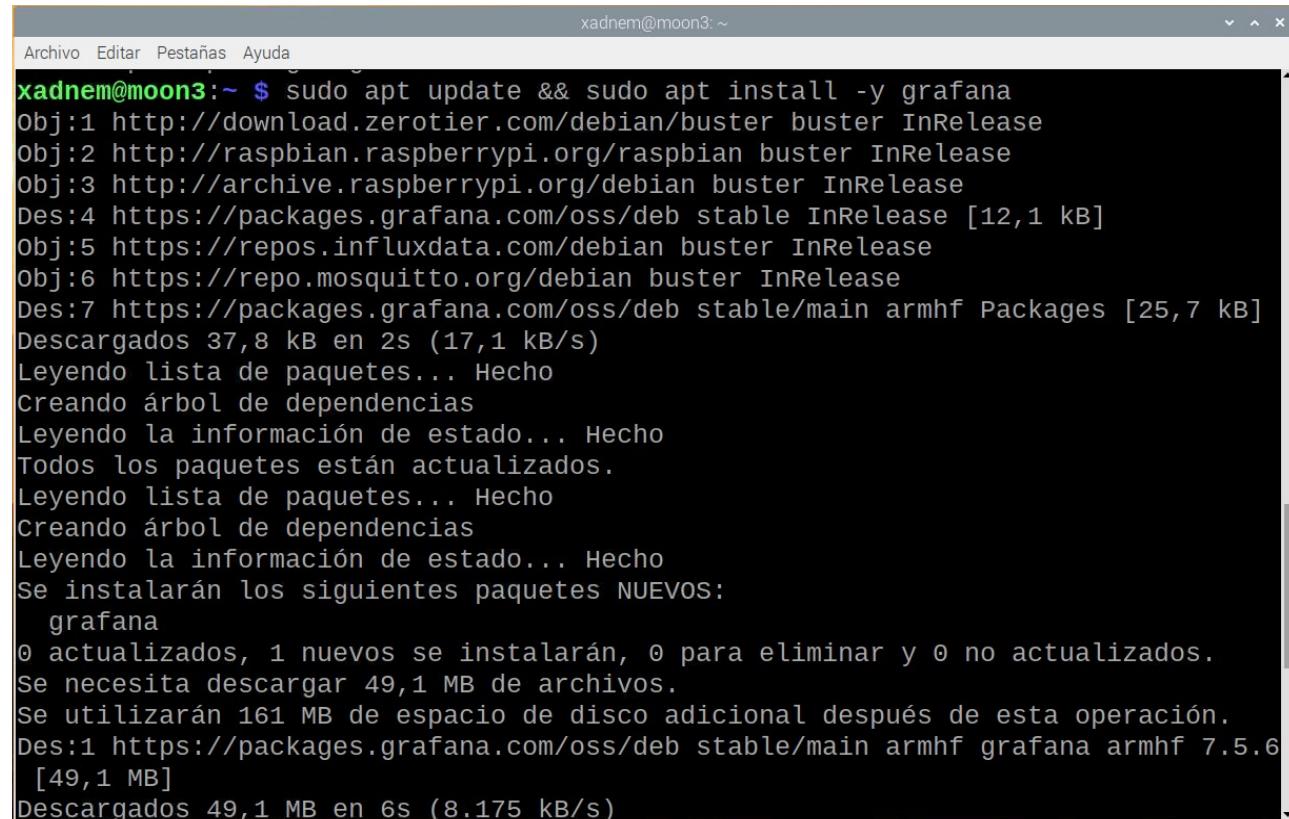


```
xadnem@moon3:~/Documentos$ ./probainflux.py
ResultSet([('consum', None): [ {'time': '2021-05-16T17:27:21.214318Z', 'litres': 10.0}, {'time': '2021-05-16T17:27:30.686512Z', 'litres': 100.0}, {'time': '2021-05-16T17:27:39.961592Z', 'litres': 30.0}]])
xadnem@moon3:~/Documentos$
```

## Instal·lar Grafana i connectar-lo amb InfluxDB



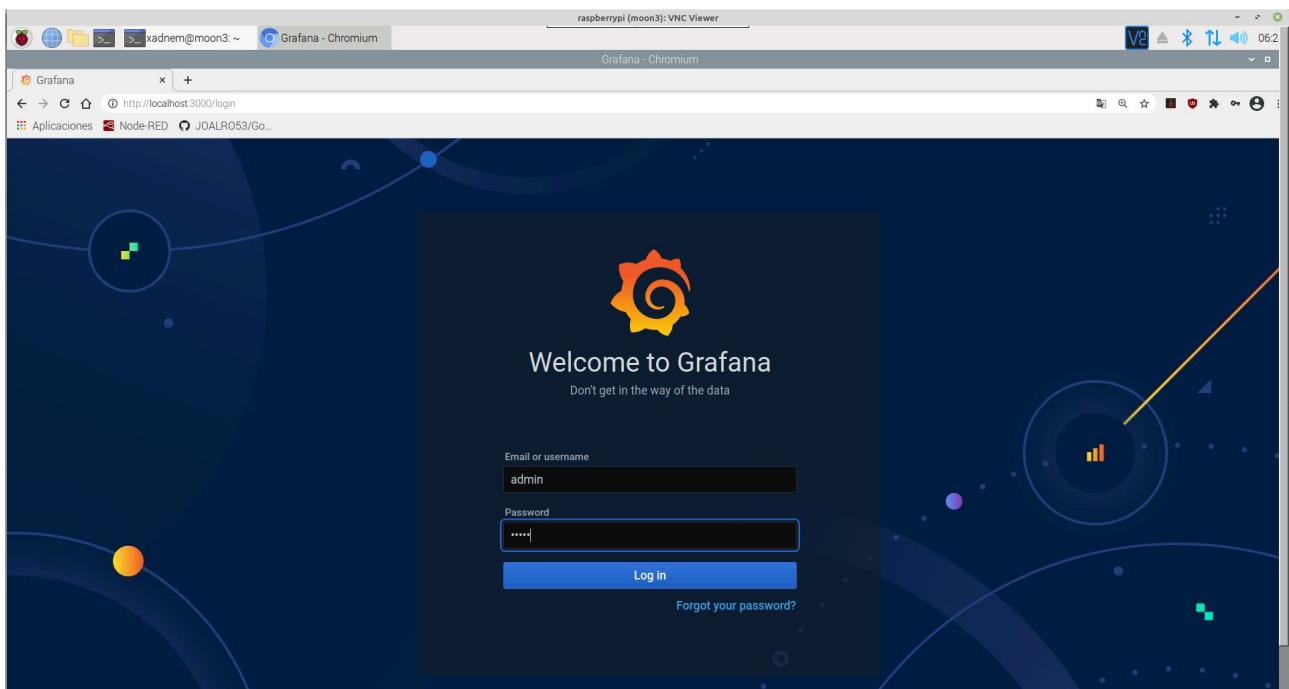
```
xadnem@moon3:~$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
[sudo] password for xadnem:
Sorry, try again.
[sudo] password for xadnem:
OK
```



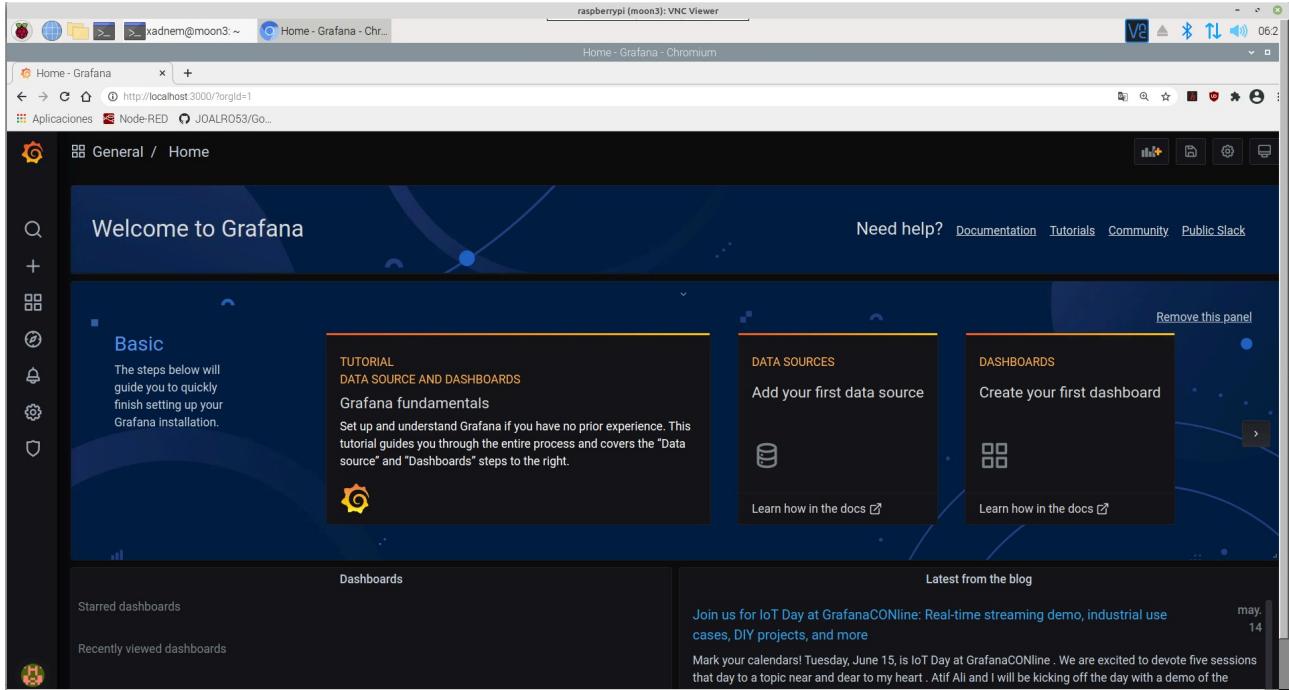
```
xadnem@moon3:~$ sudo apt update && sudo apt install -y grafana
Obj:1 http://download.zerotier.com/debian/buster buster InRelease
Obj:2 http://raspbian.raspberrypi.org/raspbian buster InRelease
Obj:3 http://archive.raspberrypi.org/debian buster InRelease
Des:4 https://packages.grafana.com/oss/deb stable InRelease [12,1 kB]
Obj:5 https://repos.influxdata.com/debian buster InRelease
Obj:6 https://repo.mosquitto.org/debian buster InRelease
Des:7 https://packages.grafana.com/oss/deb stable/main armhf Packages [25,7 kB]
Descargados 37,8 kB en 2s (17,1 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  grafana
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 49,1 MB de archivos.
Se utilizarán 161 MB de espacio de disco adicional después de esta operación.
Des:1 https://packages.grafana.com/oss/deb stable/main armhf grafana armhf 7.5.6
 [49,1 MB]
Descargados 49,1 MB en 6s (8.175 kB/s)
```

```
xadnem@moon3:~ $ sudo systemctl unmask grafana-server.service
xadnem@moon3:~ $ sudo systemctl start grafana-server
xadnem@moon3:~ $ sudo systemctl enable grafana-server.service
Synchronizing state of grafana-server.service with sysV service script with /lib
/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.servi
ce → /lib/systemd/system/grafana-server.service.
xadnem@moon3:~ $
```

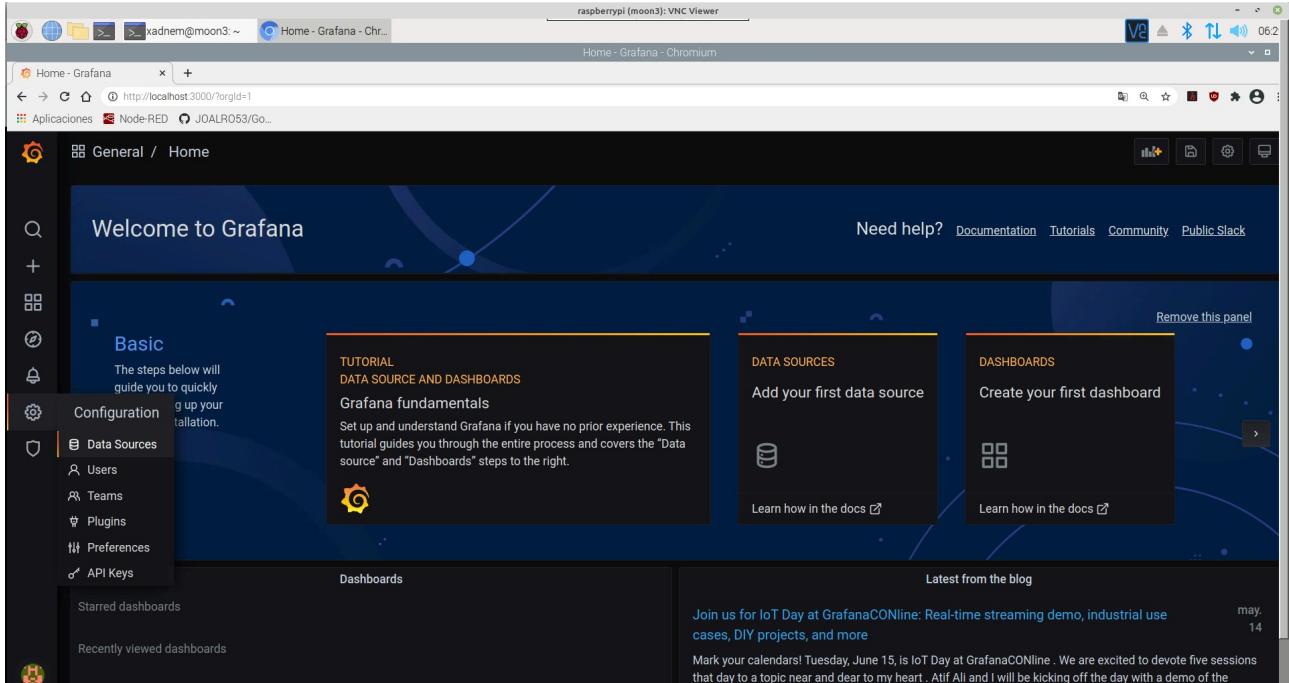
user = admin, password = admin



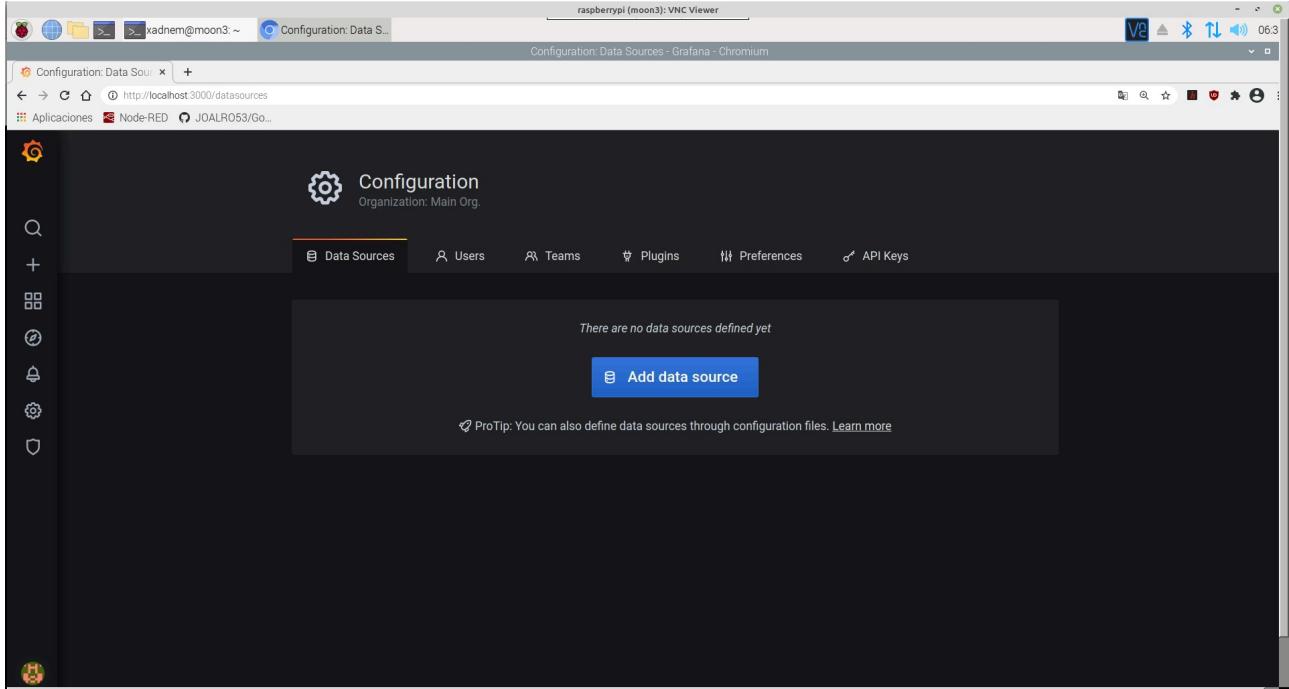
Despres password definitiu



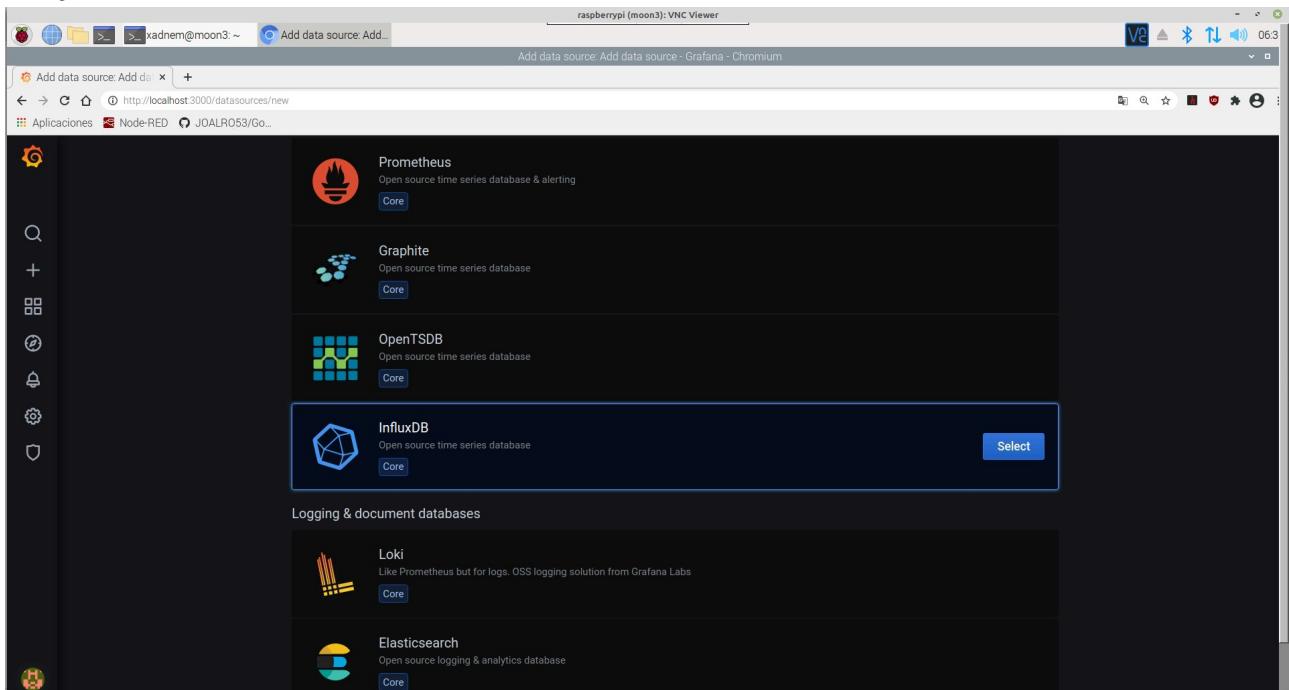
## Data sources

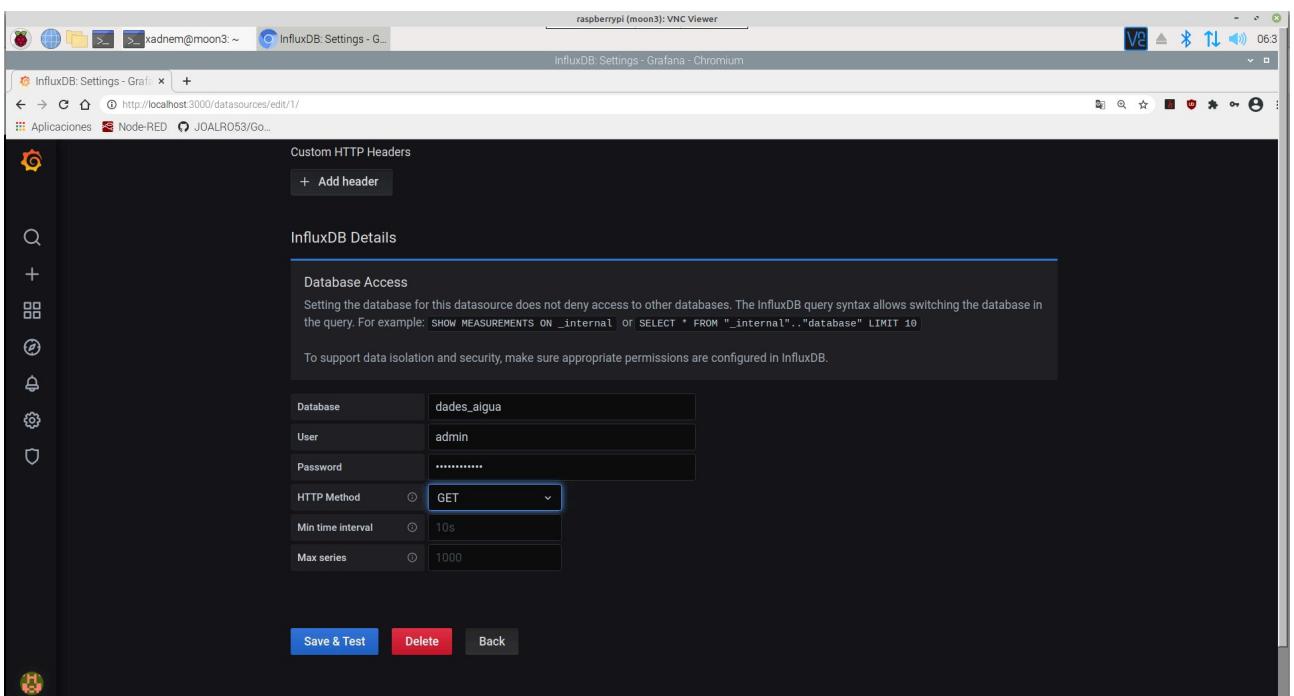
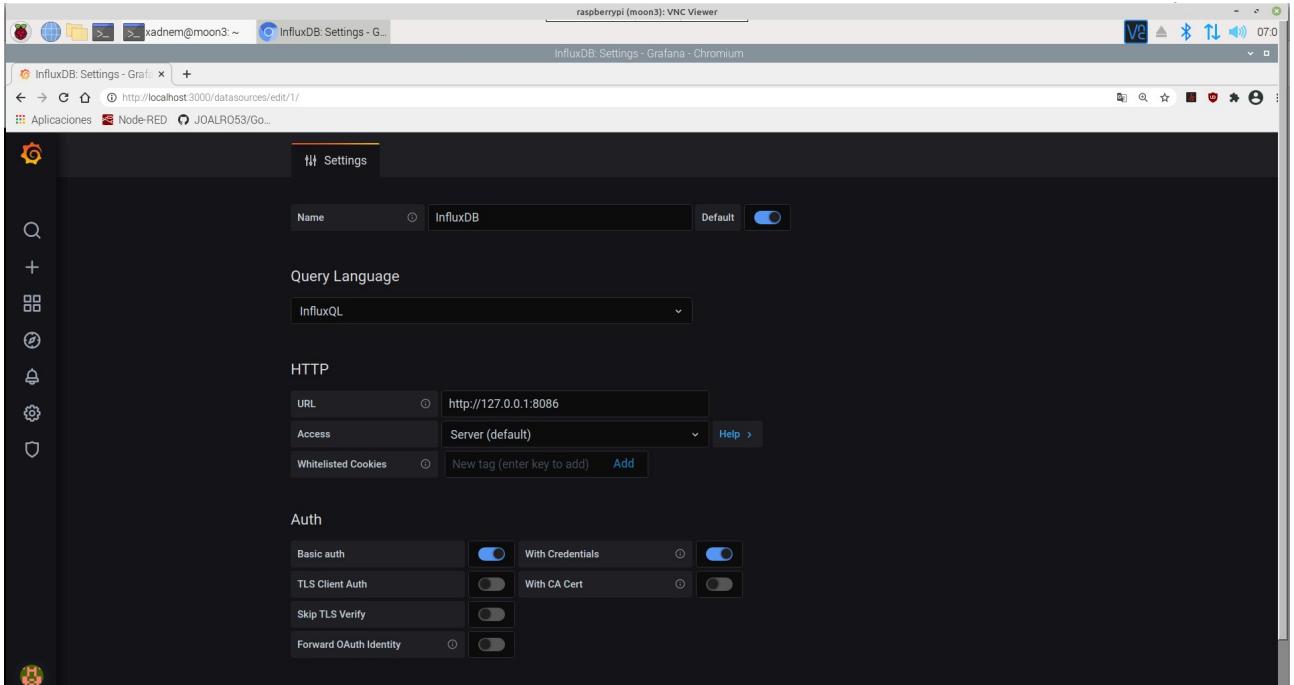


## Add data source

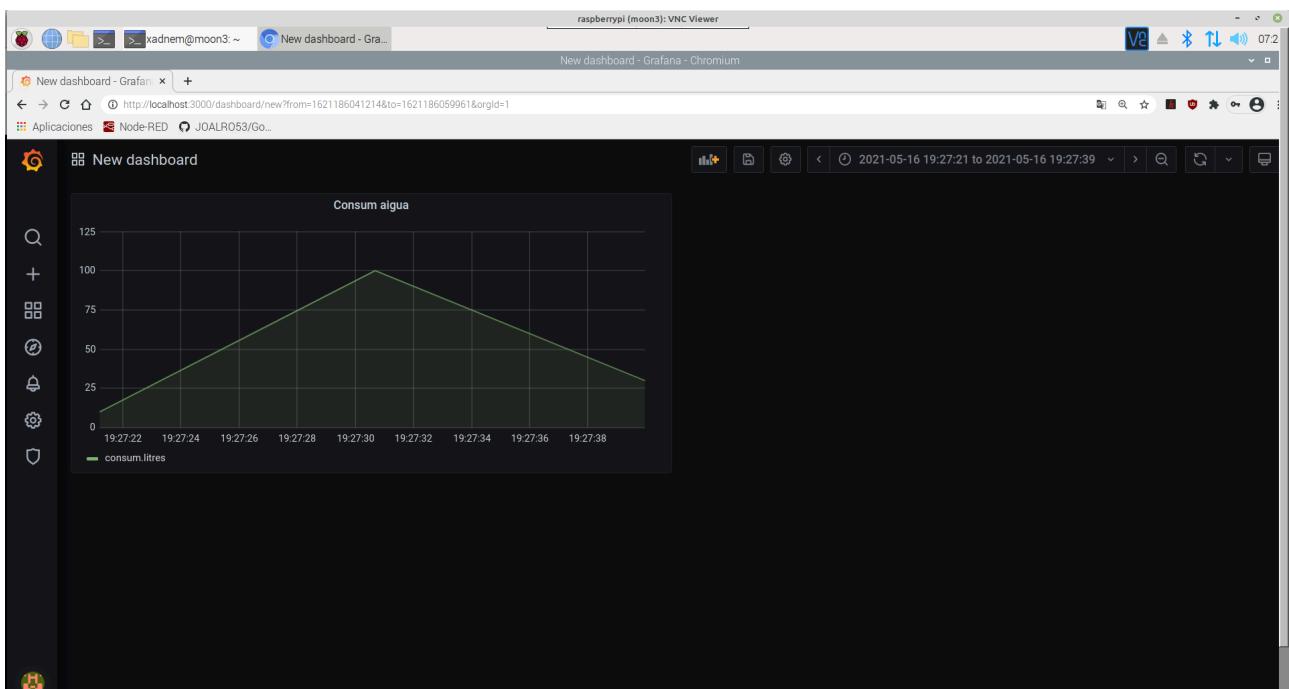


## InfluxDB





The screenshot shows the 'InfluxDB Details' configuration page in Grafana. The 'Database' field is set to 'dades\_aigua', 'User' to 'admin', and 'Password' to 'configured'. The 'HTTP Method' is set to 'GET', 'Min time interval' to '10s', and 'Max series' to '1000'. A green success message at the bottom states 'Data source is working'. Below the form are 'Save & Test', 'Delete', and 'Back' buttons.



## 5

# Implantacio i proves



## 5.1 Implantació

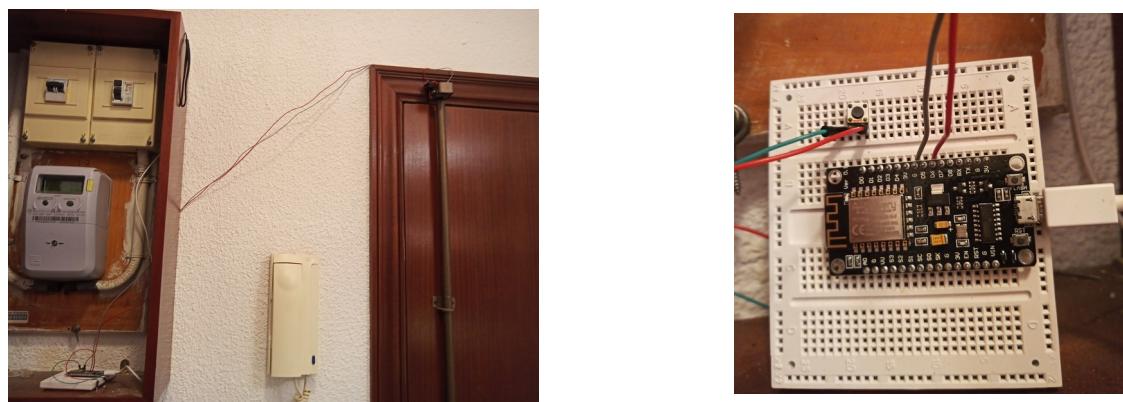
### 5.1.1 Implantació del Sistema de control d'accés

La primera qüestió per a implementar aquest sistema, va ser com detectar el tancament del pany de la porta.

Aprofitant què el pany del meu habitatge té una barra d'ancoratge que puja quan el pany es tanca i baixa quan s'obre, vaig muntar un botó què fos accionat per aquesta barra.



Vaig connectar el botó amb la placa esp8266 amb un cable, ( de una manera burda s'ha de dir, però per a fer proves reals no calia que fos estètic !!!)



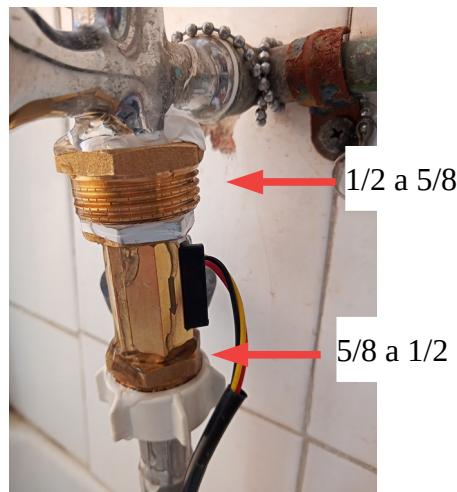
He alimentat la esp8266 amb un carregador de mòbil vell que tenia per casa. Tot llest per a la primera prova real !!!

## 5.1.2 Implementació del sistema de mesurament de consum d'aigua

Per fer un control total del consum d'aigua del meu habitatge, hauria tingut que connectar el sensor a la canonada de presa d'aigua. Com els meus coneixements de fontaneria són molt bàsics, vaig decidir posar el sensor a la presa d'aigua de la rentadora, ja que era relativament fàcil.

El més difícil va ser aconseguir els ràcords per a passar dels 5/8 de polzada de la rosca del sensor, al 1/2 de polzada de la presa d'aigua de la rentadora.

Vaig aconseguir els ràcords amb una mica de treball d'investigació i aquest va ser el resultat



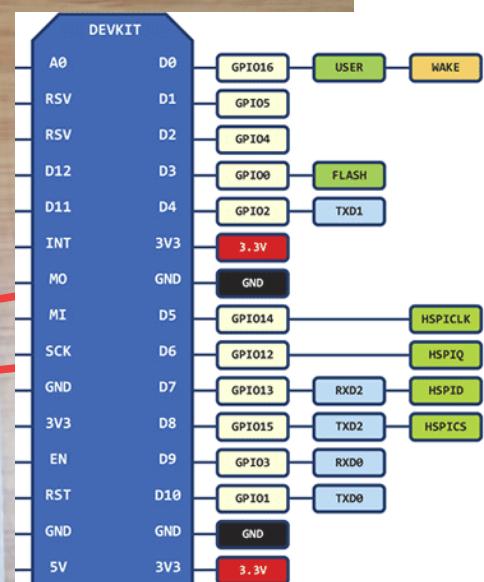
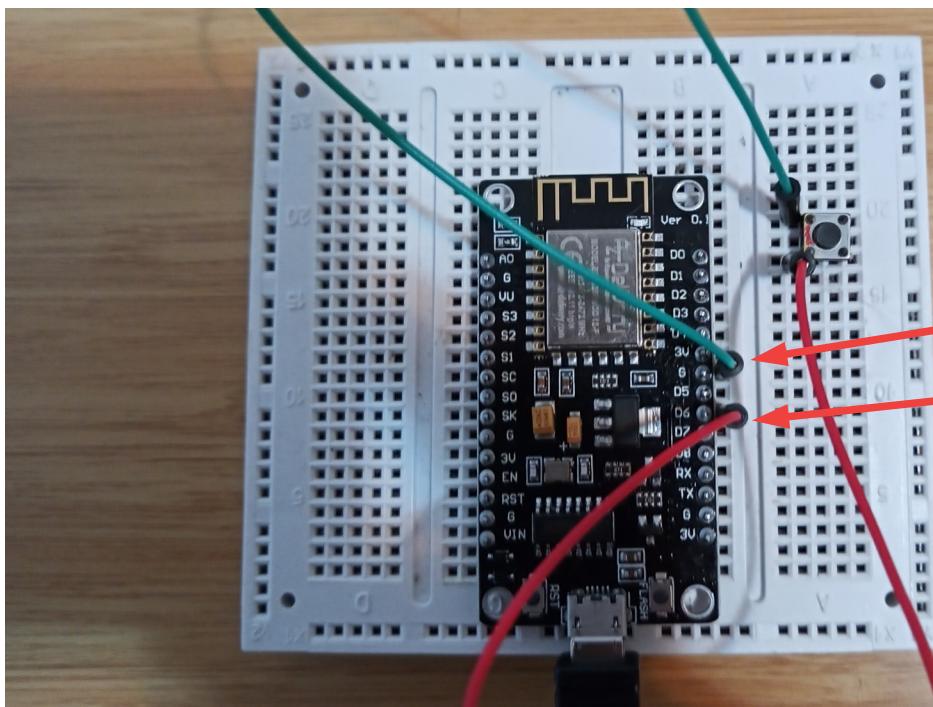
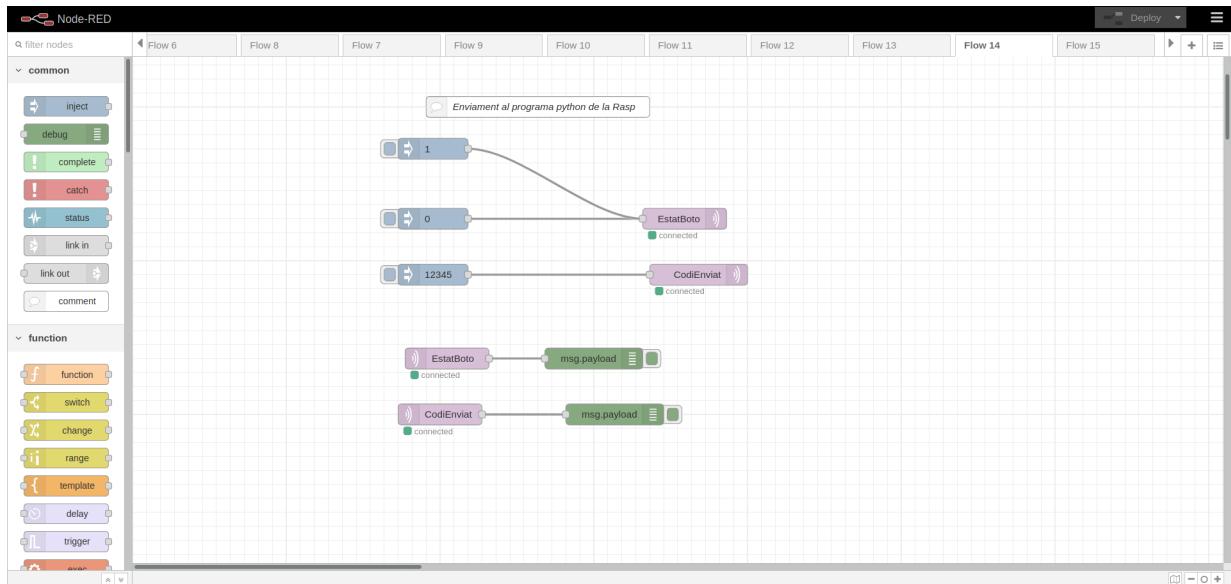
Val d'acord, una mica chapu però de moment suficient !!!

## 5.2 Proves

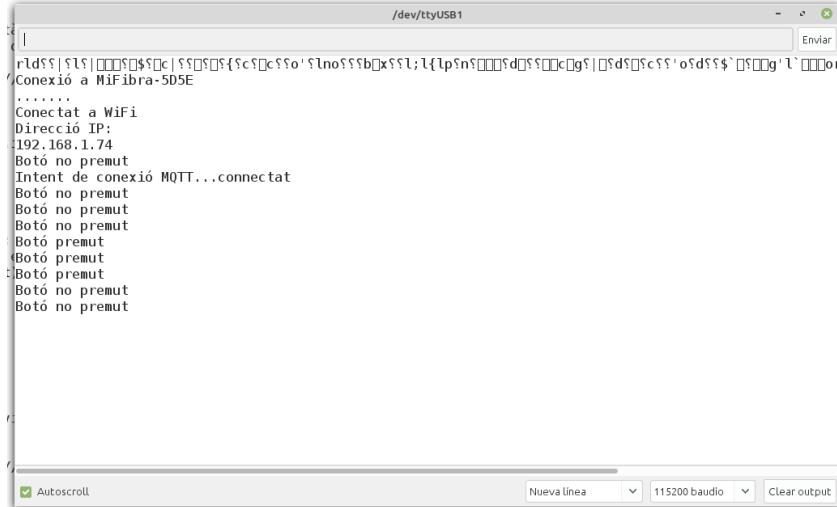
### 5.2.1 Proves inicials

Per a provar la rebuda i transmissió de missatges mqtt per part del programa GbControl i el programa Android del telèfon, vaig crear un senzill flow de node-red.

Després de escriure el programa PulsacioBoto, vaig carregar-lo a la placa esp8266 i vaig fer proves en una protoboard.



Vaig veure que el programa funcionava correctament.

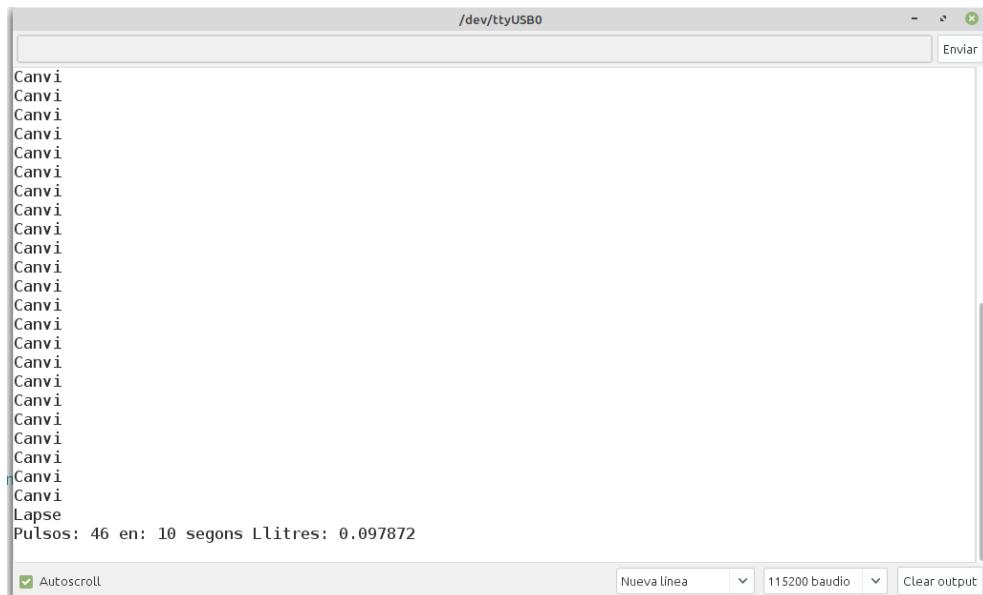


```

Conexió a MiFibra-5D5E
.....
Conectat a WiFi
Direcció IP:
192.168.1.74
Botó no premut
Intent de conexió MQTT...connectat
Botó no premut
Botó no premut
Botó no premut
Botó premut
Botó premut
Botó premut
Botó premut
Botó no premut
Botó no premut

```

Per provar el programa MedicioAigua vaig connectar el sensor a la placa en una protoboard i vaig bufar pel sensor per a simular el pas d'aigua. Vaig veure pel monitor serie que es detectaven els pulsos enviats pel sensor i es feia el recomte i calcul dels litres correctament.



```

Canvi
Lapse
Pulsos: 46 en: 10 segons Llitres: 0.097872

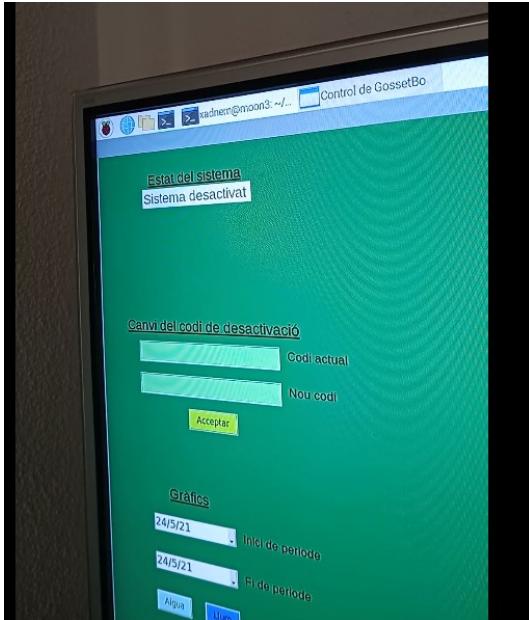
```

## 5.2.2 Proves reals

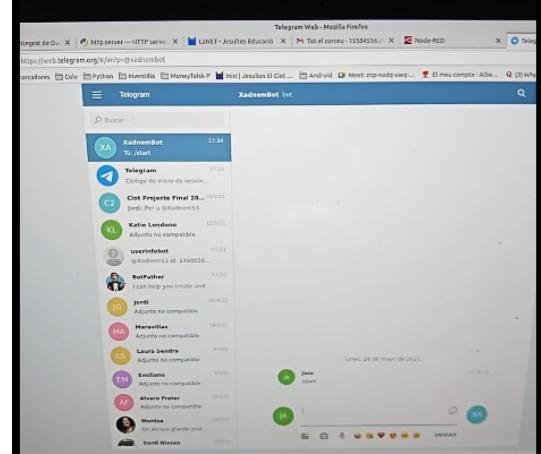
### Proves del sistema de control d'accés

Aquesta seqüència de fotogrames, està estreta del vídeo que vaig preparar a una de les proves reals del sistema d'accés adjunt a aquest dossier.

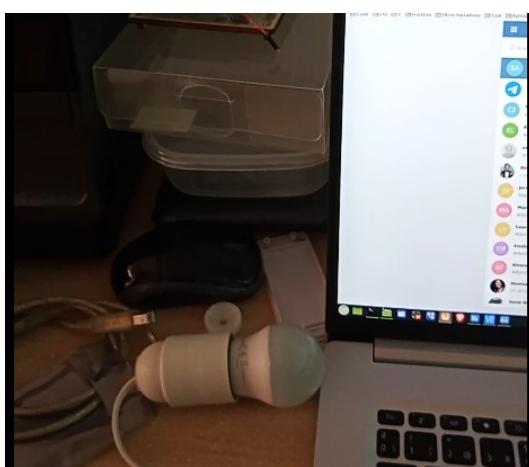
El sistema està desactivat



El bot de Telegram està buit



El Sonoff està apagat

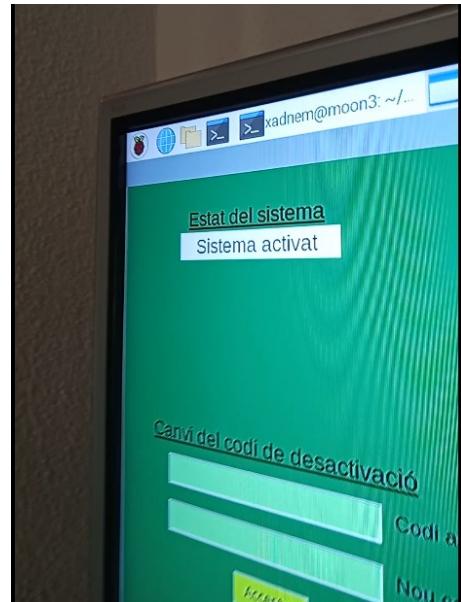


1

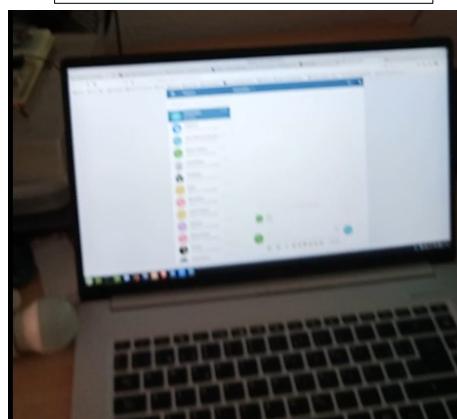
Tanco el pany de la porta



El sistema s'activa



El bot continua buit i el  
Sonoff apagat

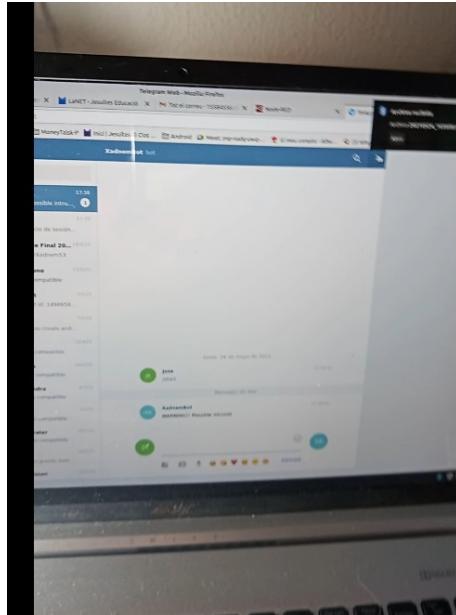


2

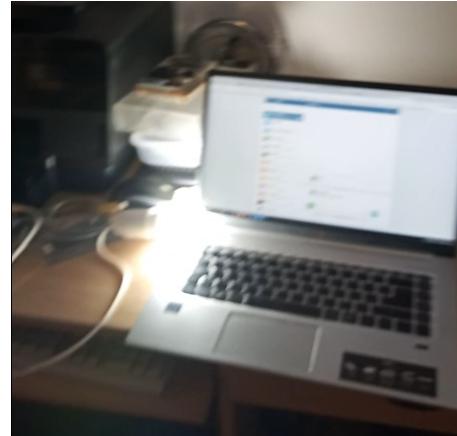
Obro el pany sense introduir  
El codi de desactivació



El Bot ha rebut el missatge  
WARNING

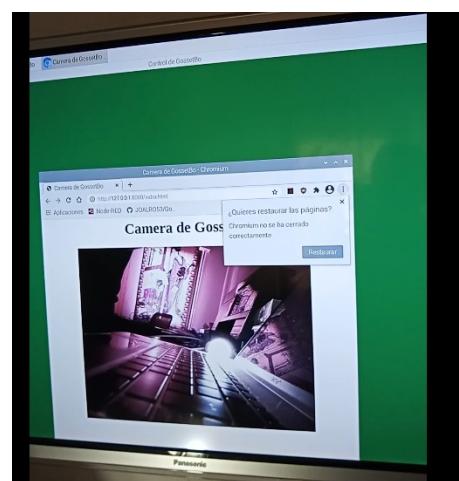
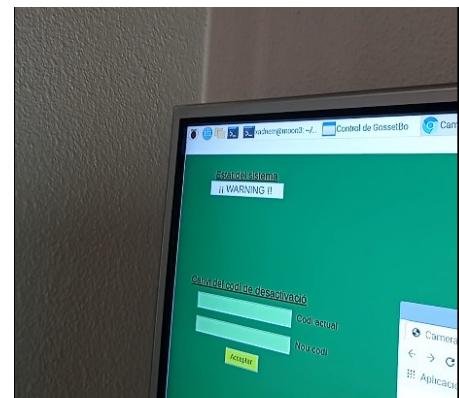


El Sonoff s'ha activat

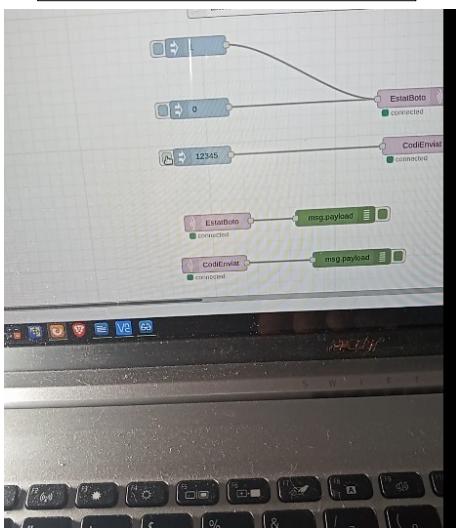


La GUI mostra el  
WARNING i el navegador  
amb la càmera

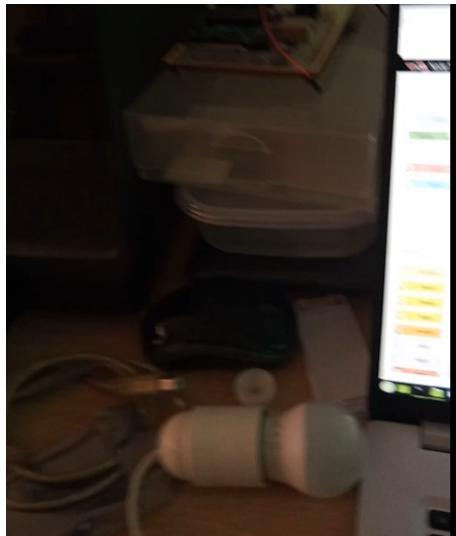
3



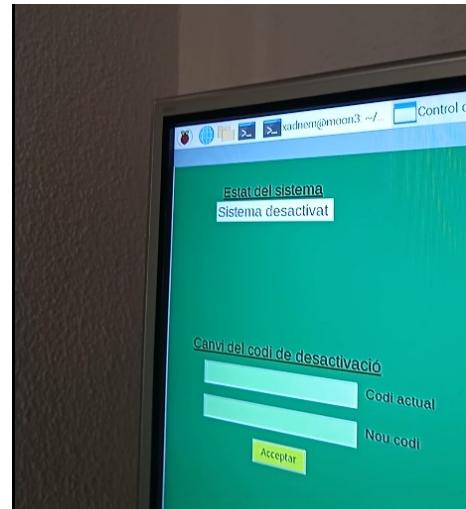
Envio el codi de desactivació



El Sonoff es desactiva



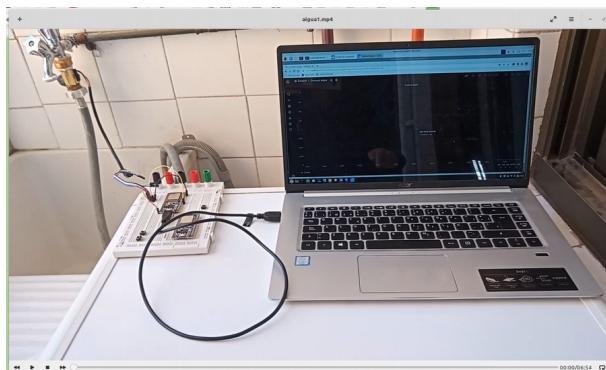
El sistema es desactiva el  
Navegador es tanca i la camera  
Deixa de grabar



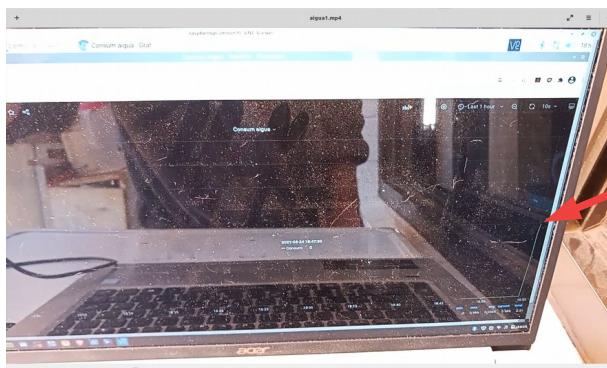
4

## Proves del sistema de control de consum d'aigua

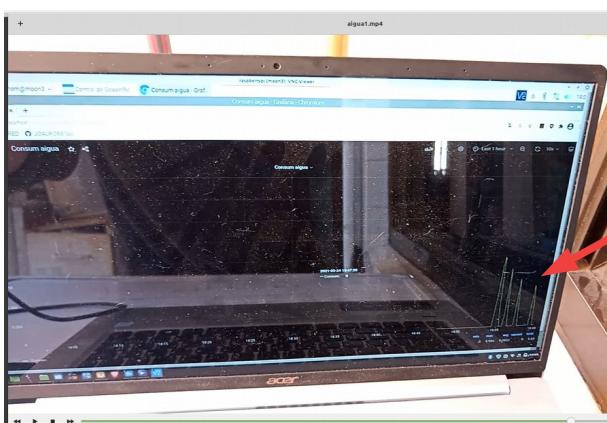
Aquesta seqüència de fotogrames, està estreta del vídeo que vaig preparar a una de les proves reals del sistema de consum d'aigua adjunt a aquest dossier.



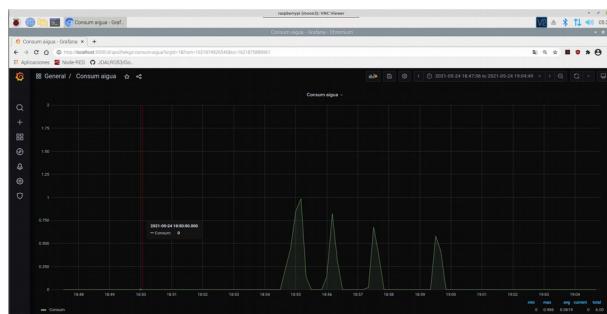
Inici de la prova



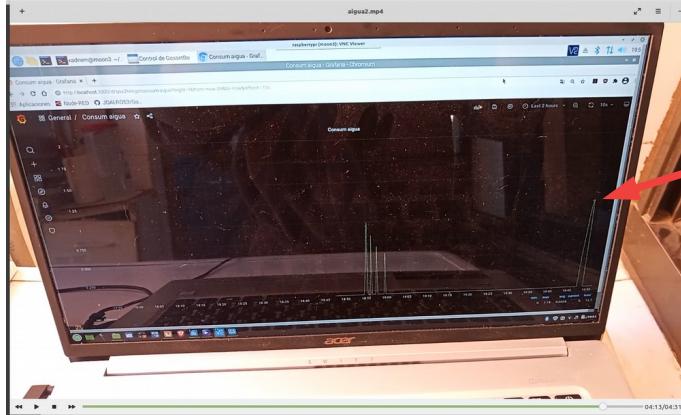
La rentadora comença la Presa d'aigua



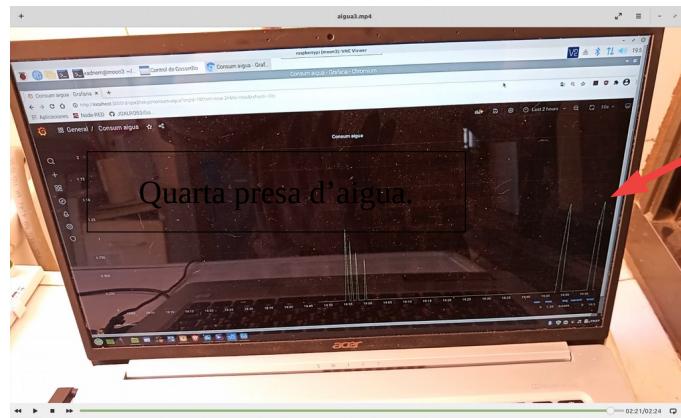
Finalitza la primera Presa d'aqua



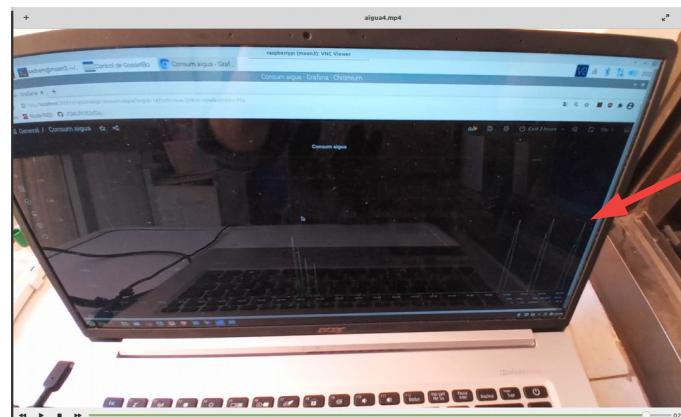
Resultat de la primera Presa d'aqua



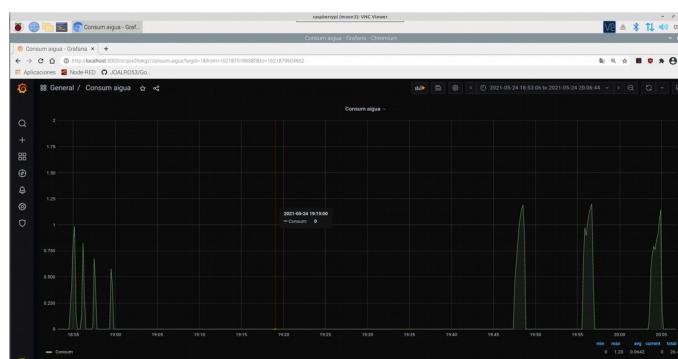
Segona presa d'aigua.



Tercera presa d'aigua.



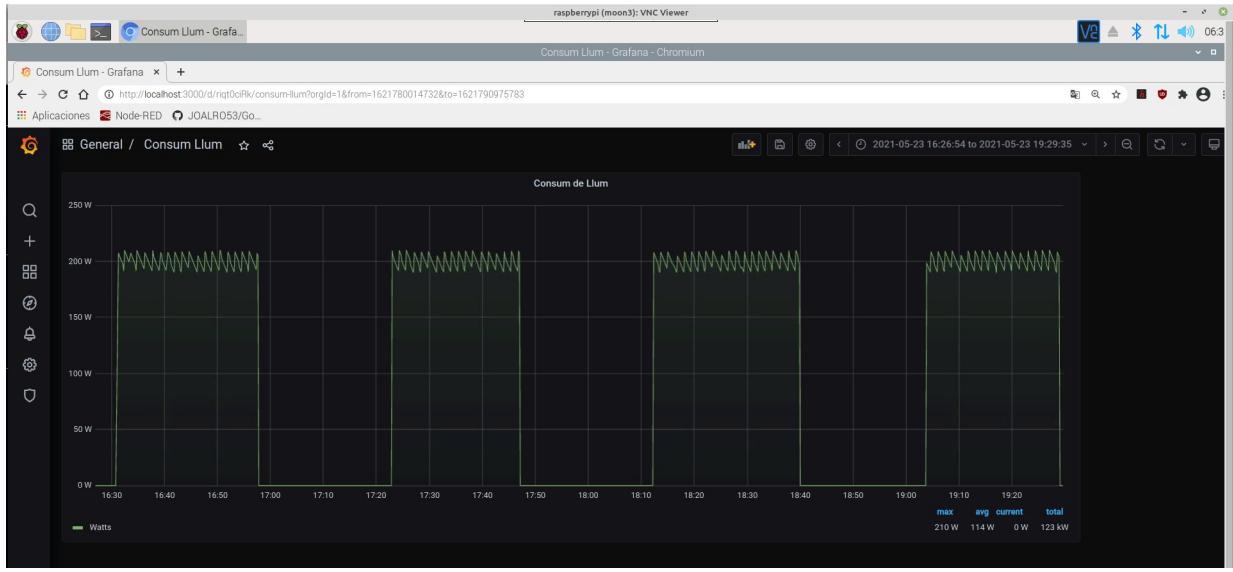
Quarta presa d'aigua.



Resultat final

## Proves del sistema de control de consum d'Llum

Aquest gràfic és el producte de la simulació de consum de llum d'un frigorífic portada a terme pel programa Medicio\_Llum.ino



# 6

## Conclusions i enllaços



## 6.1 Conclusions

### 6.1.1 Sistema de control d'accés

En general, crec sincerament què el sistema compleix força bé les expectatives, si bé es susceptible d'algunes millores, com:

- Detectar el estat del pany de la porta per un sistema més fiable, a ser possible no mecànic.
- Fer una versió per a dispositiu mòbil de GbControl on s'integri el visualitzat de la càmera ,l'enviament del codi de desactivació i la consulta als gràfics de consum d'aigua i Llum.
- Fer que la esp8266 s'apagui quan el sistema estigui desactivat.

### 6.1.2 Sistema de control del consum d'aigua

Encara què m'ha servit com a primer contacte amb el sensor i pel desenvolupament del programa de control del mateix, tal i com està ara, el sistema es d'escassa utilitat.

Si, ha estat curiós veure el consum de la meva rentadora i com fa les presses d'aigua, però vist una vegada, vistes totes.

El següent pas serà provar altres sensors i fer la instal·lació del sensor a la canonada de presa d'aigua general del meu habitatge, això si, amb l'ajut d'un professional i de manera que el sensor sigui fàcilment substituïble.

### 6.1.3 Sistema de control del consum de llum

Aquesta es una assignatura pendent. Com he explicat abans, aquest sensor és per un amperatge massa gran per al consum domèstic.

A més, he de millorar els meus coneixements sobre electricitat i electrònica per a fer una implementació mínima que hem permeti fer unes proves representatives.

D'altra banda, puc explorar altres tipus de sensors. El què si tinc clar, es què no llenço la tovallola. Del fracàs es poden i es deuen treure experiències molt valuoses.

## 6.2 Enllaços

### Mqtt

[www.steves-internet-guide.com/install-mosquitto-linux/](http://www.steves-internet-guide.com/install-mosquitto-linux/)

<https://www.sourcerebels.com/blog/2018/10/consumir-mensajes-mqtt-desde-android/>

<https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/>

<https://ricveal.com/blog/sonoff-mqtt>

<https://domoticaencasa.es/tasmota-sonoff-esp8266-arduino-ide/>

<https://minibots.wordpress.com/2017/11/26/ejemplo-de-utilizacion-del-protocolo-mqtt-con-esp8266/>

### Video amb Raspberry

<https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/>

<https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/4>

### Telegram

<https://www.flopy.es/crea-un-bot-de-telegram-para-tu-raspberry-ordenale-cosas-y-habla-con-ella-a-distancia/>

## NodeMCU

<https://www.luisllamas.es/guia-de-programacion-del-esp8266-en-entorno-arduino/>

<https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/>

<https://www.electronicwings.com/nodemcu/nodemcu-gpio-interrupts-with-arduino-ide>

## Sensor de corrent SCT-013

<https://programarfacil.com/blog/arduino-blog/sct-013-consumo-electrico-arduino/>

<https://www.luisllamas.es/arduino-sensor-corriente-sct-013/>

## Sensor caudalimetre

<https://www.luisllamas.es/caudal-consumo-de-agua-con-arduino-y-caudalimetro/>

## InfluxDB i Grafana

<https://simonhearne.com/2020/pi-influx-grafana/>

<https://influxdb-python.readthedocs.io/en/latest/examples.html>

<https://pimylifeup.com/raspberry-pi-influxdb/>

<https://www.influxdata.com/blog/how-to-send-sensor-data-to-influxdb-from-an-arduino-uno/>