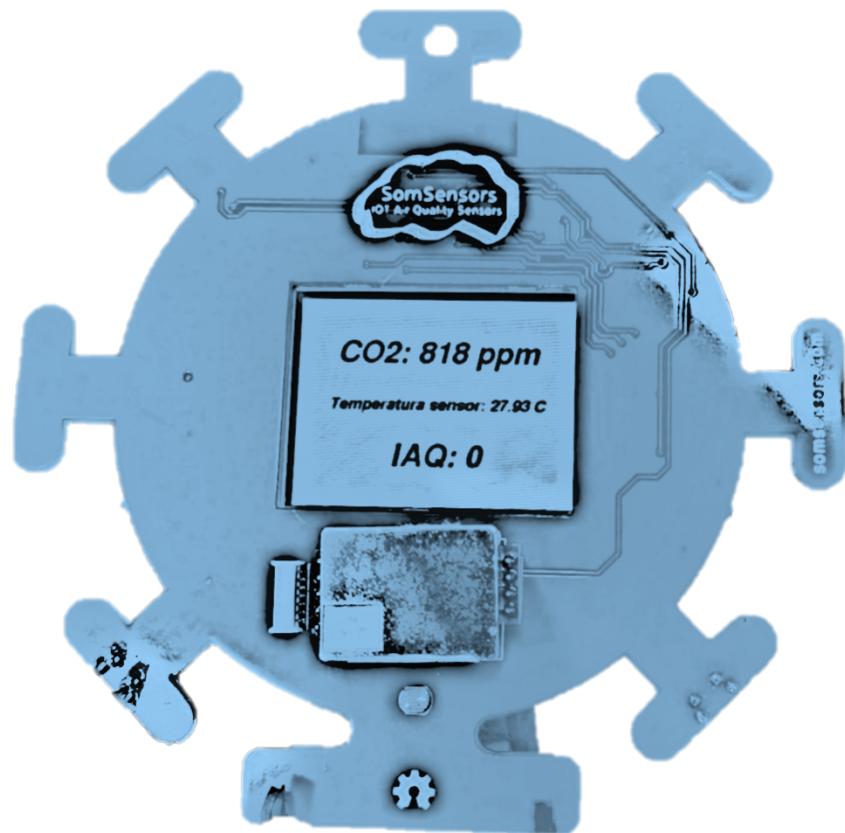


Air Quality Project

PROJECTE FINAL DAM



Alumne: **Albert Lahoz Trigueros**
Tutor: Jordi Binefa i Martínez
DAM – Jesuites el Clot
Data: 26/05/2021

Air Quality Project és un sistema que per millorar el benestar de les persones. Permet controlar la qualitat de l'aire en espais interiors per garantir la correcte ventilació. A més disposa de un registre de dades ambientals per a poder-ne fer un anàlisi i tenir un control de cada espai.

És un sistema programat amb per a ESP32, ESP8266 amb Influx DB, Grafana i node-red. Fa us del protocol MQTT.

Air Quality Project es un sistema ideado para mejorar el bienestar de las personas. Permite controlar la calidad del aire en espacios interiores y cerrados garantizando una correcta ventilación. Además, dispone de un registro de datos ambientales que permite analizar tener un control de los espacios.

Es un sistema desarrollado en ESP32, ESP8266 con Influx DB, Grafana y node-red. Se comunica con el protocolo MQTT.

Air Quality Project is a System developed to improve the healthcare. Air Quality control the environmental quality on a closed space to ensure proper ventilation. Also, Air Quality have a registre of enviornmental data of each space to analyze and control each room.

The System is developed on ESP32, ESP8266 connected with Influx DB, Grafana and node-red. Uses MQTT protocol.

Índex

1. Estudi de viabilitat.....	4
1.1. Objectiu del sistema	4
1.2. Definició dels requisits del sistema.	4
1.3. Estudi de les alternatives de solució.	5
1.4. Valoració de les alternatives.	5
1.5. Selecció de la solució.	6
2. Anàlisi del sistema	7
2.1. Definició del sistema.	7
2.2. Establiment de requisits	7
2.3. Definició d'interfícies d'usuari.	7
2.4. Especificació del pla de proves.	9
3. Disseny del sistema	10
3.1. Arquitectura del sistema.....	10
3.1.1. Definició de nivells d'arquitectura del sistema.	10
3.1.2. Especificació de ports, serveis i protocols.	12
3.1.3. Identificació de subsistemes.	13
3.2. Revisió de casos d'ús.....	14
3.2.1. Revisió dels subsistemes segons els casos d'ús.....	14
3.2.3. Requisits d'implantació.	14
3.3. Persistència de dades	14
3.4. Estructura de la Programació de la Placa	15
3.4. Estructura de la Programació Android	16
4. Desenvolupament.....	17
4.1. Planificació de les activitats del desenvolupament	17
4.4. Rerefons en Node-Red	17
4.4. Documentació tècnica del microprogramari	21
4.5. Documentació tècnica de la VPS	22
5. Implantació.....	24
5.1. Us del sistema.....	24
5.2. Implantació del sistema i proves.	26
6. Manteniment i versions futures.....	27
ANNEX A: Bibliografia.....	29

1. Estudi de viabilitat.

1.1. Objectiu del sistema

Actualment, en el context de pandèmia en el que vivim actualment cal vigilar constantment la qualitat de l'aire i prendre consciència de l'estat dels espais tancats per afavorir una bona ventilació i la renovació de l'aire. Aquesta és l'única manera de garantir una respiració amb aire el més net possible.

Aquest projecte té com a objectiu un sistema que permeti fer un estudi en temps real de la qualitat de l'aire amb l'objectiu que establir una bona ventilació en espais tancats i evitar contagis per via aèria quan es comparteix una sala durant una estona prolongada de temps.

1.2. Definició dels requisits del sistema.

a) Funcions del Hardware

- R001 Lectura sensors ambientals (C02, Temperatura, Lluminositat, Humitat).
- R002 Càcul del IAQ.
- R003 Mostrar C02 per pantalla..
- R004 Mostrar IAQ per pantalla.
- R005 Web Server integrat per reprogramar la placa via Wifi (OTA).
 - i. Protocol per publicar dades via Wifi i guardar-les a una BD.

b) Requisits mínims de la placa (Hardware)

- R001 Chip ESP32 amb Connexió Wifi.
- R002 Pantalla TFT a Color per mostrar la qualitat de l'Aire
- R003 Sensors Ambientals (Temperatura i Humitat)
- R004 Alimentació 3.3V 1A
- R005 3 Polsadors
- R006 Connector de 6 Pins per a carregar dades amb un xip FTDI
- R007 LDR

c) Software

- R001 Control en temps real del Hardware.
- R002 Anàlisi de les dades dels sensors a temps real.
- R003 Base de dades amb el històric.
- R004 Automatització de lectura i escriptura de dades.
- R005 Panell de telecontrol de la placa.
- R006 Panell de anàlisi de dades.
- R007 Ginys visuals per a indicadors de qualitat de l'Aire.
- R008 Gràfiques detallades amb l'hystòric de dades.
- R009 Versió Mòbil per visualitzar les dades des de smartphone.
- R0010 Seguretat en el sistema; administració d'usuaris i contrasenyes per a controlar l'accés i edició del sistema.

1.3. Estudi de les alternatives de solució.

Opció 1: Raspberry + Placa amb ESP32

El S.O que porta la Raspberry contindrà el broker, la BD i un rerefons en Local.

La pròpia Raspberry conté Debian 10 i està sempre encesa i fa de mini-server.

Opció 2: Placa ESP32 + VPN

Els serveis de Dashboard, telecontrol i broker aniran en local a un ordinador virtual al núvol amb SO Debian. Per controlar-ho es farà amb un Terminal i una connexió SSH.

Opció 3: Placa ESP32 + Cloud

Connectar la placa ESP32 amb un broker MQTT i un rerefons basat en serveis cloud en comptes de tenir els serveis en un ordinador local.

Existeixen Grafana, influxDB, Firebase, Moquitto... tots com a serveis Web i per tant elimina la necessitat de serveis funcionant en local.

Opció 4: Placa ESP32 + PC amb Docker

Configuració de el broker i el rerefons en un PC utilitzant contenidors Docker per a publicar els serveis.

Bases de dades: Les bases de dades seleccionades per a la possible solució són Firebase i Influx DB.

Telecontrol: Node-red o Influx DB

Dashboard: Grafana, Node-red

1.4. Valoració de les alternatives.

Opció 1: Raspberry + Placa amb ESP32

És una opció molt òptima ja que es disposa el Hardware i el software de forma física i permet ampliar el sistema vinculant més d'una placa. A més tot el sistema queda en local de forma que es poden estudiar altres protocols per substituir el MQTT. No obstant és un sistema molt simple i amb pocs recursos. A més requereix configuració i manteniment de més hardware.

Opció 2: Placa ESP32 + VPN

És una opció molt versàtil ja que permet tenir els serveis, base de dades i altres programaris de forma local. Fàcil manteniment i gran control sobre el sistema.

Permet aplicar dominis i facilitar l'ús dels serveis associats al VPN.

Opció 3: Placa ESP32 + Cloud

Facilita la instal·lació i manteniment del rerefons del sistema tot i que aporta poc control sobre aquest rerefons. Tot i que no cal instal·lació ni manteniment és més complicada la configuració sobretot entre base de dades i Dashboard.

Node-red no te servei web només en local el telecontrol estaria sobre un servei del PC portàtil i per tant només s'activaria quan aquest estigués encès. El Broker personalitzat MQTT segur també hauria de ser local o fer us d'un broker cedit.

Opció 4: Placa ESP32 + PC amb Docker

És igual de versàtil i manejable que amb una VPN o amb Raspberry. Però el sistema cau quan el portàtil s'apaga no esta actiu sempre ja que no es un servidor.

Bases de dades: La primera opció que es contempla per la BD temporal és Firebase. És un servei Android però també es pot connectar amb ESP32 i permet

fer consultes en temps real. No obstant no es gens versàtil ja que esta molt orientada a Android.

InfluxDB (o en el seu defecte Maria) son bases de dades que estan orientades a BD temporal. Tenim una opció gratuïta per instal·lar en local i cada taula té un nom, una data i hora guardada en un enter i el valor que es vol guardar. Te aplicacions en Panells i Gràfiques temporals i per emmagatzemar i analitzar dades en el temps.

Dashboard: Grafana no permetrà un telecontrol del al placa ja que no es connecta directament a la placa. No obstant aquest servei permetrà millors dashboards i més complexos.

Node-red disposa de panells més simples però te connexió directa amb el broker de manera que és ideal per al telecontrol de la placa.

1.5. Selecció de la solució.

La solució necessitada contará de 3 parts: primer tindrem una placa amb ESP32 i sensors ambientals que proporcionaran informació.

Seguit tindrem una VPN, un ordinador virtual al núvol amb Debian 10 que contindrà tot el rerefons i serveis del sistema.

Finalment disposarem un Node-red i un Grafana que ens permetran el telecontrol i els Dashboard que conformaran la interfície d'usuari del sistema.

Aquesta és una barreja de solucions proposades on combinarem el millor de cadascuna de les propostes per a crear el sistema més òptim.

2. Anàlisi del sistema

2.1. Definició del sistema.

El sistema consta de 3 parts:

Primera una placa amb ESP32 i sensors ambientals que captarà la informació de la sala

Segona part consistirà en un rerefons que llegirà la informació dels sensors, guardarà les dades en una Base de dades influx DB i finalment el rerefons tindrà el contenidor docker amb els panells.

La tercera part consistirà en la interfície d'usuari. Tindrem dues versions de dashboard una més completa amb gràfiques i un històric de temps. La segona interfície consistirà en un Node-Red que permetrà tele controlar la placa. Veure si hi ha connexió amb Wifi i l'estat actual de la sala amb ginys.

2.2. Establiment de requisits

d) Hardware

Placa amb XIP ESP32 i una pantalla TFT 240x320 a color per interactuar.

Sensors ambientals de Temperatura i humitat i un LDR per la llum ambient

Sensor MHZ19 per mesurar el CO2 en l'Ambient.

Led: RGB de control

Polsador: Reset, i dos addicionals per al control de la placa.

Array de 6 pins per a connexió del xip FTDI

e) VPN i Domini

El software, els serveis i el servidor es troben a un VPN, un ordinador virtual al Núvol. Aquest ordinador disposa de: 1 Vnucli, 2Gb de Ram, 40Gb de disc SSD

El sistema operatiu és Debian 10. Per treballar es fa amb una línia de comandes i una connexió SSH.

La VPN està disponible amb una direcció IPv4 però per facilitar l'accés des de fora s'aplicarà un DNS sota el domini iot.airqualityproject.com

f) Rerefons i Software

El rerefons del sistema serà una instància node-red que s'executa en local a la VPN. Aquesta instància requerirà usuari i contrasenya per accedir.

Les dades s'emmagatzemaran a una BD local

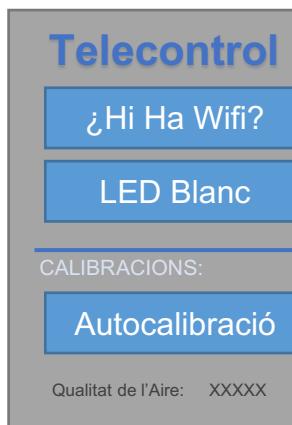
El panell principal de l'aplicació funcionarà amb un Grafana que estarà en un contenidor Docker.

2.3. Definició d'interfícies d'usuari.

Existeixen dues interfícies d'usuari:

La primera és un telecontrol i un panell d'informació a temps real sobre el node-red aquest serà en format vertical adaptat al navegadors i telèfons mòbils.

La segona interfície serà un panell de dades amb Grafana. Aquest contindrà un

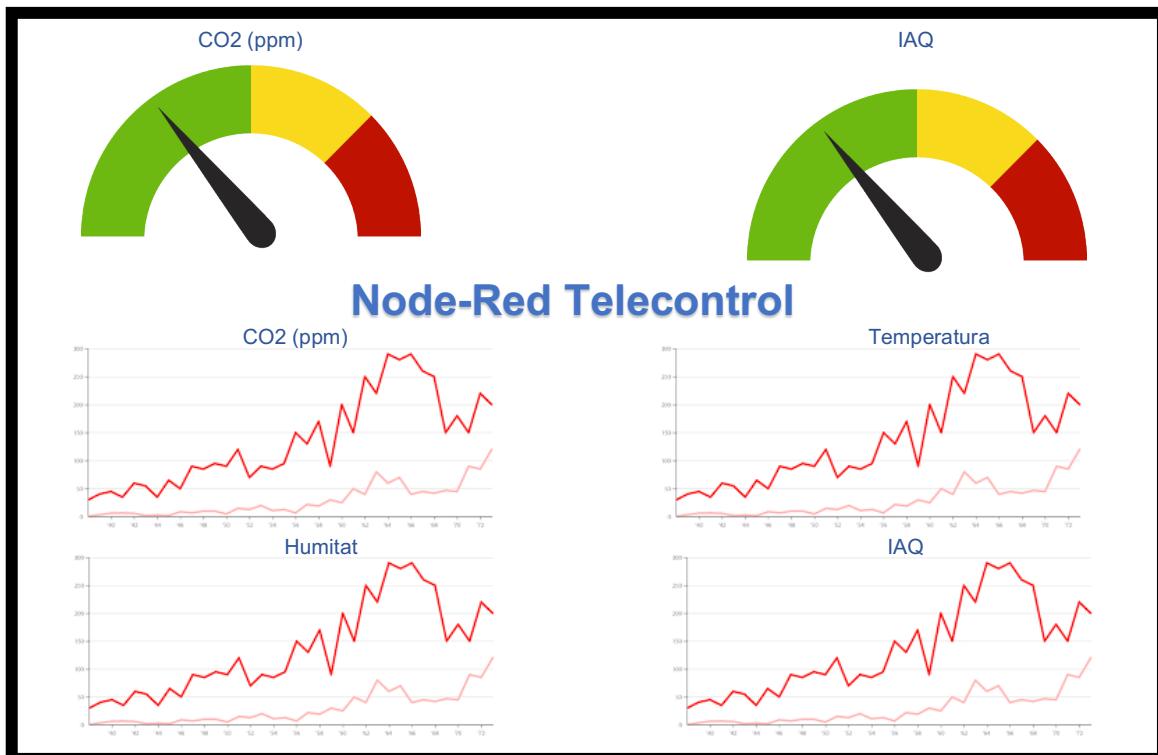


Pantalla 1 Node-red Telecontrol de la



Pantalla 2 Node-red Giny versió
Mòbils

parell de ginys sobre la qualitat de l'aire i després disposarem de gràfiques de les ultimes 6 a 24 hores per veure la evolució i analitzar la qualitat de l'aire en diversos moments del dia. També disposarà d'un enllaç al panell de Node-Red.



2.4. Especificació del pla de proves.

Per verificar el funcionament del sistema es farà mitjançà 5 jocs de proves acumulatives que s'aniran desplegant a mida que es desenvolupi el sistema.

Set. 1 Prova de SomSensors V7

Aquest primer joc de proves consistirà en veure com funciona el codi inicial de la placa. Quines modificacions caldrà fer i si funcionen les noves credencials

Set. 2 Proves de MQTT-Node-Red

El segon joc de proves consistirà en testar el funcionament del Broker MQTT i del rerefons amb node-red.

A més aquí es constitueix la primera versió funcional del sistema. No té persistència de dades però SI és un sistema funcional que permet monitorar en temps real la qualitat d'aire d'una sala.

Set. 3 Proves BD-Node-Red

Aquestes seran les primeres proves amb influx que consistiran en gravar dades de la temperatura a la base de dades (BD) i després fer querys per extreure resultats. Es faran amb node-red i permetran testar la configuració del sistema de presidència de dades.

Set. 4 Producció

És una prova múltiple que consisteix en deixar hores la placa encesa amb l'objectiu d'acumular dades i visualitzar el Panell de Grafana, els panells de node-red i el funcionament de la base de dades.

Permet fer proves en el sistema complet i funcional per poder veure si hi ha errors a reparar.

Set. 5 Programació amb Arxius Binaris OTA

L'últim joc de proves consisteix en afegir petites modificacions al codi de la placa VirKO a través del sistema OTA de programació remota. Sense us del xip FTDI i el port sèrie.

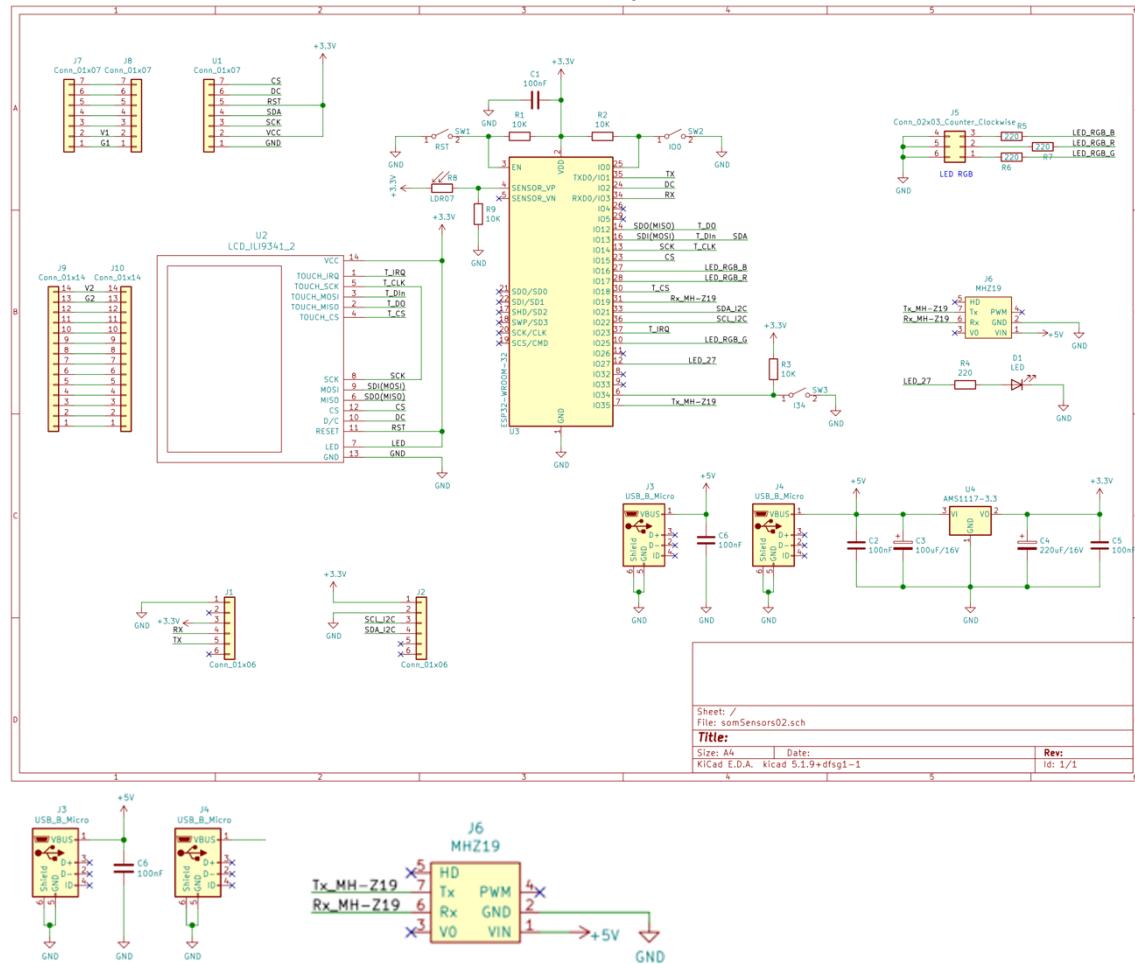
3. Disseny del sistema

3.1. Arquitectura del sistema.

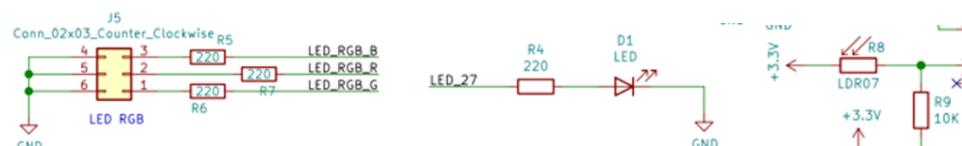
3.1.1. Definició de nivells d'arquitectura del sistema.

NIVELL LÒGIC

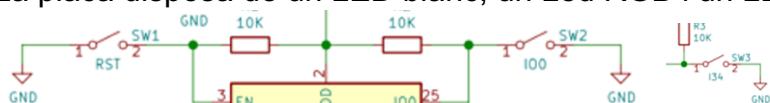
Aquest nivell inclou l'arquitectura de la placa VirKO, és a dir una placa amb ESP32 i els sensors ambientals necessaris per a la recollida d'informació.



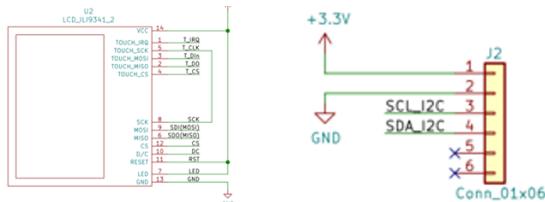
La placa disposa de 2 endolls micro-USB per a l'alimentació per dreta o esquerra. També disposem de un sensor MHZ19 de Gas CO₂.



La placa disposa de un LED blanc, un Led RGB i un LDR (Fotoresistor).



Pel que fan els polsadors tindrem 3; un vermell de Reset un boto negre I0 i un Botó I34.



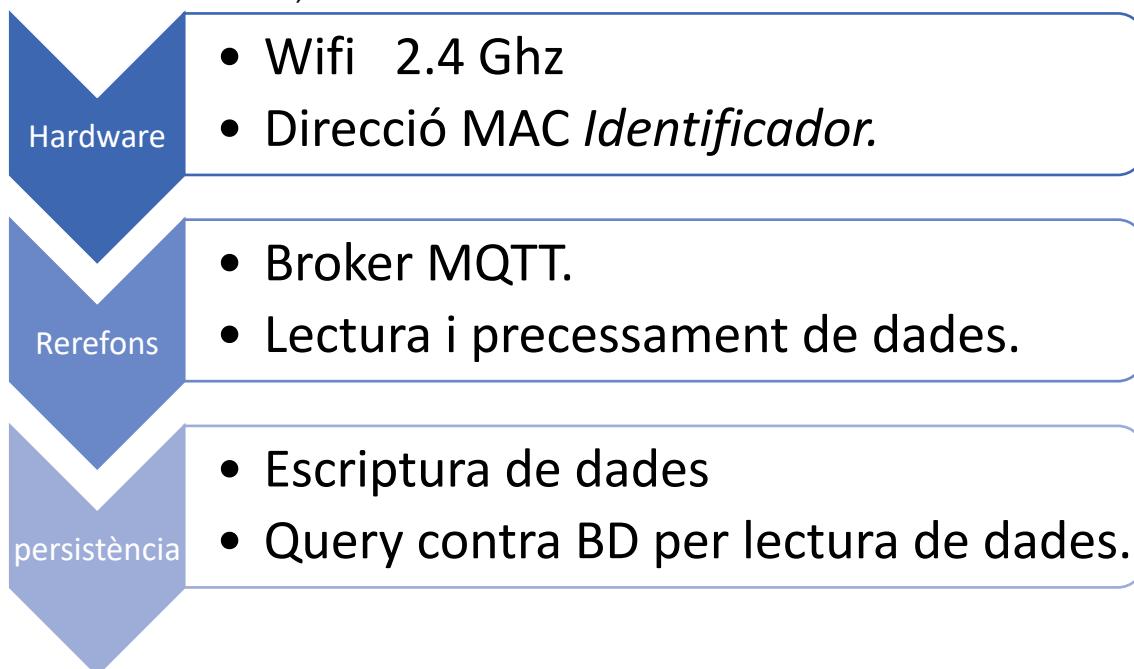
Finalment disposem de una pantalla TFT a color U2.
A J2 disposem de un array de pins on connectarem el sensor de temperatura i humitat.

COMUNICACIONS

Les comunicacions del sistema seran a través de xarxa sense fils. La Placa ESP32 incorpora Wifi 2.4Ghz.

El protocol escollit per a transmetre dades serà MQTT segur. Disposarem un broker propi pe un port TCP. Serà un port segur i disposarà de credencials i un certificat per garantir la protecció de les dades.

El MQTT disposarà de diversos temes per publicar i llegir dades. Tots els temes començaran per la Mac del dispositiu VirKO (ex: /XXXMACXXX/json, /XXXMACXXX/ledW)

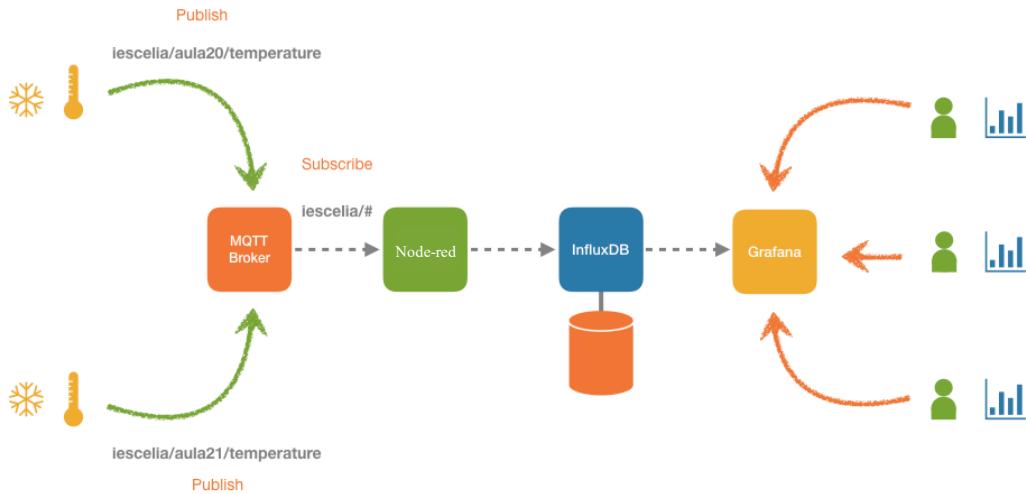


REREFONS

El rerefons estarà instal·lat en una VPS amb Debian 10.

Constarà de una instància Node-Red amb el control i rerefons del sistema.

Base de dades: També estarà a la VPS sobre un servei InfluxDB, cada dada que s'emmagatzemi tindrà una taula i cada taula disposarà del temps i el valor guardats. Per controlar l'influx DB de forma automatitzada utilitzarem la instància de Node-Red.



<https://josejuansanchez.org>

INTERFÍCIE D'USUARI

Disposem de una interfície de usuari dissenyada amb Node-Red dashboard que permetrà visualitzar les dades en temps real i estarà optimitzat per al telèfon mòbil. A més aquest panell contindrà el telecontrol per comprovar la connexió wifi de la placa i fer l'Autocalibració del Sensor de CO₂. El panell principal serà amb Grafana. La instància estarà a un contenidor Docker i tindrà les gràfiques temporals de les mesures que tinguem guardades a la Base de dades.

3.1.2. Especificació de ports, serveis i protocols.

El domini que durà el rerefons és airqualityproject.com però disposarem de diferents subdominis per als serveis:

- Node-Red → iot.airqualityproject.com:2222 (port 2222)
Autenticació amb usuari i contrasenya només per accedir al disseny
- InfluxDB → iot.airqualityproject.com:8086 (port 8086)
Cal autenticar-se amb Versió 1.X i usuari admin
- Grafana → iot.airqualityproject.com:3000 (port 3000)
*Usuari administrador: admin
Usuari editor: usuari1
Visualització: convidat*
- Panell Node-Red → iot.airqualityproject.com:2222 (port 2222)
- OVH → ovhcloud.com
- VPS → iot.airqualityproject.com (connexió SSH usuari debian)
- Broker MQTT → iot.airqualityproject.com (port 8883)

Per el que fa el MQTTS caldrà un certificat generat amb moquitto. El certificat caldrà a la placa. A Node-red només caldran credencials per al MQTT utilitzarem l'usuari *airquality*.

```

const char* mqtt_server = "iot.airqualityproject.com"; // Your MQTT broker
const int mqtt_port = 8883;
const char* mqtt_user = "airquality";
const char* mqtt_password = "Jleba1375";
const char* ca_cert = \
    "-----BEGIN CERTIFICATE-----\n" \
    "MIIFtTCCASgAwIBAgIUQlMqFgb7LRGksT/PeSA3MZolNcwDQYJKoZIhvcNAQEN\n" \
    "BQAwajEXMBUGA1UEAwwOQW4gTVFUVCBicm9rZXIxFjAUBgNVBAoMDU93b1RyYWNr\n" \
    "cy5vcmcxFDASBgNVBasMC2d1bmVyxXR1LUNBMSewlwYKoZIhvcNAQkBFhJubZJv\n" \
    "ZHI1AZXhhbXbS2SSuZXQwHhcNNjEwNTA2MTQyNzE5WhcNMZIwNTAzMTQyNzESWjbq\n" \
    "MRcwfQYDVQQDDAsBbiBNURUIGJyb2tlcjEWMBQGA1UECgwNT3duVHJhY2tzLm9y\n" \
    "ZzEUMB1GA1UECwLZ2VuZXJhdGUtQ0ExITAFBgkqhkiGw0BCQEWEm5vym9keUB1\n" \
    "eGftcGx1Lm51dDCCAiIwDQYJKoZIhvcNAQEEBBQADggIPADCAgoCggIBAMfjgTz\n" \
    "2bdXiMSZHxeHtg230fUnFmYsxhruPCqpSS803Ixw19w3yFeVSeEM4ShAGhaCV2G\n" \
    "WADu0U602poziKL1kvjtP1UFze16tjVmTwuSeuysdbaEb80u+2581NjK2DBf38\n" \
    "6tzmSC/eaTbmCkAsw06B54gk8Bm84RDnHU7iDyq5yDnd+t8Qu/fUJ/y2PM0Vg+WI\n" \
    "bXMRmmrsijCo2zsTbErmyxEjxYbsNS9TR7ePSUNA6Mb7FnjPKvJEmn2cYk7oAx\n" \
    "NuFg7rH8jKYfqxRm9yFQpSLAn9dTEnL8s7mHQjh/Ops/1CDC+8D9FD1#720n\n" \
    "kE9itgQfqmTbGxnHndkzghMy51cXqTROZvp6CLs+GZyrGNch7kgHixSFwmPs\n" \
    "72N8zPoGXG5GqrPx5M//a10XXIoqEK6jMrVqxefjhkB6b1VTNGzbH8b41QcyG\n" \
    "vAlFFEKEviyRF4uH8p4F&zMYZQ0gxeYdewbvouNyinTp4deT134/3eJ/FEECkD\n" \
    "xsDZwmBoHjdoGShb/87/Vtb21kygSLISMXkzV5j11HgxabZMNE2R/FT9Ap3dLw\n" \
    "12NMUK5kwCJP/FwyICtFneeMeX++IhgI3anWEfpJwaiLSYBDsaTSSTWNzkc\n" \
    "LPmB4X8yf081+IYr3zd1vYNFwmtb1A1zdeJ1AgMBAAGjUzBRM80GA1udgQWB\n" \
    "tOBhytxkeoSvPHH3BTV7ipC3+DAFBgNVHSMEGDAwBR5tOBhytxkeoSvPHH3BT\n" \
    "ipC3+DAP8gNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBDQAA4ICAQATdv4sqJA\n" \
    "d0KWeI4vBybz0bTFOcq1fUzfxu/KsRkkd+xjcnxRNBNNXFi3WpNEAIm6sLN90\n" \
    "98Donnrw5Qyl4ANBGN61oLWZpjX-i96idBwgkGMgwk4iots56orp3gvAK3RN\n" \
    "/22h1czK0pD2QvgVG9CsZ/8Pdsww9TIVImeMrHULLkhNA2T8NmT30MzY5L0\n" \
    "4taZTvdBDmCa5jG1btppdkCkq1Bpw8im04Bn2rc19UuV1+wEcorYHiLRBa2e\n" \
    "upj10QnAxnqmm348Vnc0gGjurAk0NgYFwqkwcHSh7pousBYLz1SDgH1173\n" \
    "HHGfODXQI6pdqwZDBeemvd1U4dU64pRj+1iPSHYn+dcRxkPCqz+Vr50Dit\n" \
    "XvaxVn" \
    "X7/ChcrWmjjeAxGULcRNMKvuSTonBlqmIsa59DnlfcGmrVvlFd3FI6/n8t7\n" \
    "c4g4spPTKcw/mktrC2wrReBF72Gj5T8CiTu9e9478LFIOxpioCQLUAz6uwB\n" \
    "qwDRq4z8EDUuAj1.jcc61ABCuyQVft2NXU/YWBsd9+CwNbrEQp7j132imL93\n" \
    "TWUJHvSeGjpElWY5ls/w/CmhUntc0cQ3zhdHpkNk7fJHqjryDkxZm9ViwiK\n" \
    "QuPpAbJ2aeEnZWx/+DU2sp4nW1jb1pKrg==\n" \
    "-----END CERTIFICATE-----\n";

```

3.1.3. Identificació de subsistemes.

- SUB01 Pantalla TFT color
(Interacció directe amb la placa)
- SUB02 Android WebApp
(panell node-red)
- SUB03 Navegador Web
(panell node-red)
- SUB04 Navegador Web
(panell Grafana)

3.2. Revisió de casos d'ús.

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

Podem definir tres subsistemes segon les interaccions que fa l'usuari i el sistema final.

CU01 Interacció directe amb la placa

En aquest cas l'usuari controlarà l'estat de la sala a través de la informació que troba en la pantalla TFT

CU02 Interacció des de dispositius mòbils

L'usuari es connecta a través d'una webApp que obre el node-red i o el Grafana. Aquest cas inclou telecontrol o panell de dades.

CU03 Telecontrol

L'usuari utilitza el panell node-red per interactuar i fer el telecontrol però encomptes d'utilitzar la webApp accedim des de un navegador amb l'ordinador.

CU04 Navegador Web

L'usuari accedeix al panell principal de visualització de dades i interactua amb Grafana des de un navegador web (preferiblement a un ordinador). Per aquest panell cal contrasenya.

3.2.3. Requisits d'implantació.

Per el correcte funcionament del sistema cal disposar de connexió Wifi amb les credencials carregades al microprogramari de la placa.

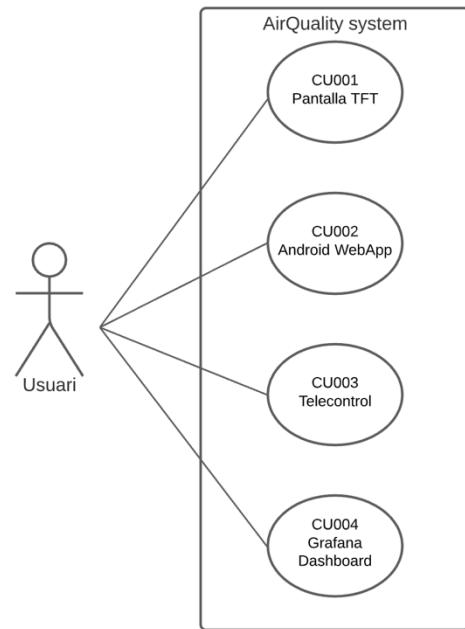
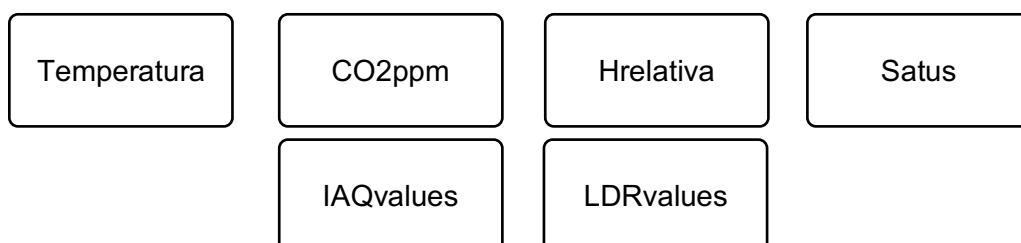
El rerefons esta instal·lat en una VPS per tant cal un terminal Mac, Linux o en cas que el SO sigui Windows un programa per connexions ssh (Putty)

Per a la visualització de les dades i la interacció amb la placa cal un Telèfon amb S.O Android o un ordinador amb navegador web.

3.3. Persistència de dades

Les dades entren a Node-Red en forma de Json Object. El rerefons el processa i es guarden les dades en a una base de dades Influx.

De tota la cadena Json les dades que es guarden són: Temperatura, CO2, IAQ, LDR, Status, Humitat Relativa.



Influx és una base de dades temporal. Les Variables es guarden en taules que no estan relacionades entre si i cada taula conte dues columnes: Time & Value

3.4. Estructura de la Programació de la Placa

La placa es programa mitjançant el IDE Arduino per tant l'estructura bàsica dels programes és pròpia de Arduino.

El codi està basat en:

```
void setup() {  
    // put your setup code here, to run once:  
  
}
```

El setup és el primer que s'executa en el microprogramari i permet inicialitzar pins, funcionalitats i procediments que s'utilitzaran durant l'execució del codi.

```
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

A continuació tenim el Void loop que és el procediment principal i més bàsic del codi. Es diu loop ja que es un bucle infinit que s'executa constantment al microprocessador.

Disposarem de arxius .h que permetran importar configuracions, constants i sobre tot credencials. El fitxer principal .ino importarà tota la informació.

Mètodes Sensors:

Per inicialitzar el MHZ19 i el BME 680 cal un mètode `vSetupMZ19` `vSetupBME680` que ens permetin detectar i inicialitzar les dades per a la posterior lectura.

IAQ (índex de qualitat de l'Aire) requerirà un mètode que el calculi amb el GasScore i la humitat. `CalculateAQ`

Mètodes TFT i LEDS:

La pantalla necessita un mètode que la inicialitzi; `SetupDisplay`, a més de un `vPResentaPantalla` que ens permeti mostrar nova informació i mantenir-la actualitzada.

Per mostrar la informació de connexió utilitzarem un mètode `MostraMACip` que ens mostrarà la Mac i la Wifi per pantalla quan el boto i34 estigui premut.

També necessitem un mètode `vHihaWifi` que mostri el text "Hi ha Wifi!" que va subscrit a un tòpic MQTT i forma part del sistema de telecontrol de la placa.

L'últim mètode d'aquesta secció és el `vLEDblanc` que igual que el mètode anterior forma part del telecontrol de la placa. Aquest encén el LED blanc segons un tòpic MQTT que s'activa publicant des de el Node-Red.

Mètodes OTA:

Per al sistema OTA el que farem és un Web server. Necessitem un mètode `vOTAServerSetup()` que ens permeti arrencar aquest petit Web server amb el que podrem reprogramar la placa.

Mètodes MQTTS i xarxa:

El protocol MQTT necessita subscriure's per publicar i llegit informació. És per això que cal un mètode *MQTTconnect* i *SetupMQTT* per a connectar al broker i al servidor i verificar la connexió segura amb el certificat.

També disposarem un mètode *ReconnectMQTT* i *ReconnectWIFI* amb el que cercar noves connexions wifi en cas de pèrdua de la xara.

Per la Xarxa necessitem un mètode que ens permeti conèixer la IP i la MAC de la placa.

Void Setup:

- Inicialitzar variables globals
- Inicialització dels sensors i els pins
- Declarar els fils (multitasca)

Void Loop:

1. Llegir de Sensors
2. Publicar informació
3. Refrescar Pantalla TFT
4. Controlar Botons Premuts
5. Controlar si cal encendre el LED blanc

Es repeteixen aquestes 5 funcions de forma constant.

Other TASKS:

El programa inclou 3 tasques, la principal i dues addicionals:

Tasca MQTT → s'encarrega de connectar amb el broker i mantenir oberta aquesta via de comunicació amb els tòpics subscrits perquè el loop pugui publicar o rebre informacions.

Tasca LED RGB → s'encarrega de controlar el LED RGB de la placa i posar-lo vermell, verd o blau n funció de les mesures i de la qualitat de l'Aire.

Credencials necessàries:

Per funcionar en tot moment s'ha de tenir en compte que cal establir els tòpics on subscriure's, i calen credencials per MQTTS i per les Xarxes Wifi.

3.4. Estructura de la Programació Android

La versió per visualitzar dades en dispositius mòbils és molt simple. És bàsicament un WebView a una empty Activity.

Main Activity → en el cos del programa, en java, obrirem un WebView i li indicarem que mostri la URL de Node-red (dashboard)

Main Fragment → La interfície principal serà una pantalla en blanc amb un WebWiew responsive que es reconfigura segons la mida de la pantalla del dispositiu.

4. Desenvolupament.

4.1. Planificació de les activitats del desenvolupament

El desenvolupament d'aquest projecte durarà 4 setmanes i l'eix de les planificacions serà setmanal.

Setmana 1 (dl a dc.)

Definició dels projectes. Abast del sistema. Objectius del projecte.

Setmana 1 (dc. A dv.)

Presa de contacte amb VirKO i comprensió del codi SomSensors V0.7

Set de proves 1 – SomSensors 7

Setmana 2 (dl a dc.)

Configuració de la VPN. Instal·lació de Node-red. Instal·lació de MQTT

Proves amb Codi de Node-Red i amb el broker local MQTT

Set de proves 2 – SomSensors 7 amb broker local MQTT

Setmana 2 (dc. A dv.)

Desenvolupament de la primera versió de dashboard amb Node-Red. (Només lectura)

Desenvolupament del Telecontrol de Node-Red.

Set de proves 2 – SomSensors 7 + NodeRed + MQTTS

Setmana 3 (dl a dc.)

Instal·lació i configuració de Base de Dades, Docker i Grafana. Proves amb la Base de dades, lectura, escriptura i esborrat.

Començar aplicació Android per a fer un WebView i connectar amb Node-red i Grafana.

Set de proves 3 – Node-red i InfluxDB

Setmana 3 (dc. A dv.)

Connexió definitiva amb la BD i creació dels primers Ginys amb Grafana.

Set de proves 4 – Producció

Finalització de l'aplicació WebView d'Android.

Setmana 4 (dl a dv.)

Proves del sistema complet i millores en Dashboard.

Incorporació de OTA en el sistema (SomSensor V0.8) per reprogramar la placa amb Wifi, sense fils.

Set de proves 4 – Producció

Set de proves 5 – SomSensors 7

Setmana 5 (dl a dc.)

Detecció d'errors i finalització de la documentació associada.

Set de proves 4 – Producció

4.4. Rerefons en Node-Red

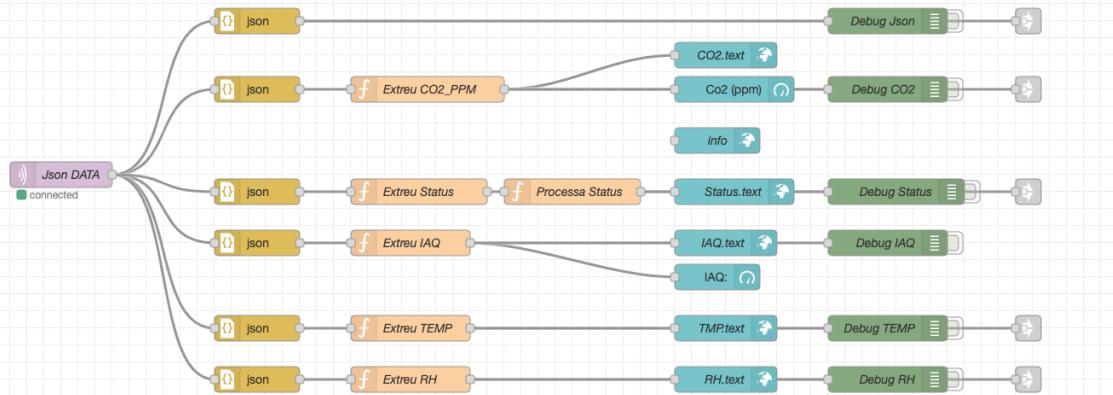
El rerefons que connecta les interfícies amb la placa esta fet amb node-red, la instància que ho controla funciona per el port 2222 i requereix de un login si vols veure o modificar el rerefons.

 No es seguro | iot.airqualityproject.com:2222/#flow/57c90885.4164e8

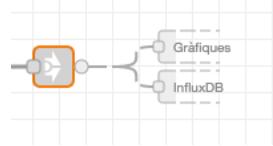


Disposem de 4 FLOWS

1. Dashboard (ginys i informació en temps real de la sala)



Aquest és el node principal. Rep informació amb el tòpic Json DATA, processa aquesta informació i la trenca en trossets on cada tros representa una dada diferent de un sensor de la placa.



A la dreta veieu una fletxa → aquest node ens permet rebotar la informació processada del json a altres pestanyes sense haver de tenir un altre listener MQTT i tot el que comporta.

```

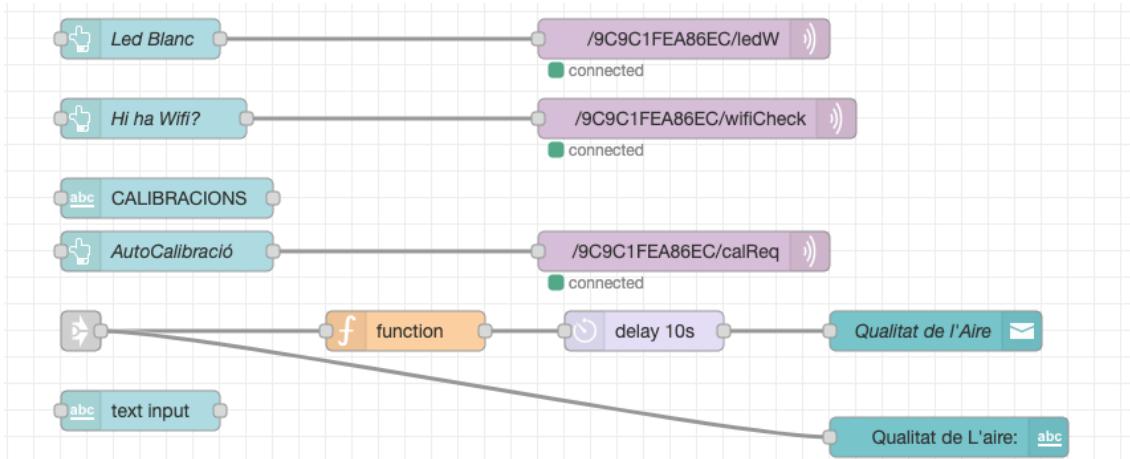
1 var nIAQ = msg.payload.IAQ
2 msg.payload = nIAQ
3 return msg;

```

Dins de cada Funció *Extreu* trobem un codi que donat un objecte (array) Json extreu la dada que en aquest cas ens interessa.

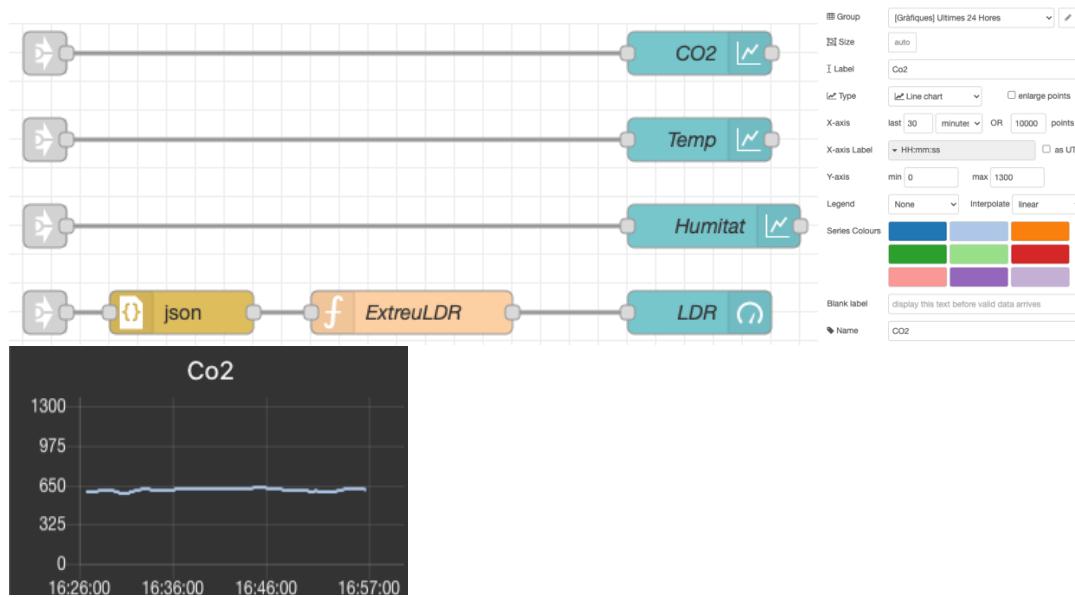
A més a més els ginys que trobem a la captura inicial formen la pestanya principal del dashboard, mostren l'estat en temps real de la placa.

2. Telecontrol de la placa



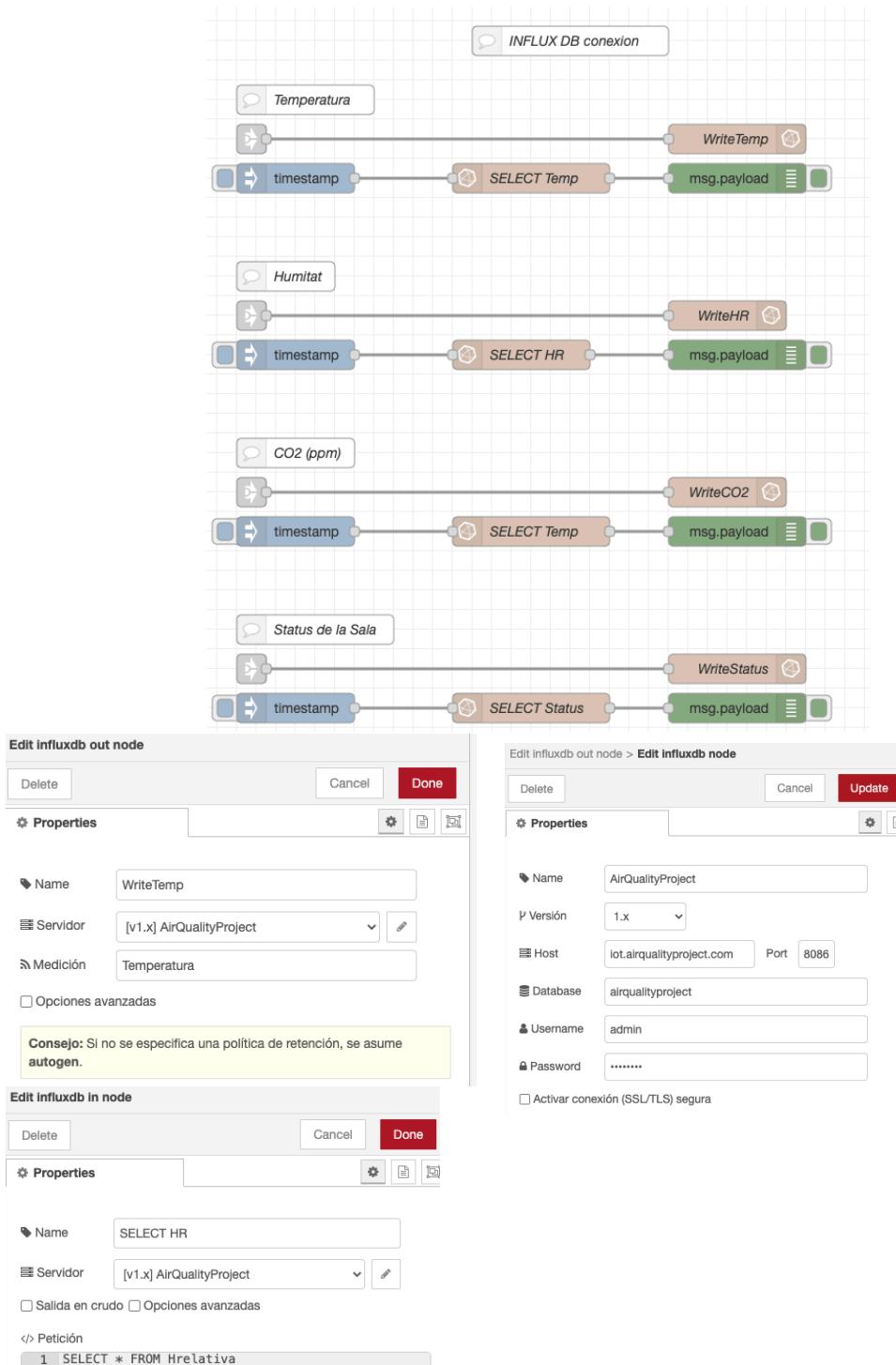
Aquesta pestanya no rep informació sinó que la publica per poder interactuar amb la placa. A més disposa de ginys en el panell de telecontrol.
 La part inferior de la imatge mostra els nodes de notificacions. Quan cal ventilar la sala aquesta funció emet una notificació al dashboard que apareix cada 10 segons durant 5 segons. I indica que cal ventilar.

3. Gràfiques (Gràfiques valor-temp de la V.1 del sistema)
 En la primera versió funcional del sistema les gràfiques temporals amb la cronologia venien fetes amb node-red.

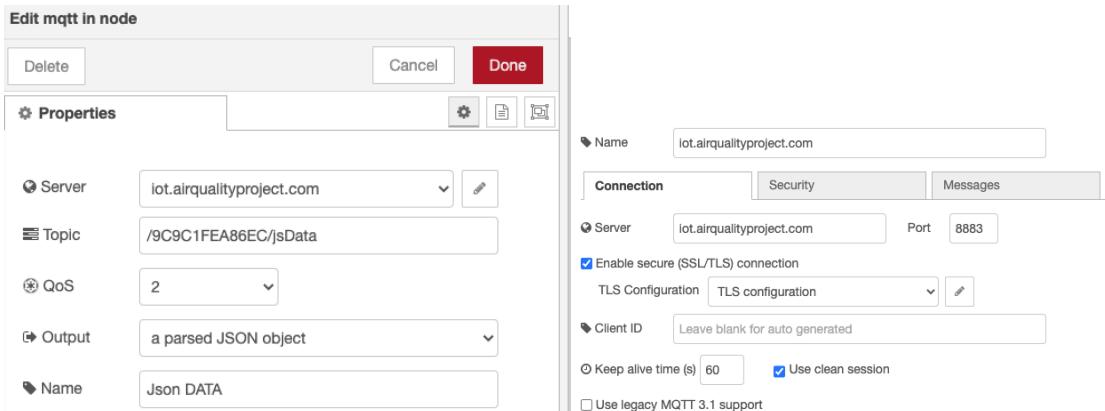


En la versió final aquest panell queda deshabilitat ja que les gràfiques son de baixa qualitat i només acumulen 1h de dades sense opció de recuperació.

4. InfluxDB
 L'última pestanya és la de influx. Aquesta rep les dades processades del node principal i les grava a la base de dades de InfluxDB **airqualityproject** a més disposa de un petit tros de prova que permet fer un SELECT de les diferents taules de cara a proves de la BD amb el debug.



Finalment mostro unes captures e la configuració del broker MQTTS a Node-Red. El tòpic subcrit és la direcció MAC de la placa per identificar-la. I /tòpic que especifica exactament quina informació passa.



4.4. Documentació tècnica del microprogramari

L'estructura bàsica del programa està definida al [apartat 3.4](#)

SomSensors_S8_Lahoz

És el nom que rep el fitxer principal i el programa pròpiament. El codi està basat en el microprogramari SomSensors_S8_08 d'en Jordi Binefa.

Aquest arxiu conté les funcions i el desenvolupament del programa.

Abstract de funcions que s'utilitzen en el codi del microprogramari:

- **Void setupDisplay()** inicialitza la pantalla TFT en cada Reset.
- **vPresentaPantalla (nCO2, lft, nIAQ)** escriu a la pantalla la informació del sensor de CO2, temperatura i IAQ a més d'establir el color de fons segons la qualitat de l'aire.
- **vPresentaMacIP(sMAC, sIP)** Escriu a la pantalla la MAC i la IP de la placa. Servei per quan mantenim premut el botó I34
- **vPantallaHiHaWifi()** Mostra per pantalla Hi Ha Wifi! Ens servirà per el tòpic HiHaWifi del Telecontrol de la placa.
- **GetGasScore()** Calcula la puntuació segons la lectura del CO2 per poder calcular la IAQ.
- **CalculateIAQ()** Aquesta funció agafa el GasScore i la humitat per calcular com un percentatge la qualitat de l'aire.
- **GetHumidityScore()** Calcula la contribució de la humitat al IAQ
- **bSetuBME680()** Cerca un sensor BME680 als pins indicats i l'inicialitza
- **bPressedButton()** controla i el I34 esta premut per mostrar MAC i IP

Tenim tres tasques que s'executen simultàniament: principal, MQTT, ledRGB

La tasca MQTT consta de 6 mètodes:

- **ip2STR()** que converteix la IP en un String
- **ReceivedCallback()** defineix un listener que escolta Tòpics MQTT que comencin amb la MAC de la placa i permet interactuar entre el telecontrol i la placa.
- **mqttConnect()** obre la connexió amb MQTT estableix el listener i subscriu a tòpics per poder publicar-hi.
- **vSetupMQTT()** obre la connexió per port segur amb el corresponent certificat.
- **vReconnectWifiMQTT()** aquest mètode connecta amb una xarxa wifi i agafa la direcció mac de la placa per als tòpics MQTT.
- **vSetupOtaWebServer()** aquest mètode conte el funcionament de el sistema OTA que a través d'un portal HTML i la direcció IP de la Placa podrem pujar binaris per programar-la sense fils.

vConnectingMqttsTask() és el nom del mètode principal que executa aquesta tasca.

La tasca LED RGB és una mica més simple **TaskSw3LedW** permet encendre un led en color Red, Green o Blue Segons l'estat de la qualitat de l'aire a joc amb el color del fons de la pantalla TFT per complementar el sistema.

Finalment la tasca principal, és la típica de tot programa Arduino; Setup que estableix la configuració i inicialitza les tasques addicionals i un loop que es reproduceix de manera infinita i conte el codi principal del programa. Que llegeix dels sensors i processa la informació per publicar-la i mostrar-la a la pantalla TFT.

El codi del microprogramari consta d'un fitxer .ino però també de uns fitxes .h que fan de llibreria i defineixen credencials i altres informacions vitals per al codi.

- Free_Fonts

Necessari per definir amb constants els paràmetres del text que es veurà a la pantalla TFT.

- MQTTcredentials

Conté les credencials i el certificat MQTT

- WifiCredentials

Conté les credencials i noms de SSID per la connexió Wifi

- WifiMng (.cpp i .h)

Contenen un seguit de funcions addicionals per a l'ús de Wifi amb el xip ESP32, ja que cal fer resets, obtenir la direcció Mac, buscar SSID disponibles... i tot plegat requereix funcions addicionals.

El codi del microprogramari també fa us d'algunes llibreries que cal descarregar i tenir a la carpeta Arduino per tal de poder compilar:

- | | |
|-------------------|----------------|
| ○ Fee_Fonts | ○ Wifi |
| ○ TFT_eSPI | ○ WofoClient |
| ○ MHZ19 | ○ WebServer |
| ○ SoftwareSerial | ○ Update |
| ○ Adafruit_sensor | ○ ESPmDNS |
| ○ Adafruit_BME680 | ○ PubSubClient |

4.5. Documentació tècnica de la VPS

Per aquest projecte s'ha disposat de un Virtual Private Server, un ordinador virtual al núvol, que esta amb la empresa OVH cloud. Les característiques són:

- 1 vCore
- 2Gb de RAM dedicada
- 40Gb de disc SSD
- OS Debian 10 pre-instal·lat

Aquest VPS se li ha aplicat el domini iot.airqualityproject.com per facilitar el seu accés.

```
albertlahoz@MacBook-Pro-de-Albert ~ % ssh -o "ServerAliveInterval 30" debian@iot.airqualityproject.com
```

Per a tenir una instància node-red activa sempre s'ha instal·lat un node-red local amb la comanda Screen. Això permet deixar la instància executant-se.

```
debian@vps-13b8fffc1:~$ screen -ls
There are screens on:
  22518.pts-0.vps-13b8fffc1          (05/26/2021 04:07:36 PM)          (Attached)
  15458.pts-1.vps-13b8fffc1          (05/07/2021 02:42:17 PM)          (Detached)
2 Sockets in /run/screen/S-debian.
```

El dashboard principal esta fet amb Grafana. En aquest cas el grafana s'ha configurat com a contenidor docker al port 3000

```
debian@vps-13b8fffc1:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
ea552aa69f41        grafana/grafana   "/run.sh"          9 days ago         Up 9 days          0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   grafana
debian@vps-13b8fffc1:~$
```

El broker mqtt Moquitto en local treballa per port segur i amb credencials.

```
debian@vps-13b8fffc1:~$ cd /etc/mosquitto
debian@vps-13b8fffc1:/etc/mosquitto$ ls
ca_certificates  certs  conf.d  mosquitto.conf
debian@vps-13b8fffc1:/etc/mosquitto$
```

L'últim que tenim a la VPS és el la persistència amb una base de dades Influx. Aquest no funciona amb contenidor docker sinó que s'ha instal·lat com a servei.

```
debian@vps-13b8fffc1:/etc/mosquitto$ influx
Connected to http://localhost:8086 version 1.8.5
InfluxDB shell version: 1.8.5
> use airqualityproject
Using database airqualityproject
UNIQUE influx.service could not be found.
debian@vps-13b8fffc1:/etc/mosquitto$ sudo systemctl status influxd
● influxdb.service - InfluxDB is an open-source, distributed, time series database
  Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2021-05-17 14:12:15 UTC; 1 weeks 2 days ago
    Docs: https://docs.influxdata.com/influxdb/
  Main PID: 12439 (influxd)
     Tasks: 10 (limit: 2319)
    Memory: 205.9M
      CGroup: /system.slice/influxdb.service
              └─12439 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

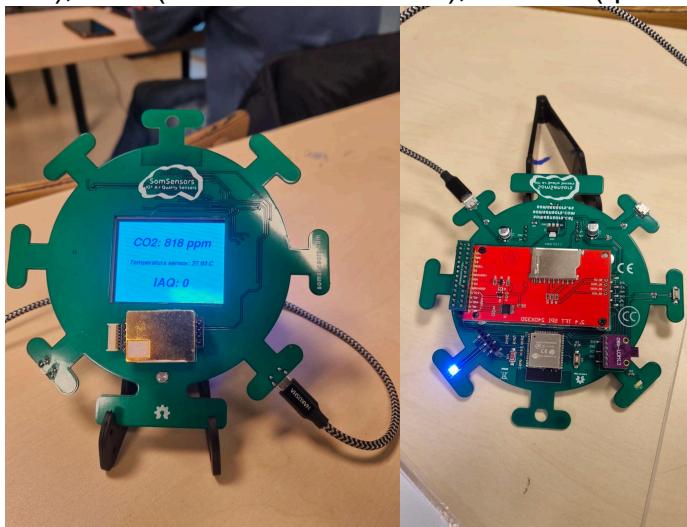
5. Implantació.

5.1. Us del sistema.

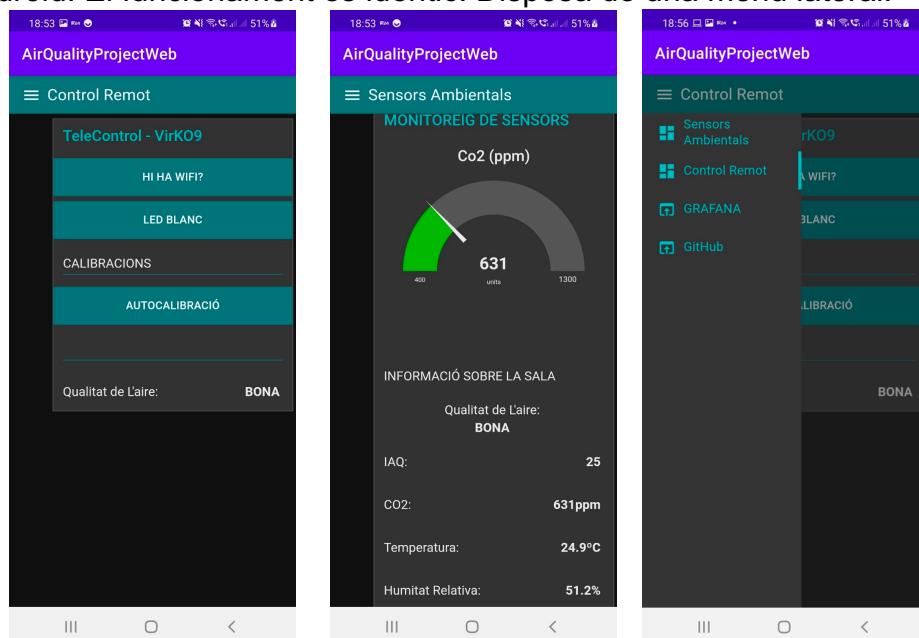
El sistema té 3 Tipus d'us:

La placa VirKO permet interactuar de manera directe. La pantalla TFT permet veure el nivell de CO₂, el IAQ. A més el fons de la pantalla (i el LED RGB que te darrere) canvien de color segons la qualitat de l'Aire:

Verd (qualitat bona), Blau (Recomanat Ventilar), Vermell (qualitat dolenta)

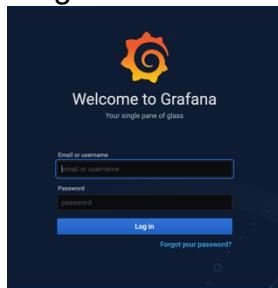


També poder fer us de Node-Red amb navegador web o l'App AirQualityProject per Android. El funcionament és idèntic. Disposa de una menú lateral:



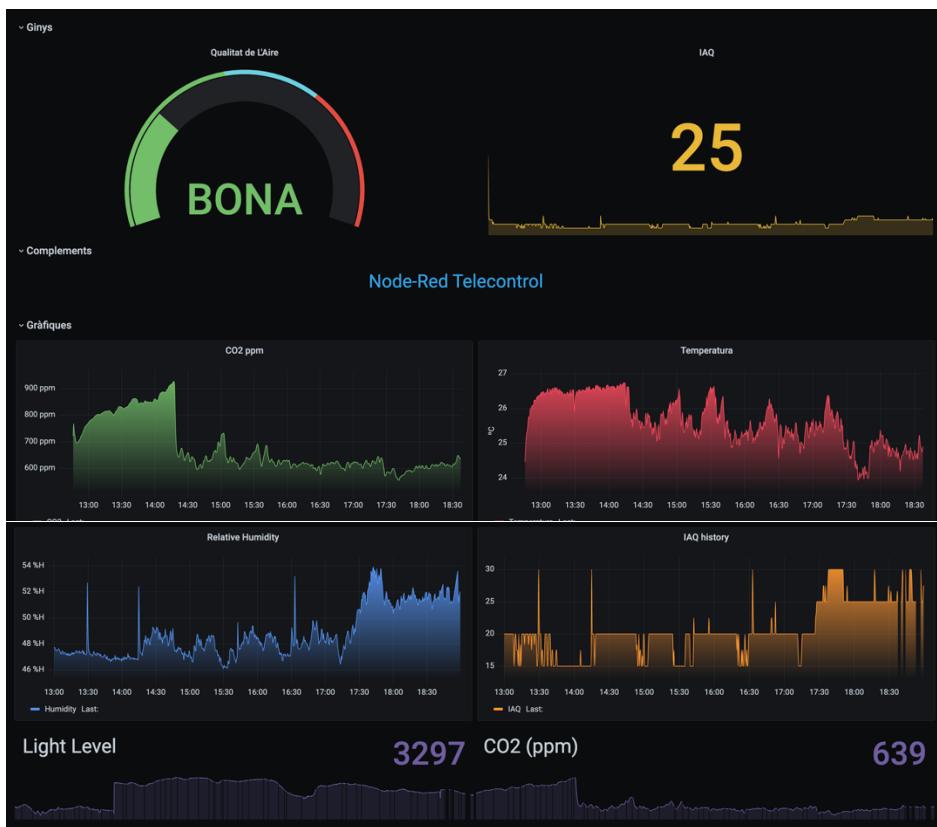
NODE-RED

Per a l'anàlisi de dades i visualitzar Gràfiques temporals podem accedir al panell Grafana. Esta optimitzat per navegadors web d'ordinadors. [GRAFANA](#)



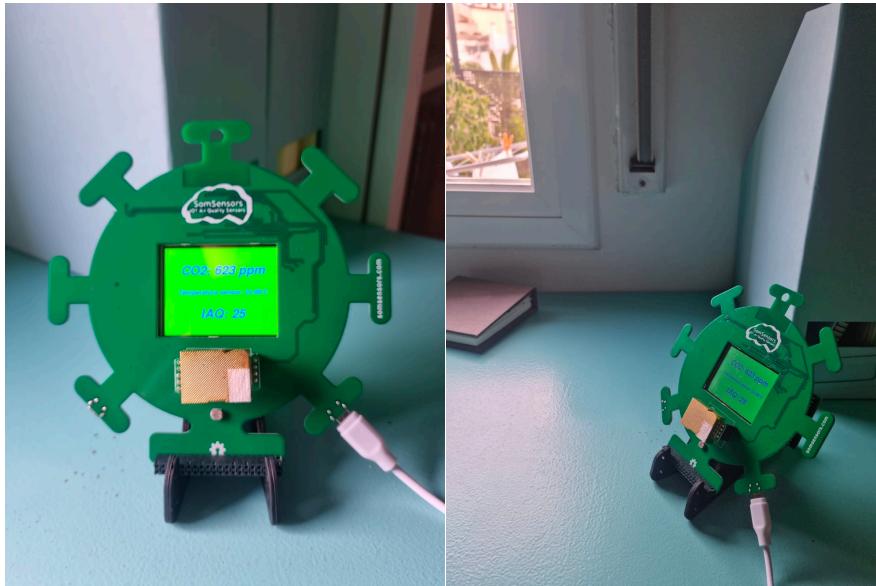
Pots accedir amb: convidat/convidat

I Veuràs directament el panell principal de AirQuality

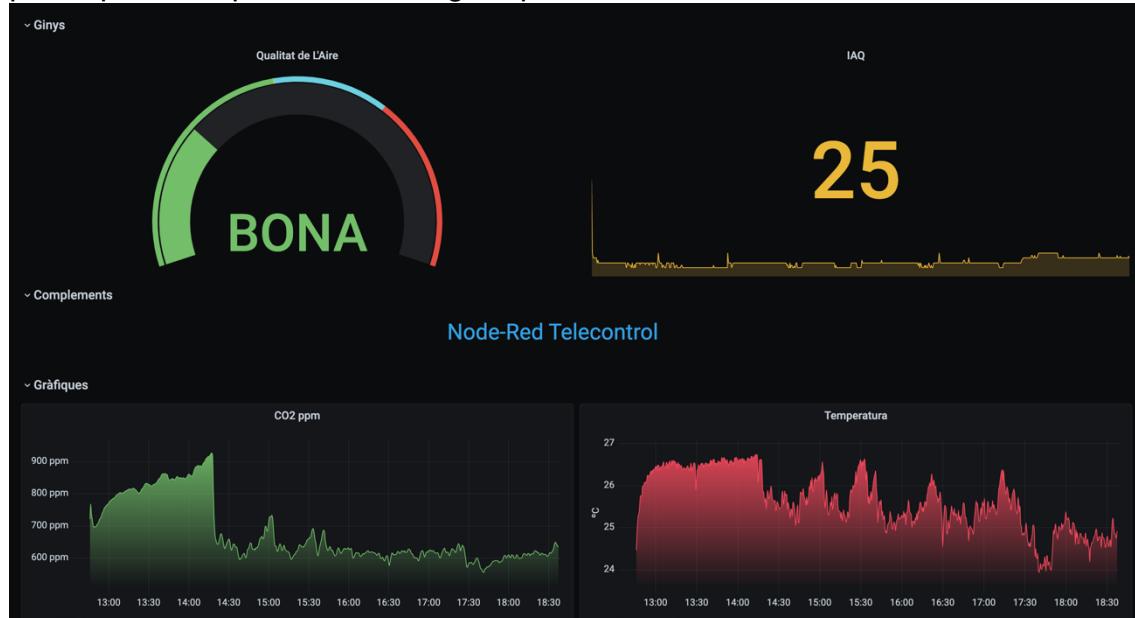


5.2. Implantació del sistema i proves.

La placa ha estat instal·lada a una habitació interior. Aquesta està prop de una finestra i per tant es pot obrir i tancar quan convé per generar lectures amb aire renovat o reciclat.



Un cop instal·lada la placa es pot procedir a fer una prova de producció on la deixem encesa tot el dia captant dades i transmetent-les a la Base de dades per poder plenar el panell amb les gràfiques.



La resta de [jocs de proves](#) es poden fer movent la placa de lloc ja que no tenen una connexió constant amb la Base de dades.

6. Manteniment i versions futures.

El primer que cal comentar sobre versions futures es que el sistema dissenyat és molt escalable.

Des de un inici l'objectiu era acumular moltes plaques i poder disposar de monitoratge de diverses sales en un mateix sistema. Per disponibilitat de Hardware no ha sigut possible però el sistema ha sigut dissenyat per poder escalar-lo i incorporar fàcilment múltiples plaques.

En versions futures es pot combinar diverses plaques simplement identificant els tòpics MQTT i les dades de la base influx amb la direcció MAC de cada placa.

Per el que fa el manteniment cal un sistema que esborri les dades de la base de dades cada cert temps per no acumular a llarg termini si no es desitja. És una opció no implementada però fàcil de fer amb nodes connectats a Influx.

La resta del manteniment és fàcil. Si es creen noves versions del programari Android només cal disposar de una VPN o estar en la mateixa xarxa WiFi que la placa i es podrà reprogramar sense fils carregant via un sistema OTA el nou fitxer binari.

ANNEX A: Bibliografia

[binefa.com](#)

[stackoverflow](#)

[Android Developers](#)

[developer.mozilla.com](#)

[node-red](#)

[install node-red](#)

[OVH cloud](#)

[influx DB](#)

[things.cat](#)

[influxdata](#)

[Grafana Labs](#)

[SoftZone](#)

[Mosquitto broker](#)

[Collados.org](#)

[Sergi Grau GitHub](#)

