Treball de Fi de Grau

Grau en Enginyeria Electrònica de Telecomunicació

Sistema Industrial de Resposta Ràpida pel Monitoratge de Processos Químics

Guillem Companys Garcia

Director: Albert Crespo Yepes

Departament d'Enginyeria Electrònica

Universitat Autònoma de Barcelona

Bellaterra, Juliol 02, 2023

Abstract

This work focuses on the development of an advanced alarm and monitoring system for *Chemplate Materials S.L.*, a chemical company operating a pilot plant requiring automated sensor readings and responsive actions. Building on a previous project that established a simple web-based system for sensor readings, the present study aims to address its limitations, such as single-plant configuration and rudimentary action system, by creating a new multi-client web page with an easy-to-use interface. The project outlines the development of the web page, the complex alarm system design and the implementation of various actions, such as motor control, photography capture, and correlation analysis. The system is evaluated in a simulated environment using a replica setup and demonstrates potential for enhancing operational safety and process efficiency in pilot plants.

Resum

En aquest projecte es desenvolupa un sistema avançat d'alarmes i seguiment per a *Chemplate Materials S.L*, una empresa especialitzada en la distribució i producció de productes químics semielaborats. *Chemplate Materials* i els seus clients disposen de plantes pilot que necessiten lectures automatitzades de sensors i accions de resposta ràpida. A partir d'un projecte previ on es va crear una pàgina web senzilla per a la lectura de sensors, l'actual investigació busca superar certes limitacions del sistema anterior com la configuració exclusiva per a una sola planta i un sistema d'acció elemental. El projecte comprèn el desenvolupament d'una nova pàgina web multiclient i el disseny d'un sistema d'alarmes intel·ligent amb diverses possibles accions, com el control d'actuadors, la notificació de noves lectures dels sensors i la captura d'imatges i possibles anàlisis posteriors. El sistema també es posa a prova en un entorn simulat utilitzant una petita rèplica de la planta pilot.

Resumen

En este proyecto se desarrolla un sistema avanzado de alarmas y seguimiento para *Chemplate Materials S.L*, una empresa especializada en la distribución y producción de productos químicos semielaborados. *Chemplate Materials* y sus clientes disponen de plantas piloto que necesitan lecturas automatizadas de sensores y acciones de respuesta rápida. A partir de un proyecto previo donde se creó una página web sencilla para la lectura de sensores, la actual investigación busca superar ciertas limitaciones del proyecto anterior como la configuración del sistema exclusiva para una sola planta y un sistema de acción elemental. El proyecto comprende el desarrollo de una nueva página web multicliente y el diseño de un sistema de alarmas inteligente con diversas acciones posibles, como el control de actuadores, la notificación de nuevas lecturas de los sensores y la captura de imágenes y posterior análisis. El sistema también se pone a prueba en un entorno simulado utilizando una pequeña réplica de la planta piloto.

Agraïments

En primer lloc, voldria agrair al meu tutor, Albert Crespo Yepes la seva col·laboració i orientació durant aquest treball, i per la seva ajuda constant, interès i professionalitat, sent una valuosa guia per tirar endavant el projecte.

Alhora voldria expressar el meu agraïment a la Universitat Autònoma de Barcelona, en concret a l'Escola d'Enginyeria per deixar-me tot el material necessari i un espai on desenvolupar el treball.

Finalment, voldria agrair a la meva parella Lídia, a la meva família i als meus companys de grau i amics per donar-me consell i acompanyar-me durant el desenvolupament d'aquest projecte.

Continguts

Abstract **1**

Resum **2**

Resumen **3**

Agraïments **4**

Llistat de Figures **10**

Llistat de Taules **11**

1 Context del Treball **12**

1.1 Introducció 12

1.2 Objectius 13

1.3 Metodologia 14

1.4 Planificació 14

2 Sistema Industrial de Monitoratge de Processos Químics **15**

2.1 Estructura del Sistema 15

2.2 Arquitectura *Hardware* 16

2.3 Arquitectura *Software* 17

2.3.1 Node Servidor 17

2.3.2 Node Client 22

2.4 Interfície Web 23

2.5 Altres Funcionalitats implementades 26
2.5.1 *Grafana Reporter* 26
2.5.2 *Picockpit* 26

3 Sistema d'Alarmes **28**
3.1 Sistema d'Alarmes i accions 28
3.2 Creació del banc de treball 30
3.2.1 Elements del banc de treball 30
3.2.2 Disseny i construcció del banc de treball 31
3.3 Implementació del Sistema d'Alarmes 33

4 Implementació de les Accions **41**
4.1 Activar i desactivar actuador 41
4.2 Prendre una fotografia 45
4.2.1 Fer correlació de la mescla 47
4.3 Enviar un correu electrònic 50

5 Interfície Web **53**
5.1 Inici de Sessió i Selecció de Client 54
5.2 Panell de Control del Client 55
5.3 Configuració dels Sensors 56
5.4 Configuració dels Actuadors 57
5.5 Configuració de les Alarmes 58
5.6 Funcionalitats Addicionals 61
5.7 Tecnologies utilitzades 64

6 Resultats **67**

7 Conclusions **70**

Apèndix **72**

A.1 Descripció del codi *Javascript* de la pàgina web (*Frontend*) 72

A.2 Descripció del codi *Python* emprat a la pàgina web (*Backend*) 73

A.3 Programes per administrar els models de la pàgina web 73

A.3.1 Gestió d'Usuaris 73

A.3.2 Gestió de Clients 76

Bibliografia **80**

Llistat de Figures

1.1 Planificació del projecte 14

2.1 Estructura de SIMPQ amb un sol client 16

2.2 *Raspberry Pi 4B 4GB* amb l'adaptador *Modbus-USB* utilitzat a SIMPQ 17

2.3 *Stack Software* del node Servidor de SIMPQ 18

2.4 Esquema de comunicació *MQTT* 19

2.5 *Stack Software* del node Client de SIMPQ 22

2.6 Pàgina web principal de SIMPQ 24

2.7 Pàgina web configuració de SIMPQ 24

2.8 Arxiu *JSON* d'exemple amb la configuració de 3 sensors 25

2.9 Captura de *Picockpit* on es mostra l'estat del node Servidor i el node Client . . . 27

3.1 Diagrama amb totes les possibles combinacions de les alarmes 29

3.2 Bomba peristàltica *Kamoer* 30

3.3 Mòdul *PiCamera* 30

3.4 Pantalla *LCD* compatible amb la *Raspberry Pi* 31

3.5 Disseny del banc de treball 32

3.6 Banc de treball amb els elements ja instal·lats 33

3.7 Diagrama del procés d'enviament de les configuracions del nou programa 34

3.8 Diagrama de flux del node Servidor 35

3.9 Nou *Stack Software* del node Servidor 36

3.10 Diagrama de flux del node Client 37

3.11 Tipus de tasques del node Client 39

3.12 Nou *Stack Software* dels nodes Client 40

4.1 Botons de control de la bomba peristàltica 42

4.2 Interfície *Modbus* de la bomba peristàltica 42

4.3 Instrucció per activar la comunicació i engegar la bomba 43

4.4 Programa de prova de la bomba peristàltica 43

4.5 Imatge de la bomba en funcionament 44

4.6 Arxiu *JSON* per importar les instruccions *Modbus* de cada model d'actuador . . 45

4.7 Ínterficie gràfica del programa de la càmera 46

4.8 Les diferents opcions de configuració del programa de la càmera 46

4.9 Interfície gràfica amb una imatge presa 47

4.10 Possibles estats de la mescla 48

4.11 Informes generats pel programa 49

4.12 Correu electrònic de prova amb informació d'un sensor fictici 51

4.13 Correu electrònic amb una (esquerra) i múltiples lectures (dreta) 52

4.14 Correu electrònic amb una imatge (esquerra) i un informe adjunt (dreta). 52

5.1 Pàgina d'inici de sessió 54

5.2 Selector de clients 54

5.3 Panell de control del client 55

5.4 Formulari de configuració dels sensors sense cap sensor configurat 56

5.5 Pàgina per configurar els sensors amb 4 sensors configurats 57

5.6 Formulari de configuració dels actuadors sense cap actuador configurat 57

5.7 Pàgina per configurar els actuadors amb 2 actuadors configurats 58

5.8 Formulari de configuració de les alarmes sense cap alarma configurada 59

5.9 Pàgina per configurar les alarmes on el formulari està sent emplenat 61

5.10 Pàgina per configurar les alarmes amb 4 alarmes configurades 61

5.11 Barra de navegació lateral 62

5.12 Panell lateral amb informació de l'usuari 63

5.13 Pàgina de tancament de sessió 63

5.14 Stack Web 64

5.15 Models de la pàgina Web 65

6.1 Sensor *ARCELI SHT20* utilitzat per llegir la temperatura i la humitat del laboratori 67

6.2 Imatge de la càmera final sense *LED IR* 68

6.3 Pantalla de la Raspberry Pi client amb el programa en execució 68

6.4 Banc de treball final 69

6.5 Panell de control de *Grafana* amb les lectures de temperatura i humitat del sensor 69

Llistat de Taules

5.1 Descripció dels models utilitzats 66

Capítol 1

Context del Treball

1.1 Introducció

Chemplate Materials S.L. és una empresa centrada en la distribució i fabricació de productes químics semielaborats, que compta amb un ampli ventall de productes químics que donen cobertura a múltiples tipus de processos en galvanotècnia i electrònica.

Mitjançant la marca *Indubond*, també comercialitzen les màquines per fer aquests processos per tot el món. *Chemplate* i els seus clients disposen de plantes pilot on es prototipen nous tractaments fent proves a petita escala. D'aquesta manera, el cost és molt reduït i assequible. Les plantes estan equipades amb diversos tancs i diferents solucions controlades a través de sensors de tota mena. Durant el procés de fabricació, els químics controlen magnituds de les mescles com el pH, la temperatura o la conductivitat i modifiquen la composició dels banys si és necessari.

Avui dia, aquest protocol de control és majoritàriament manual i els tècnics especialistes de *Chemplate* han de desplaçar-se i visitar presencialment les fàbriques dels clients per tal de recollir i analitzar les mostres. Aquest procés sol durar una setmana i l'automatització d'aquestes mesures suposa un estalvi econòmic significatiu i un augment evident de la productivitat.

Per aquest motiu, en un treball anterior es va desenvolupar un sistema per llegir els sensors de manera automàtica on les lectures eren enviades a una base de dades per a la seva posterior consulta a través d'una pàgina web [1]. Aquest projecte, malgrat ser un sistema funcional, presentava diverses limitacions que dificultaven la seva utilització a gran escala. D'una banda, tot i que el sistema seguia un paradigma *client-servidor*, no disposava de la infraestructura necessària per configurar-se per a més d'un client alhora. D'altra banda, el sistema d'acció i resposta implementat per reaccionar a les mesures obtingudes era poc flexible, molt bàsic i rudimentari.

El present projecte busca millorar el sistema creat en el treball anterior desenvolupant un sistema avançat d'alarmes que permeti als operaris de l'empresa prendre accions de forma ràpida, automatitzada i eficient. Abans de definir l'abast i els objectius del present projecte, a continuació es fa un petit resum del funcionament d'aquest sistema desenvolupat durant el projecte anterior.

1.2 Objectius

Aquest treball té com a objectiu principal crear un sistema avançat d'alarmes que permeti als operaris de *Chemplate* prendre accions de forma ràpida, automatitzada i eficient. Així i tot, a continuació es mostren propòsits específics que descriuen certes funcionalitats addicionals que es volen implementar:

- Desenvolupar i implementar un sistema d'alarma avançat per a la detecció i resposta ràpida a potencials perills en la línia de producció. Aquest sistema haurà d'incloure:
  - La generació automàtica d'informes i notificacions als operaris en cas d'alarma.
  - La possibilitat d'aplicació automàtica d'accions correctives amb l'opció de requerir autorització per part d'un operari.
  - Un sistema de notificació d'alarma específic per a la detecció d'escuma en els banys i l'alerta a l'operari més proper.
- Crear una interfície d'usuari intuïtiva per a la configuració i monitoratge dels sensors, actuadors i alarmes del sistema. Aquesta interfície haurà de:
  - Mostrar l'estat dels sensors, actuadors, alarmes i els últims *logs* en temps real.
  - Utilitzar fitxers *JSON* per importar la informació dels sensors, actuadors i alarmes.
- Implementar un sistema de visualització en temps real amb un plànol general de la línia per a la detecció ràpida d'anomalies, que inclogui una càmera d'alta definició per a l'estudi de l'estat de les mescles a través de mètodes com la correlació d'imatges.
- Assegurar la connexió directa i compatibilitat entre el sistema i la planta pilot, incloent-hi els sensors i actuadors.
- Dissenyar el sistema de tal manera que faciliti l'adaptabilitat i l'escalabilitat per a la seva implementació en diferents entorns i situacions d'operació en el futur.

1.3 Metodologia

La metodologia utilitzada durant aquest projecte ha estat *Kanban*, una metodologia que es basa en la gestió de flux de treball eficient i en l'adaptació ràpida als canvis [2].

Aquesta visualització del flux de treball permet als membres de l'equip veure l'estat actual de les tasques i identificar possibles colls d'ampolla en el procés. Això es pot aconseguir mitjançant l'ús d'un tauler *Kanban*, que representa les diferents etapes del procés (en el nostre cas, "pendent", "en curs" i "finalitzat") i les tasques que es troben en cada etapa.

1.4 Planificació

Tal i com es mostra a la Figura 1.1, les tasques del projecte es poden dividir en 6 grups:

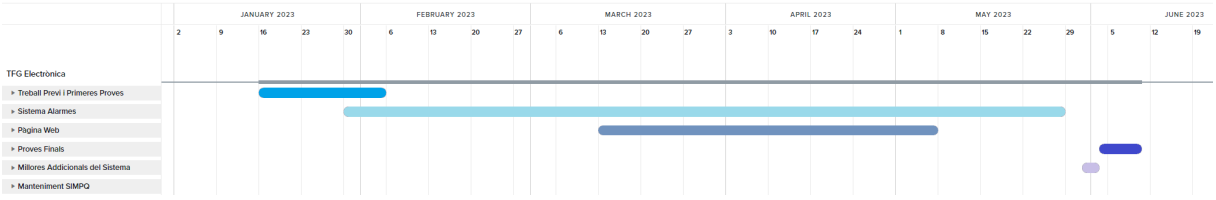

Figure 1.1: Planificació del projecte

- **Treball Previ i Primeres Proves:** conjunt de tasques dutes a terme a principis del projecte per preparar l'entorn de treball i comprovar el correcte funcionament de les diferents parts del sistema.
- **Sistema d'Alarmes:** bateria de tasques que comprenen la creació de diversos programes relacionats amb les possibles accions del sistema.
- **Pàgina Web:** compilació de tasques relacionades amb el disseny i implementació d'una nova pàgina web amb suport per més d'un client.
- **Proves Finals:** tasques a acomplir per examinar el funcionament del sistema complet en un entorn de treball realista.
- **Millores Addicionals del Sistema:** conjunt de tasques addicionals que tenen com a objectiu millorar algun aspecte tècnic del projecte.
- **Manteniment SIMPQ:** tasques heretades del projecte anterior.

El Software per realitzar el seguiment d'aquest projecte ha estat *Trello* [3].

Capítol 2

Sistema Industrial de Monitoratge de Processos Químics

A finals de 2022 i principis de 2023 vaig desenvolupar el "Sistema Industrial de Monitoratge de Processos Químics (SIMPQ)" com a Treball de Fi de Grau d'Enginyeria Informàtica [1]. L'esmentat projecte, el qual s'utilitza com a punt de partida per a la realització d'aquest, buscava desenvolupar un sistema específic per a la planta pilot de *Chemplate*, que inclogués la lectura de sensors i l'enviament de la informació al servidor per al seu processament i anàlisi.

També es proporcionava als usuaris la capacitat de personalitzar l'interval de temps per a la recollida de dades. A més, es va crear una pàgina web per configurar els sensors i accedir a les dades recollides, permetent també la visualització de la informació a través de gràfics, taules o figures personalitzades. A continuació es mostra el funcionament d'aquest primer projecte amb la finalitat de contextualitzar el present treball.

2.1 Estructura del Sistema

SIMPQ està dissenyat seguint el model de programació *client-servidor* descrit a la Figura 2.1. Es va escollir aquest paradigma, ja que es pretenia tenir un nombre elevat de clients connectats simultàniament i accedir a la informació de cada client des d'un lloc centralitzat. Un dels avantatges d'aquesta estructura és la delegació de les tasques de recol·lecció i processament de la informació rebuda dels sensors.

Per aquest motiu, cada client controla els seus respectius sensors i envia la informació ja processada al servidor. D'aquesta manera s'aconsegueix alliberar recursos del node Servidor i aquest pot atendre a més nodes del sistema. Tot i que el projecte està preparat per treballar amb múltiples clients, per a la realització pràctica del treball es va optar per utilitzar-ne un.

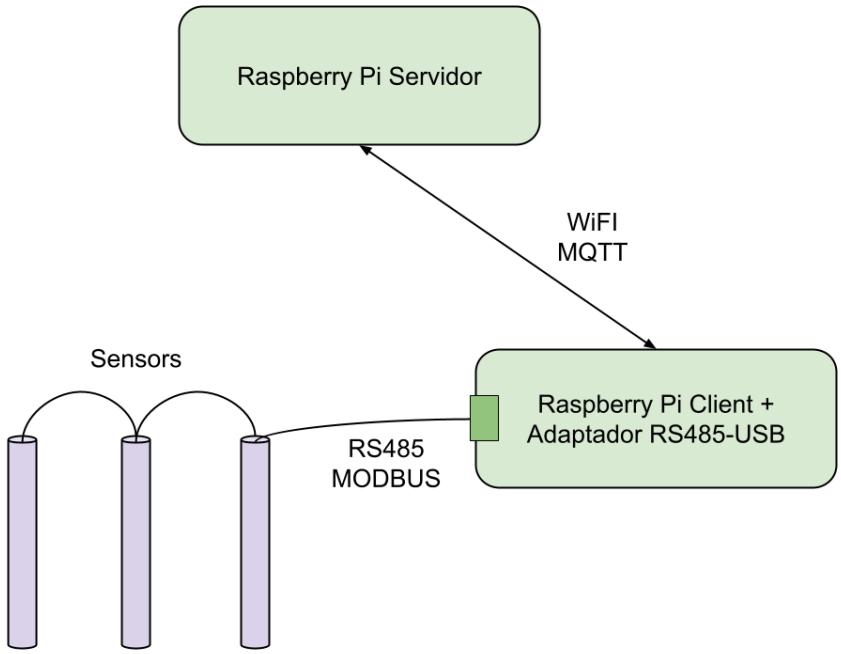

Figure 2.1: Estructura de SIMPQ amb un sol client

En els següents apartats es descriu el *Hardware* i *Software* que presenta el projecte, els protocols utilitzats i les característiques del sistema.

2.2 Arquitectura *Hardware*

SIMPQ es troba desplegat sobre dos *Raspberry Pi 4B*, on una placa pren el paper de servidor i l'altra el paper de client. Es van fer servir aquestes plaques, ja que la *Raspberry Pi 4B* és un microprocessador de baix cost, relativament potent, amb àmplia documentació i molt recomanable com a placa de prototipatge.

Així i tot, es va estudiar la possibilitat de desplegar el projecte en un servidor *cloud*. D'aquesta manera, només seria necessari *Raspberry Pi* que actuessin com a clients.

Existeixen una gran varietat d'empreses que ofereixen servidors d'aquest tipus: *Microsoft*, *Amazon*, *IBM*... Un desplegament *cloud* del servidor podria aportar resiliència al sistema i permetria delegar certes tasques de manteniment.

Els models que es van adquirir per a la realització del projecte compten amb 4 GB de *RAM* i fan ús d'un processador *ARM Quad Core Cortex-A72* a 1,5 GHz [4]. També disposen de connexió *Wifi* i *Ethernet*, essencial per establir comunicacions amb altres nodes. Aquests microprocessadors també disposen de un sistema operatiu molt similar a *Debian*, nomenat *Raspbian*.

En la Figura 2.2 es pot observar l'adaptador per llegir els sensors juntament amb el microprocessador utilitzat.

Figure 2.2: *Raspberry Pi 4B 4GB* amb l'adaptador *Modbus-USB* utilitzat a SIMPQ

2.3 Arquitectura *Software*

En aquesta secció es descriu el *software* de SIMPQ tant del node Servidor com dels nodes Client.

2.3.1 Node Servidor

La majoria de programari desplegat a la *Raspberry* Servidor s'executa sobre *Docker*, una plataforma de *software* que permet crear, provar i implementar aplicacions ràpidament. *Docker* empaqueta el *software* en unitats estandarditzades anomenades contenidors, que inclouen tot el necessari perquè el programa s'executi [5].

Els serveis i aplicacions necessàries per al correcte funcionament del sistema es despleguen a través d'un *docker-compose*, un arxiu que s'utilitza per definir i configurar els contenidors. A més a més, el sistema compta amb un arxiu de configuració addicional on es poden modificar paràmetres interns del sistema, com noms d'usuaris o contrasenyes.

La *Raspberry Pi* Servidor té instal·lat el Sistema Operatiu *Raspbian*. Sobre seu, es troben instal·lats *Docker*, *Python* i el *software* de monitoratge. SIMPQ fa servir 5 contenidors:

- El contenidor *InfluxDB*, que és l'encarregat d'emmagatzemar les lectures dels sensors dels

clients.

- El contenidor *Grafana*, aplicació que s'empra per mostrar la informació rebuda de manera visual (taules, figures, gràfics...). Existeix també un contenidor addicional amb una imatge de *Grafana* modificada, que és necessària per poder implementar una funcionalitat clau del sistema que s'explica més endavant.
- El contenidor *Flask*, que fem ús per programar i desplegar la pàgina web.
- El contenidor *Mosquitto*, el qual utilitzem per implementar el protocol de comunicació *MQTT*.
- El contenidor *Node-RED*, que ens permet automatitzar les respostes del sistema a variacions i errors en les mescles que s'han de corregir.

A continuació es mostra l'arquitectura del node Servidor (Figura 2.3) i les diferents capes de *software* que la formen.

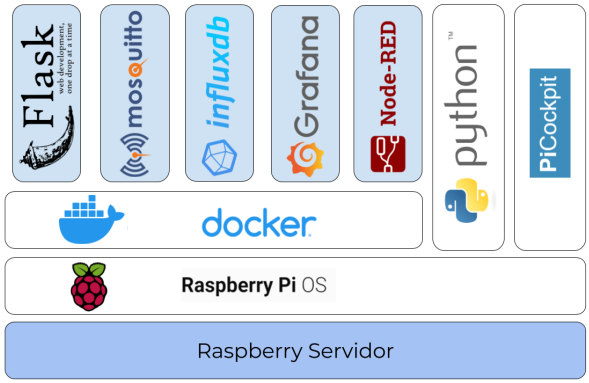

Figure 2.3: *Stack Software* del node Servidor de SIMPQ

2.3.1.1 Recull de les tecnologies principals usades al node Servidor

Les tecnologies principals que es troben al node Servidor de SIMPQ són les següents:

- ***MQTT* (*MQ Telemetry Transport*):** És un protocol de xarxa lleuger, de tipus *publicació-subscripció*, i màquina a màquina per al servei de cues de missatges [6]. Està dissenyat per a connexions amb ubicacions remotes, que tenen dispositius amb restriccions de recursos o una amplada de banda de xarxa limitada. Ha d'executar-se sobre un protocol de transport que proporcioni connexions ordenades, sense pèrdues i bidireccionals. En el cas d'aquest projecte *MQTT* utilitza *TCP/IP*.

  El protocol *MQTT* defineix dos tipus d'entitats de xarxa: un broker de missatges i una sèrie de clients.

- Un broker *MQTT* és un servidor que rep tots els missatges dels clients i després encamina els missatges als clients de destí apropiats.
- Un client *MQTT* és qualsevol dispositiu que es connecta i comunica a un broker *MQTT* a través d'una xarxa.

En aquest protocol la informació s'organitza per *topics*. Quan un client té una nova dada a distribuir, envia un missatge al broker, indicant el *topic* on vol escriure. A continuació, el broker distribueix la informació als clients que s'hagin subscrit a aquest *topics*. Aquest protocol permet delegar la comunicació a mig i baix nivell al servidor, ja que els clients no necessiten saber la morfologia i els detalls de la xarxa. Un exemple d'aquest protocol de comunicació es pot veure a la Figura 2.4

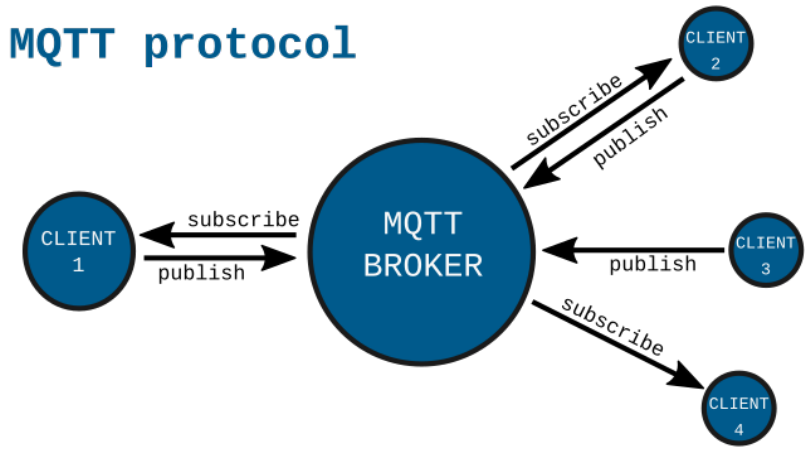

Figure 2.4: Esquema de comunicació *MQTT*

Per a la realització de SIMPQ es va optar per implementar el broker *Eclipse Mosquitto*, un *software open source* encapsulat dins *Docker* [7].

- ***InfluxDB*:** És una base de dades de sèries temporals (TSDB) *open source* ideal pel monitoratge d'operacions, de mètriques d'aplicacions o de dades de sensors de la Internet de les Coses (*IoT*) [8].

  InfluxDB és una base de dades que opera amb tres tipus d'estructures de dades: amb *measurements* (mesures), *series* (sèrie) i *points* (punts). Cada punt es compon de diverses parelles *clau-valor* denominats *fieldset* i una marca de temps. Un conjunt d'aquests punts agrupats per una parella *clau-valor* específica formen una sèrie. Un conjunt d'aquestes sèries agrupades amb un identificador formen una mesura.

  InfluxDB s'organitza en *buckets*, una localització específica de la base de dades on s'emmagatzema informació. Dins de la mateixa instància d'*InfluxDB* es poden tenir diferents *buckets*, de manera que les dades es troben més ordenades i són més fàcils d'accedir.

  Per tal d'inserir i llegir dades, *InfluxDB* compta amb dos llenguatges propis: el primer d'ells, *InfluxQL*, és similar sintàcticament a *SQL* i és el llenguatge originari de la base de

dades. En posteriors versions de la base de dades es va introduir *Flux*, una alternativa més senzilla d'utilitzar i més flexible [9]. Per a la lectura i escriptura d'informació a la base de dades SIMPQ fa ús d'aquest segon llenguatge.

- ***Grafana*:** És un component molt popular en serveis de monitoratge, sovint emprat en combinació amb bases de dades de sèries temporals com *Prometheus*, *Graphite* o, en el nostre cas, *InfluxDB* [10]. Aquesta aplicació web d'anàlisi i visualització interactiva és *open source* i multiplataforma, proporcionant una àmplia varietat d'eines i funcionalitats com taules, gràfics, alertes i panells de control personalitzables.

  Grafana permet crear panells de control complexos utilitzant constructors de consultes interactives que faciliten l'accés a les dades i les representen de manera intuïtiva i eficient. A més, ofereix una gran quantitat de *plugins* i integracions amb altres eines i serveis, com *Elasticsearch*, *AWS CloudWatch*, *Microsoft SQL Server* i molts més.

  Una de les funcionalitats més importants de *Grafana* és el seu sistema d'alertes, que permet als usuaris establir i gestionar condicions específiques per a les dades monitorades. Això permet als administradors de sistemes ser notificats en temps real quan es produeixen canvis significatius o es superen llindars preestablerts. *Grafana* permet configurar aquestes alertes de manera flexible, usant expressions i llindars específics per a cada cas d'ús. A més, les alertes poden ser enviades a diferents canals de notificació com ara correu electrònic, *Slack*, o, en el cas de SIMPQ, un *webhook* de *Node-RED*.

- ***Node-RED*:** És una eina de programació visual àmpliament utilitzada per a connectar diferents dispositius, *hardware* i *software*, amb *APIs* i serveis en línia [11]. Aquesta eina, desenvolupada per *IBM*, està dissenyada per facilitar el desenvolupament d'aplicacions de l'Internet de les Coses (*IoT*) i la comunicació entre dispositius.

  La seva interfície web ens ofereix un editor visual que facilita la creació de *workflows* o fluxos de treball fent servir l'àmplia varietat d'opcions i llibreries compatibles. Aquests fluxos de treball són creats mitjançant la connexió de diferents blocs funcionals, anomenats nodes, que representen funcions específiques com ara la lectura de sensors, el processament de dades o l'enviament de missatges.

  En el projecte SIMPQ, quan un sensor sobrepassa cert llindar, *Grafana* alerta automàticament a *Node-RED* a través d'un *webhook* propi. Aquest *webhook* és una funcionalitat que permet a *Grafana* comunicar-se amb *Node-RED* mitjançant una petició *HTTP*, enviant informació rellevant sobre l'alerta generada. Això permet a *Node-RED* processar aquesta informació i prendre les accions necessàries, com ara ajustar els paràmetres del sistema, per corregir les variacions o errors detectats en les mescles dels cubells.

  En el cas de SIMPQ, es va crear una alerta específica per a quan el valor del sensor superava un llindar establert prèviament. Aquesta alerta, generada per *Grafana*, es comunicava

automàticament amb *Node-RED* mitjançant un *webhook*. *Node-RED*, al seu torn, canviava l'estat d'un pin *GPIO* de la *Raspberry* Client, provocant una acció en resposta a aquesta situació. Tèoricament, aquest pin *GPIO* era l'encarregat de controlar una bomba peristàltica. Tal i com es mostra al capítol 4, el control d'aquesta bomba és força més complex i requereix l'enviament d'instruccions específiques a través del protocol *Modbus*.

La implementació d'aquesta alerta com a prova de concepte va demostrar tenir cert potencial. No obstant això, es va descartar la idea, ja que la creació d'alertes requeria intervenció humana i accedir directament a *Grafana*. *Grafana*, tot i presentar una interfície intuïtiva, pot ser complicada per a usuaris no experts. Per aquest motiu, aquesta solució no resultava idònia per *Chemplate Materials S.L.* A més, aquesta implementació no proporcionava la facilitat d'ús i l'escalabilitat desitjada.

Per aquest motiu, en aquest nou projecte es busca crear un sistema que sigui molt fàcil d'utilitzar i al mateix temps molt potent, de manera que pugui adaptar-se a les necessitats de *Chemplate* sense requerir coneixements especialitzats en la configuració d'alertes. Aquest nou sistema hauria de ser capaç d'automatitzar la creació i gestió d'alertes, simplificar la configuració per als usuaris i proporcionar una solució més escalable i flexible que permeti un major control i eficiència en el monitoratge i resposta a variacions i errors en les mescles dels cubells.

- ***Python*:** És un llenguatge de programació d'alt nivell, interpretat i orientat a objectes. Destaca per la seva simplicitat, llegibilitat i versatilitat i la seva sintaxi clara i senzilla el fa fàcil d'aprendre i d'emprar.

  Python ofereix una àmplia varietat de llibreries i mòduls de tercers, que permeten als desenvolupadors estendre les seves funcionalitats i abordar una gran varietat de tasques, com ara desenvolupament web, anàlisi de dades, intel·ligència artificial, automatització i molts més. *Python* és compatible amb múltiples sistemes operatius, com ara *Windows*, *macOS* i *Linux*, i permet la creació d'aplicacions multiplataforma.

  Tots els nodes de SIMPQ estan controlats per un script *Python* dividit en diverses llibreries pròpies, per tal de millorar la llegibilitat i reusabilitat del codi.

  El programa *Python* del node Servidor compta amb les llibreries següents:

  - **Llibreria *MQTT***: és l'encarregada de permetre la comunicació entre els nodes del sistema.
  - **Llibreria *InfluxDB***: és la utilitzada per enviar la informació dels sensors rebuda per *MQTT* a la base de dades.
  - **Llibreria *Email***: llibreria per enviar correus electrònics que en SIMPQ s'usa per enviar un resum de les lectures del dia en format *PDF*.

2.3.2 Node Client

El programari de les *Raspberry* Client consta d'un *script Python*, el *software* de monitoratge i una instància de *Node-RED*.

Per aquest motiu, el *stack* de la *Raspberry* només compta amb el Sistema Operatiu, el llenguatge de programació *Python* i una instància de *Node-RED* (Figura 2.5) .

Figure 2.5: *Stack Software* del node Client de SIMPQ

2.3.2.1 Recull de les tecnologies principals usades als nodes Client

Les tecnologies principals que es troben al node Client de SIMPQ són les següents:

- ***Modbus*:** És un protocol de comunicació basat en un model mestre-esclau o client-servidor, creat l'any 1979 per *Modicon* (ara *Schneider Electric*) per a la comunicació entre dispositius electrònics a través de xarxes [12]. Originalment desenvolupat per a controladors lògics programables (*PLCs*), el protocol *Modbus* s'ha convertit en un estàndard de la indústria per a la comunicació entre dispositius industrials i automatització de processos.

  Modbus pot ser implementat sobre diferents tipus de xarxes, incloent-hi la comunicació serial (*RS-232*, *RS-422* i *RS-485*) i *Ethernet* (*Modbus TCP/IP*). El protocol utilitza un esquema d'adreçament únic que permet la comunicació entre un dispositiu mestre (o client) i múltiples esclaus (o servidors) al mateix temps, facilitant la recopilació d'informació i el control de dispositius en un sistema d'automatització.

  El protocol *Modbus* opera amb dades numèriques i suporta diferents tipus de registres, com ara registres de bobines (valors binaris), registres d'entrada discreta (valors binaris de lectura), registres d'entrada (valors de lectura analògics) i registres de manteniment (valors analògics de lectura i escriptura). Aquests registres permeten la manipulació i monitoratge de dades en dispositius connectats a través del protocol.

  En el cas de SIMPQ, *Modbus* es troba implementat sobre el protocol sèrie *RS-485* i s'empra per comunicar-se amb el controlador de la planta pilot per llegir els valors de les lectures dels sensors.

- ***Node-RED*:** La instància de *Node-RED* del node client és l'encarregada de comunicar-se amb el motor. Aquest motor, control·lat a través del protocol *Modbus*, es connecta a la *Raspberry Pi* i es controla a través de crides a funcions *Modbus* que es transformen en senyals diferencials. En el capítol 4 existeix una explicació més extensa sobre el funcionament de la bomba peristàltica que es busca fer servir per addicionar substàncies i corregir les mescles. Durant la implementació de SIMPQ es va simplificar el funcionament del motor, controlant-lo mitjançant un únic *GPIO* que alternava entre estat lògic alt i estat lògic baix. L'estat lògic alt activava el motor, mentre que l'estat lògic baix el parava.

- ***Python*:** El programa *Python* de la *Raspberry* Client compta amb les llibreries següents:
  - **Llibreria *MQTT***: és l'encarregada de permetre la comunicació entre la Raspberry Client i la Servidora.
  - **Llibreria *Modbus***: és la utilitzada per comunicar-se amb els sensors per tal de llegir-los.
  - **Llibreria de Control**: és la llibreria que permet interconnectar totes les parts del programa i defineix la lògica de control del sistema.

  També existeix una classe de tipus *Sensor* on s'emmagatzema tota la informació relativa als sensors: nom, tipus, interval de lectura, funció *Modbus* a usar... entre altres.

2.4 Interfície Web

La pàgina web de SIMPQ fa ús dels llenguatges *HTML*, *CSS* i *Javascript*. Aquest últim llenguatge és l'utilitzat per modificar el comportament de la pàgina en temps d'execució. El *back-end* de la pàgina fa servir *Flask*, un *microframework* web programat en *Python* [13].

Una vista de la pàgina web de SIMPQ es troba recollida a la Figura 2.6. Tal com es pot observar, a través de la pàgina web l'usuari té la possibilitat de mostrar les lectures dels sensors per pantalla, configurar-los, consultar directament la base de dades o configurar el sistema d'alarmes. També existeixen funcionalitats addicionals com la possibilitat d'enviar un resum de les lectures per correu o consultar l'estat actual del sistema.

Figure 2.6: Pàgina web principal de SIMPQ

Per a la consulta de les dades recollides dels sensors s'envia a l'usuari al panell de *Grafana* corresponent. Les opcions "*Base de datos*" i "*Configuración de los actuadores*" redirigeixen a l'usuari cap a la interfície web d'*InfluxDB* i *Node-RED* respectivament.

L'opció "*Generar Reporte del Sistema*" fa una crida al contenidor auxiliar de *Grafana* per tal de generar l'informe de resum desitjat i l'opció "*Estado del Sistema*" redirigeix a l'usuari cap al servei de monitoratge extern. Ambdós serveis es troben descrits en la secció 2.5

Quan l'usuari prem "*Configuración de los sensores*" l'operari es troba amb la interfície de la Figura 2.7.

Figure 2.7: Pàgina web configuració de SIMPQ

La pàgina ofereix un formulari web dinàmic amb els següents elements a emplenar: el nom del sensor a afegir, el tipus, l'interval de lectura desitjada i paràmetres específics com la funció

de *Modbus* a utilitzar per a la lectura o l'adreça i el nombre de registres a llegir.

Els sensors emprats a *Chemplate* es llegeixen a través del controlador *50 Series* de *Chemitec* [14]. Aquest dispositiu té un mapa de registres que accepta diferents funcions *Modbus* cadascuna amb els seus registres corresponents.

La pàgina web carrega de manera dinàmica els registres disponibles quan l'usuari selecciona la funció *Modbus* a usar. Per a fer-ho, quan es modifica el camp "Funció" del formulari es fa una crida *Javascript* on es comprova la funció seleccionada per després realitzar una càrrega dels registres des d'un arxiu. Aquest comportament dinàmic s'ha potenciat considerablement en el present projecte, tal com es detalla més profundament al capítol 5 d'aquest document. Els sensors configurats es recullen a la taula de la part superior de la Figura 2.7, on es mostra la informació de cada sensor juntament amb 3 accions possibles.

Quan es carrega la pàgina web per primer cop, la taula no existeix. No és fins que s'afegeix el primer sensor que aquesta es crea. Similarment, si s'elimina el sensor i la taula queda buida, aquesta s'elimina automàticament.

L'estat del sistema i la configuració dels sensors s'emmagatzema en un arxiu *JSON*. Quan es recarrega la pàgina s'accedeix a aquest fitxer per comprovar si ja existeix una configuració prèvia. Quan l'usuari guarda la configuració perquè sigui enviada a la *Raspberry* Client, *Javascript* envia una petició amb la informació de la taula a *Flask* en format *JSON* i *Flask* la guarda en un fitxer emmagatzemat a l'arrel del projecte.

En la Figura 2.8 es troba recollit una mostra d'aquest arxiu amb la configuració de tres sensors ficticis.

```
1  [
2  {
3  "nombre":"Sensor 1",
4  "tipodesensor":"Temperatura",
5  "intervalodetiempo":"Cada minuto",
6  "funciónmodbuslectura":"Function 01 (Read Coil Status)",
7  "dirección":"15",
8  "nºregistros":"2",
9  "comentarios":"Sensor del cubell 1"
10 },
11 {
12 "nombre":"Sensor 2",
13 "tipodesensor":"Presión",
14 "intervalodetiempo":"Una vez al día",
15 "funciónmodbuslectura":"Function 04 (Read Input Registers)",
16 "dirección":"18",
17 "nºregistros":"8",
18 "comentarios":"Sensor de la planta situat a la dreta del termòmetre"
19 },
20 {
21 "nombre":"Sensor 3",
22 "tipodesensor":"Otro",
23 "intervalodetiempo":"Cada hora",
24 "funciónmodbuslectura":"Function 16 (Preset Multiple Registers)",
25 "dirección":"20",
26 "nºregistros":"2",
27 "comentarios":"Nou Sensor de ORP"
28 }
29 ]
```

Figure 2.8: Arxiu *JSON* d'exemple amb la configuració de 3 sensors

La pàgina també permet a l'usuari eliminar cada sensor a través del botó "*Eliminar*". Per tal de guardar els canvis, s'ha de pitjar "*Guardar cambios*". Les modificacions no es guarden automàticament per tal de donar cert marge d'error a l'usuari. D'aquesta manera, si s'elimina un sensor per error, aquest no s'esborra del sistema fins que es prem el botó de confirmació.

2.5 Altres Funcionalitats implementades

2.5.1 *Grafana Reporter*

Un dels objectius de SIMPQ era generar de forma automàtica informes de seguiment dels sensors llegits. Tot i que *Grafana* presenta una opció de pagament per a fer-ho, es va decidir utilitzar un programa extern gratuït per a aconseguir-ho.

Es van crear petits casos d'ús, però, a causa de dificultats tècniques, es va optar per fer ús d'un servei extern ja establert [15], que s'executa sobre la *Raspberry* Servidor. El servei extreu el panell de *Grafana* i el converteix en un document *PDF* en format *Latex*.

Dins del *script Python* del node Servidor existeix un comptador que crida a aquest servei un cop al dia o quan es prem un botó de la pàgina web. Després de generar l'arxiu, aquest es pot enviar per correu a l'usuari.

L'enviament del *PDF* a través del correu és automàtic i es realitza a través de la llibreria *Email* implementada amb *Python* que s'ha descrit anteriorment.

2.5.2 *Picockpit*

Picockpit [16] és una eina de monitoratge i control de *Raspberry Pi* emprada pel sistema SIMPQ per supervisar els nodes del sistema. Aquesta aplicació proporciona una interfície web intuïtiva que permet als usuaris mantenir un seguiment dels seus dispositius i controlar-los de forma remota.

A més a més, *Picockpit* també ofereix funcions de diagnosi per avaluar l'estat dels dispositius i identificar possibles problemes. Això ajuda a assegurar que els dispositius funcionin de manera eficient i sense interrupcions, minimitzant els temps d'inactivitat i optimitzant el rendiment del sistema.

Picockpit recopila diverses estadístiques útils relacionades amb el rendiment dels dispositius *Raspberry Pi*, com ara l'ús de la *CPU*, la temperatura del processador, la quantitat de memòria *RAM* disponible, l'ús del disc dur i la velocitat de la xarxa, entre d'altres.

Aquestes estadístiques permeten als usuaris mantenir-se informats sobre el funcionament dels

seus dispositius i prendre mesures proactives per solucionar problemes abans que esdevinguin crítics.

En la Figura 2.9 es mostra el servei *Picockpit* amb dues *Raspberry Pi* configurades.

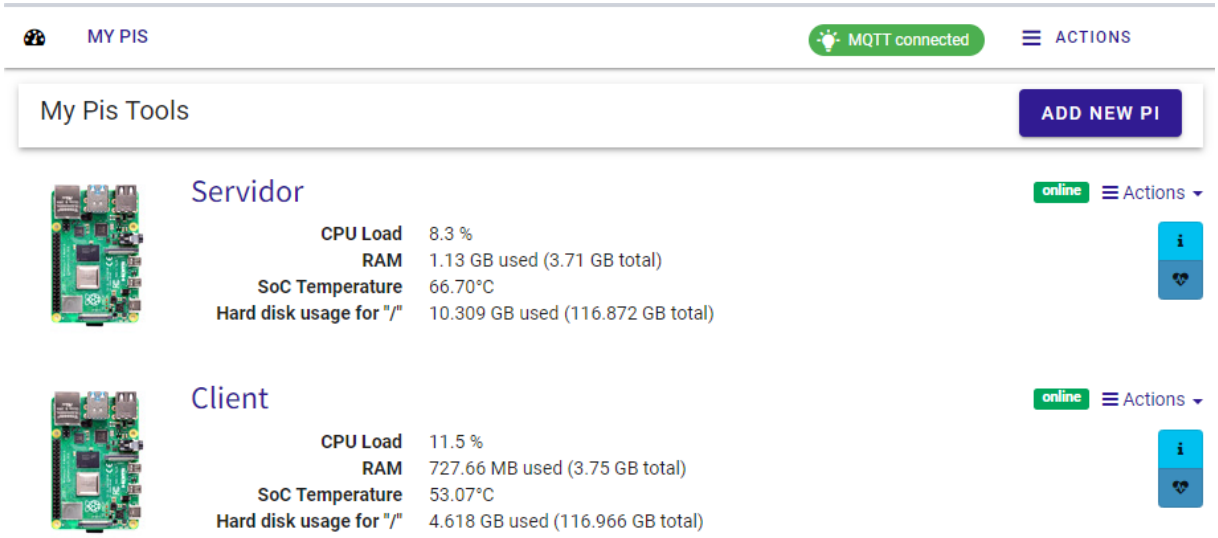

Figure 2.9: Captura de *Picockpit* on es mostra l'estat del node Servidor i el node Client

A més de les seves funcions de monitoratge i control, *Picockpit* també inclou funcions addicionals com ara la possibilitat de gestionar actualitzacions del sistema, reinicis i apagades dels dispositius, i configurar alertes personalitzades basades en llindars específics per ajudar a mantenir el sistema SIMPQ funcionant de manera òptima.

Capítol 3

Sistema d'Alarmes

En aquest capítol es descriu el sistema d'alarmes desenvolupat, les seves característiques i les possibles accions que aquest pot dur a terme. A més a més, es mostra el banc de treball utilitzat per desenvolupar el sistema d'alarmes i la seva implementació.

Tal com s'ha descrit al capítol 1, SIMPQ compta amb una arquitectura *client-servidor* on cada client és responsable de gestionar la seva pròpia planta pilot, incloent-hi la lectura dels sensors. Per desenvolupar el sistema d'alarmes, s'ha decidit crear una rèplica a petita escala del sistema on se simula la presència dels sensors. D'aquesta manera es pot desenvolupar i provar les alarmes en un entorn controlat. A continuació es fa una síntesi dels requisits d'aquest sistema i les seves accions.

3.1 Sistema d'Alarmes i accions

El Sistema d'Alarmes desenvolupat, dissenyat per a la detecció i resposta ràpida a potencials perills en la línia de producció ha de ser capaç de realitzar una sèrie d'accions automàtiques, que inclouen l'addició d'additius a les mescles dels cubells de les plantes pilot, l'enviament d'avisos amb les lectures dels sensors per correu electrònic, o la visualització en temps real dels comportaments dels banys, entre altres.

És per això que el Sistema d'Alarmes permet a l'usuari realitzar les següents accions:

- Activar o desactivar un actuador (motors, bombes, etc.). D'aquesta manera, es pot afegir els additius i reaccionar davant canvis sense la necessitat de la presència d'un operari.
- Prendre una fotografia a través d'una càmera instal·lada a la planta.
- Enviar un correu electrònic amb les últimes mesures d'un sensor per mantenir als usuaris informats sobre l'estat del sistema en tot moment.

A banda d'aquestes accions, també existeixen accions complementàries per l'acció "Prendre una fotografia", ja que és dona a l'usuari l'oportunitat de rebre la imatge per correu, guardar-la localment o analitzar-la per extreure informació. Aquesta última funcionalitat és especialment útil per a la detecció d'escumes en els banys i per monitorar el nivell dels additius del sistema. En la Figura 3.1 es mostra totes les accions del sistema amb les seves possibles combinacions.

La implementació de totes aquestes accions es troba emmarcada dins del capítol 4.

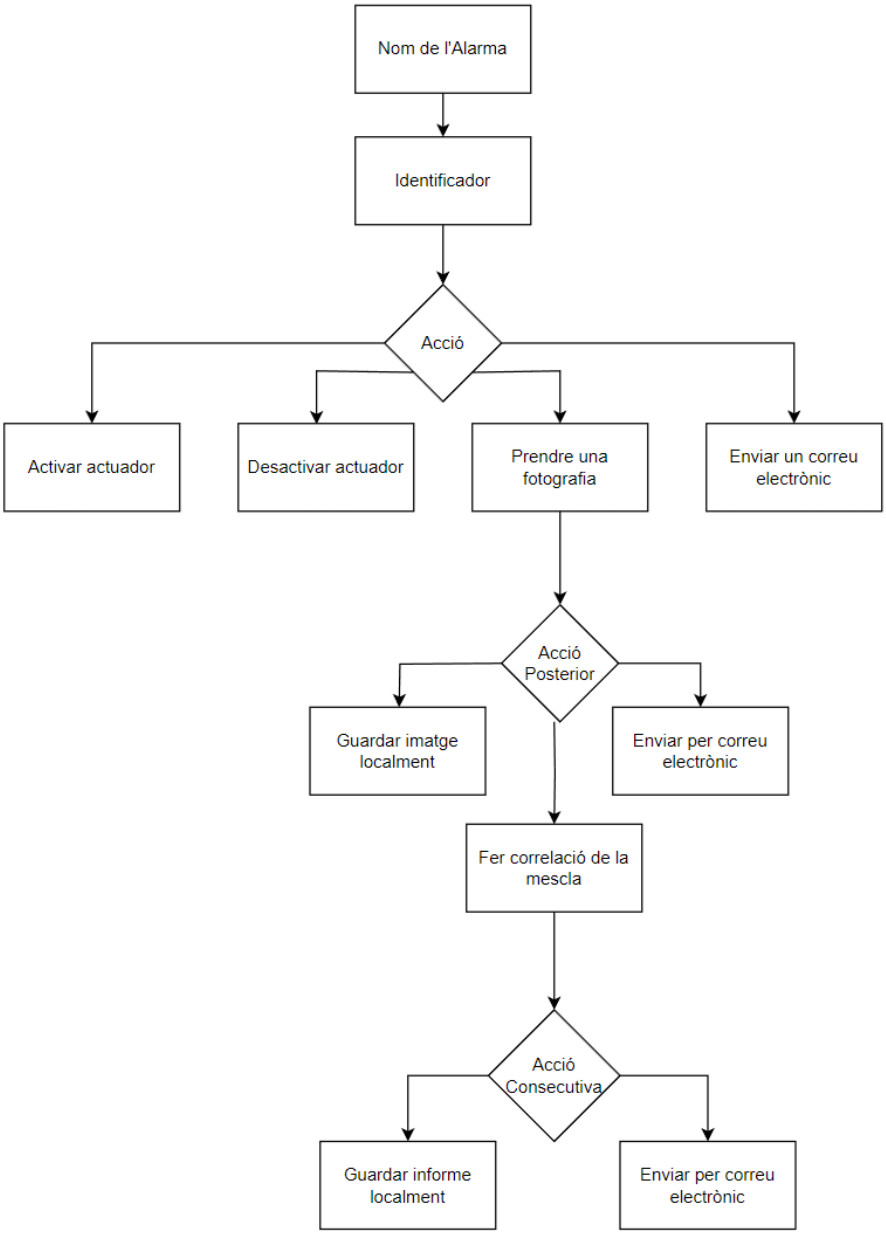

Figure 3.1: Diagrama amb totes les possibles combinacions de les alarmes

3.2 Creació del banc de treball

Per desenvolupar i provar adequadament el sistema d'alarmes, és essencial disposar d'un entorn de treball que simuli la planta pilot real. Aquest entorn ha de permetre provar les funcions del sistema d'alarmes en condicions que imiten les de l'entorn real.

3.2.1 Elements del banc de treball

Per tal de dissenyar el banc de treball es va procedir a la selecció dels components necessaris. A banda de la *Raspberry Pi*, com actuador s'ha optat per utilitzar la bomba peristàltica *Kamoer* de la Figura 3.2 ([17]). Aquest actuador compta amb una interfície de control *Modbus* i un conjunt de comandaments manuals que permeten accionar la bomba, augmentar la velocitat o modificar el sentit de gir d'aquesta.

Figure 3.2: Bomba peristàltica *Kamoer*

Com a càmera s'ha utilitzat un mòdul integrat *PiCamera* amb visió nocturna ([18]). Aquesta càmera, de 5 MP és capaç de capturar vídeo i imatges a una resolució de 1920x1080p.

Figure 3.3: Mòdul *PiCamera*

A fi de poder veure el programa del Sistema d'Alarmes en funcionament, també es va adquirir

una petita pantalla *LCD* de 5" com la que es pot veure a la Figura 3.4. Aquesta pantalla és l'encarregada de mostrar a l'usuari l'estat del programa i permet a aquest interactuar amb el sistema.

Figure 3.4: Pantalla *LCD* compatible amb la *Raspberry Pi*

3.2.2 Disseny i construcció del banc de treball

El procés de disseny i construcció del banc de treball va ser possible gràcies a l'ús de la plataforma Onshape, un programari CAD basat en el núvol que permet la creació, modificació, compartició i col·laboració de dissenys en 3D de forma eficient i eficaç.

Tot i començar sense coneixements previs de disseny i modelatge 3D, vaig aconseguir familiaritzar-me amb els conceptes bàsics de *Onshape*. Un dibuix acotat del banc es troba representat a la Figura 3.5.

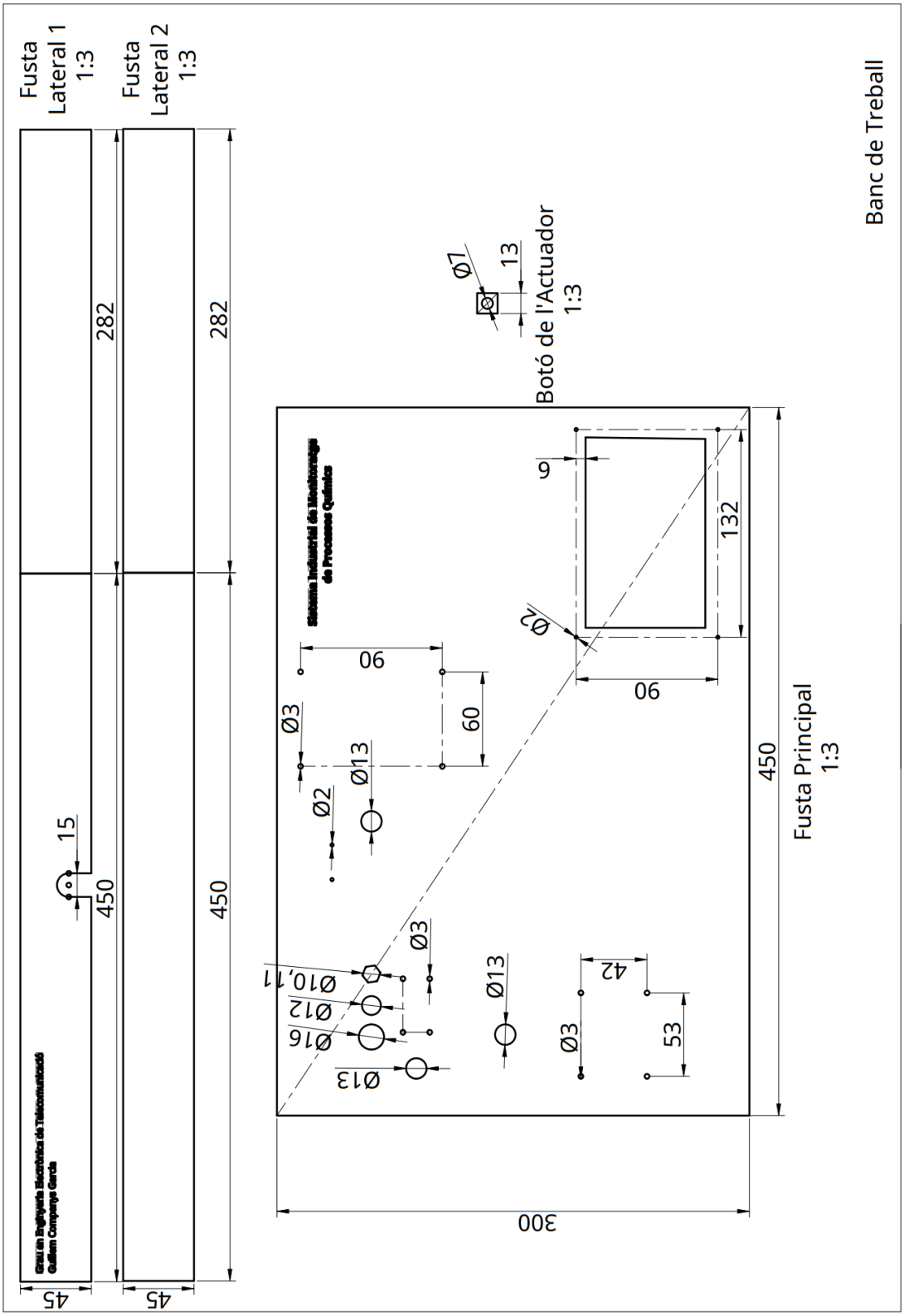

Figure 3.5: Disseny del banc de treball

Per a la construcció del banc, realitzada a l'espai de treball *OpenLabs* de la UAB ([19]) es va utilitzar la talladora làser del laboratori. Una imatge del banc amb els components anteriorment definits ja acoblats es pot veure a la Figura 3.6.

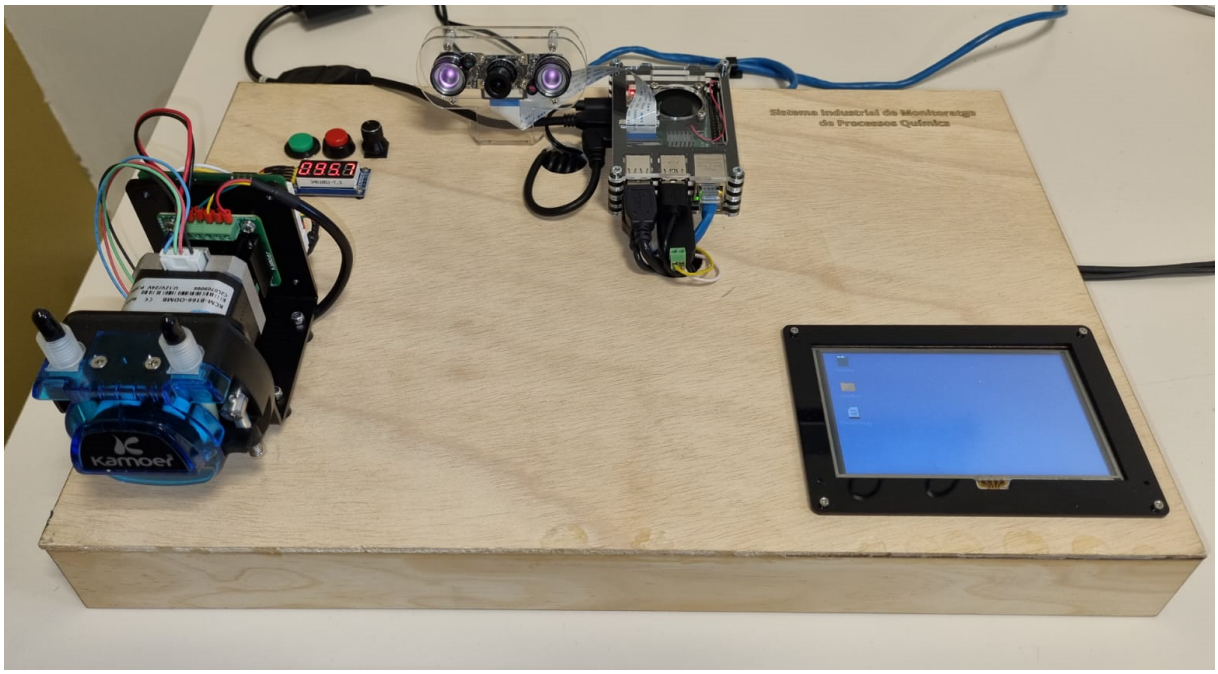

Figure 3.6: Banc de treball amb els elements ja instal·lats

Tal com es pot observar, el banc consta d'una superfície de fusta amb dos suports per donar-li profunditat. A la part inferior dreta es troba la pantalla, que es troba connectada amb la *Raspberry Pi* a través d'un cable *HDMI*.

Juntament amb la *Raspberry Pi* hi ha el mòdul de la càmera, que està format per una lent i dos *LED IR*. La bomba peristàltica ocupa la part esquerra del banc i darrere seu es troben els comandaments que permeten controlar-la. També hi ha una petita pantalla *7-segment display* de 4 dígits on es mostra la velocitat de la bomba en ml/min. La part inferior del banc s'utilitza per ocultar tot el cablejat del sistema.

3.3 Implementació del Sistema d'Alarmes

SIMPQ disposa d'una pàgina web on l'usuari pot configurar els sensors d'una manera fàcil i senzilla. Aquesta configuració després es converteix en un arxiu tipus *JSON* que el node Servidor envia al node Client.

El nou programa utilitza el mateix procés, tot i que aquest envia tres arxius de configuració: un arxiu de configuració pels sensors, un pels actuadors i un altre per les alarmes. Un diagrama

d'aquest procés es pot veure a la Figura 3.7. Els *JSON* de configuració són similars al de la Figura 2.8 del capítol 2.

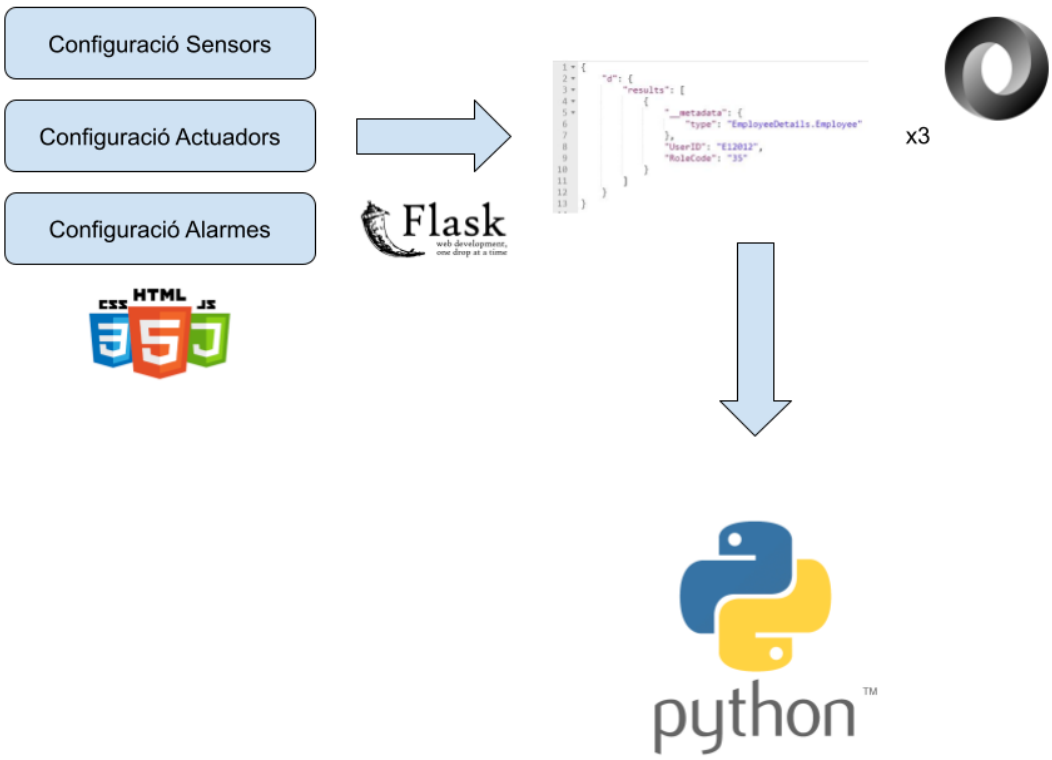

Figure 3.7: Diagrama del procés d'enviament de les configuracions del nou programa

Per tal d'enviar els diversos arxius de configuració a cada client s'ha creat un nou programa *Python* pel node Servidor. Aquest programa *Python*, que pren com a base el programa que es va dissenyar pel node Servidor de SIMPQ, compta amb una nova llibreria i una estructura de carpetes diferent.

Les diferents llibreries del programa Servidor són les següents:

- **Llibreria *MQTT***: és l'encarregada de permetre la comunicació entre la Raspberry Client i la Servidora.
- **Llibreria *InfluxDB***: és la utilitzada per enviar la informació dels sensors a la base de dades.
- **Llibreria FILE**: és la llibreria que controla i envia els arxius de configuració a cada client.

A banda d'aquestes llibreries, també existeix un fitxer on s'emmagatzemen variables compartides anomenat "*globals.py*" i un arxiu per guardar els *logs*, entre altres. Un diagrama del comportament del codi *Python* del node Servidor es pot veure a la Figura 3.8.

Tal com es pot observar, en iniciar-se el programa el node Servidor envia tots els arxius de configuració disponibles als clients corresponents. Un cop enviats, el node Servidor comprova si ha rebut noves lectures de sensors. Si és així, aquest procedeix a enviar la nova mesura a la base de dades a través de la llibreria *InfluxDB*. Tot seguit es comprova si s'ha modificat algun arxiu de configuració degut a canvis realitzats pels usuaris de la pàgina web. Si és així, s'envia a l'arxiu al client que pertoca. Aquestes dues comprovacions formen part d'un bucle que es repeteix indefinidament.

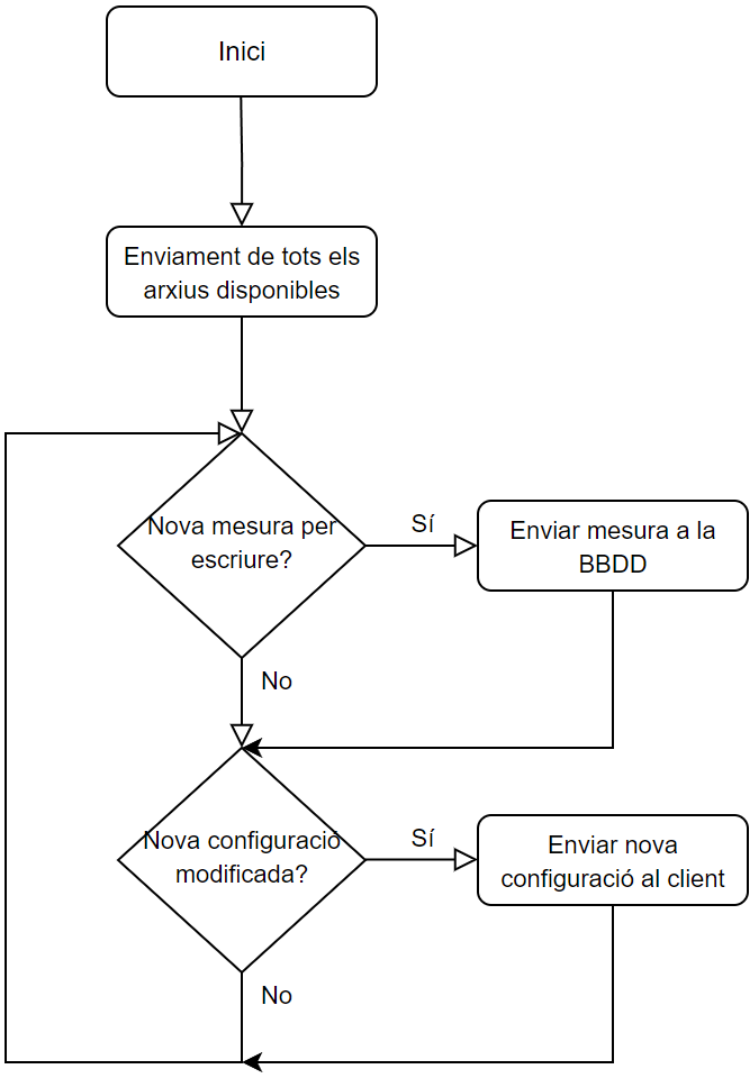

Figure 3.8: Diagrama de flux del node Servidor

A banda dels canvis realitzats en el programa *Python*, també s'ha modificat el *Stack Software* del node Servidor. Tal i com es pot veure a la Figura 3.9 s'ha eliminat la instància de *Node-RED*, ja que el Sistema d'Alarmes implementat reemplaça l'antiga funció d'aquest contenidor. També s'ha eliminat el contenidor *Grafana* addicional que forma part de la funció *Grafana Reporter* per incompatibilitats del contenidor amb l'arquitectura *hardware* del node.

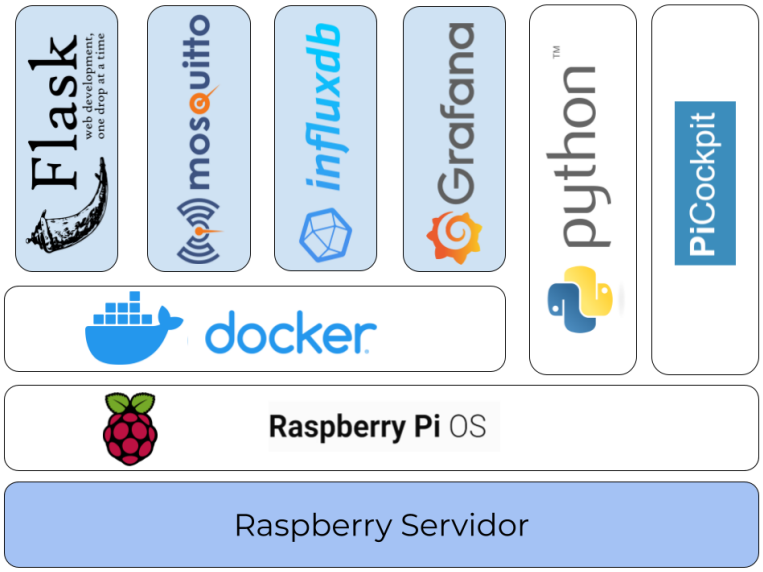

Figure 3.9: Nou *Stack Software* del node Servidor

De manera paral·lela, el programa *Python* dels nodes Client ha s'ofert nombrosos canvis. Tot i també utilitzar el programa dissenyat pels nodes Client de SIMPQ, la implementació del Sistema d'Alarmes ha forçat canvis profunds en la seva estructura i en els fitxers que componen el programa.

Les llibreries del programa del node Client són:

- **Llibreria *MQTT***: és l'encarregada de permetre la comunicació entre la *Raspberry* Client i la Servidora.
- **Llibreria *Modbus***: és la utilitzada per comunicar-se amb els sensors a llegir.
- **Llibreria de Control**: és la llibreria que permet interconnectar totes les parts del sistema.
- **Llibreria Email**: és la llibreria que s'utilitza per enviar les mesures, les imatges preses a la planta i els informes que es detallen al capítol 4.
- **Llibreria Camera**: és la llibreria que es fa servir per prendre les fotografies i analizar-les.

També existeixen dues classes addicionalment a la classe *Sensor*. Aquestes classes són la

classe *Actuator*, on s'emmagatzema tota la informació relativa a aquest i la classe *Alarm*, que defineix les alarmes del sistema.

A banda d'aquestes llibreries i classes, el programa dels nodes Client també compta amb un fitxer "*globals.py*" i un arxiu de *logs*. Un diagrama del comportament del codi *Python* del node Client es pot veure a la Figura 3.10.

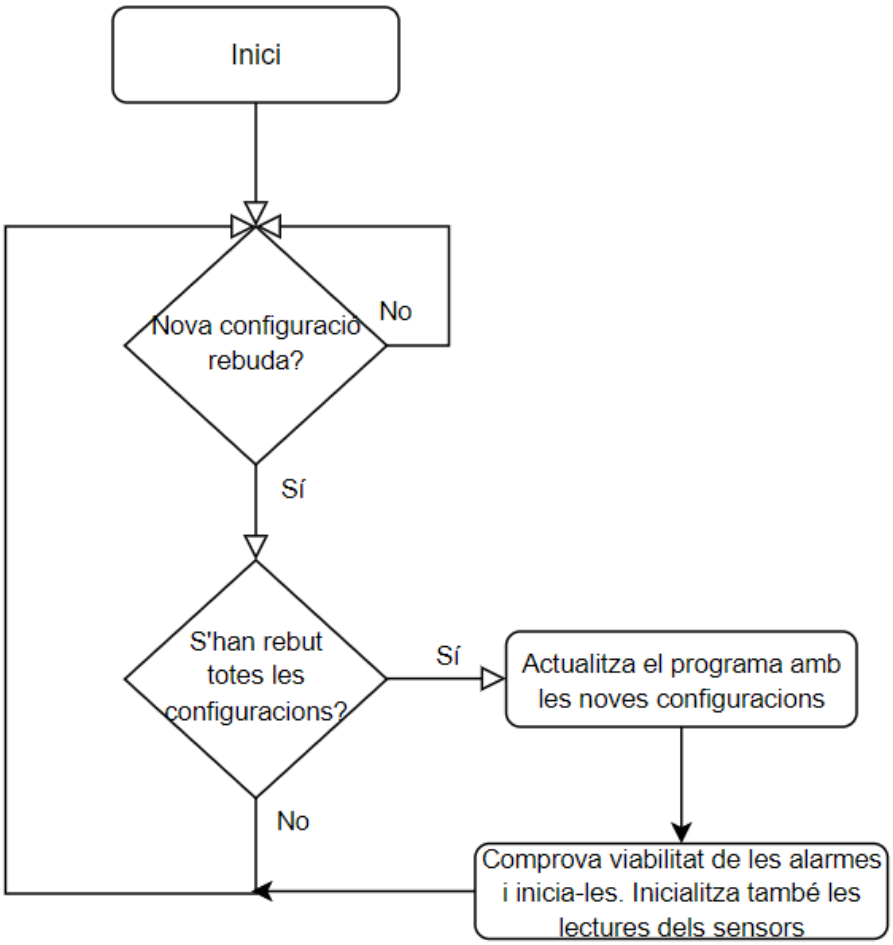

Figure 3.10: Diagrama de flux del node Client

En iniciar-se, revisa si hi ha noves configuracions a la cua de missatges *MQTT*. En cas d'haver-n'hi, cancel·la tots els esdeveniments programats i actualitza les configuracions amb les noves dades rebudes. A continuació, verifica si totes les configuracions necessàries (incloent-hi sensors, actuadors i alarmes) han estat rebudes. Si és així, actualitza les llistes de sensors, actuadors i alarmes segons la nova configuració.

En cas que la configuració d'alarmes hagi estat modificada, es revisa la llista d'alarmes per

determinar quines són viables (aquelles que disposen de tots els sensors o actuadors associats disponibles) i quines no. Després, inicialitza les alarmes viables segons el seu tipus: si són recurrents o si depenen de les lectures dels sensors (Figura 3.11). Finalment, el programa planifica esdeveniments per a comprovar les condicions de les alarmes i dels sensors en els intervals de temps especificats per la seva configuració.

Així, aquest programa permet un control dinàmic sobre un sistema de sensors, actuadors i alarmes, amb la capacitat de rebre actualitzacions de configuració mitjançant missatges *MQTT* i adaptar-se a aquestes actualitzacions en temps real.

El programa del node Client compta amb un planificador de tasques on, un cop executada una tasca, aquesta es torna a programar per ser executada en el futur. Les tasques poden ser de dos tipus (Figura 3.11):

1. Les tasques basades en sensors, implementades per la funció "*sensor_driven_alarm_block*". Aquesta funció llegeix els valors dels sensors, els afegeix a un diccionari global de lectures, i envia aquestes lectures mitjançant *MQTT* al node Servidor. A continuació, programa la seva pròpia execució futura basada en el temporitzador del sensor. Tot seguit aquesta funció comprova les alarmes relacionades amb el sensor que acaba de llegir, i dispara aquelles que es compleixin.

2. Les tasques basades en temporitzadors, implementades per la funció "*timer_driven_alarm_block*". Aquesta funció dispara directament l'alarma quan la tasca és executada. Després, la funció es programa a si mateixa per a una futura execució basada en el temporitzador de l'alarma.

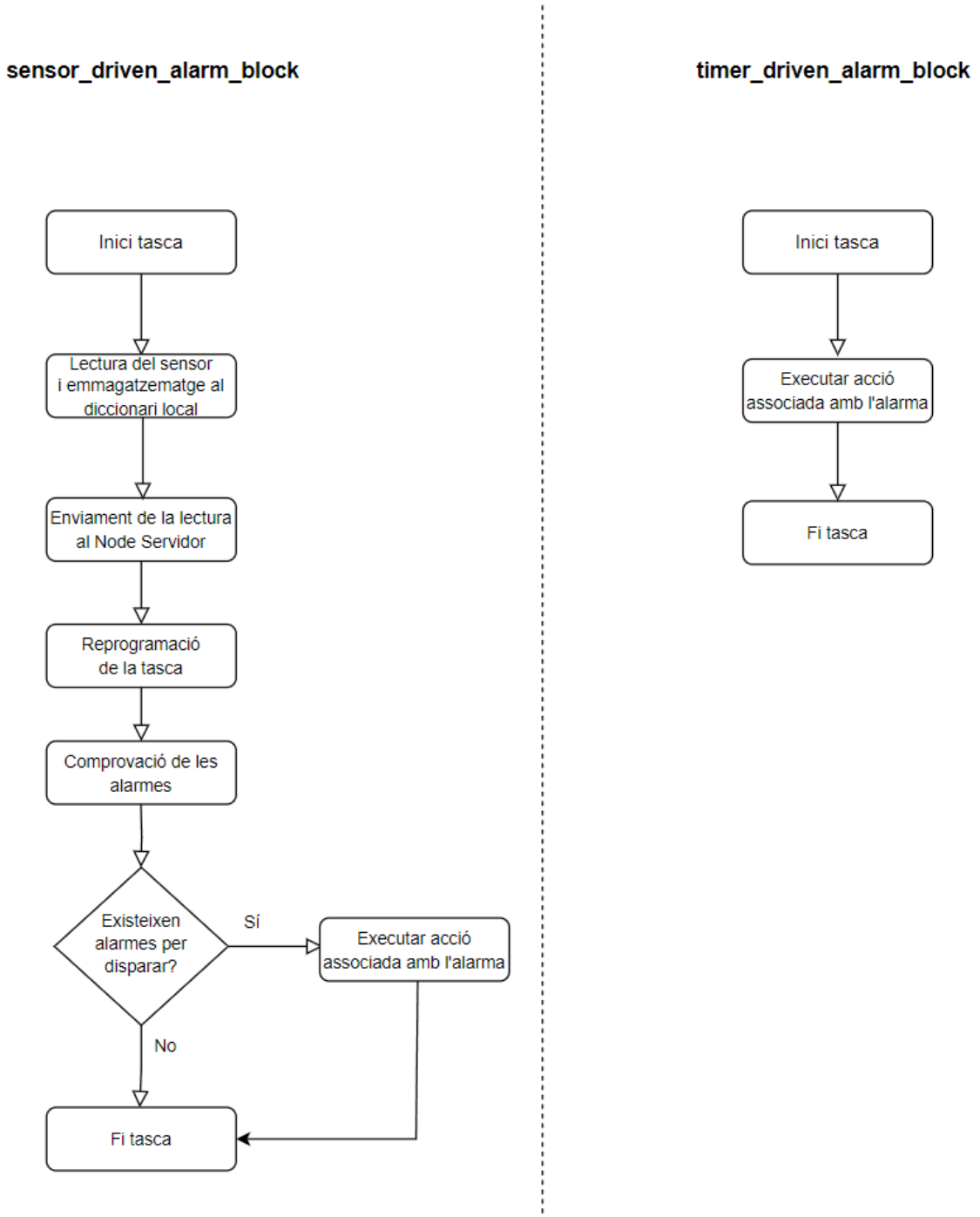

Figure 3.11: Tipus de tasques del node Client

A banda dels canvis realitzats en el programa *Python*, també s'ha modificat el *Stack Software* dels nodes Client.

Tal i com es pot veure a la Figura 3.12, d'igual manera que al *Stack Software* del node Servidor, s'ha eliminat la instància de *Node-RED*.

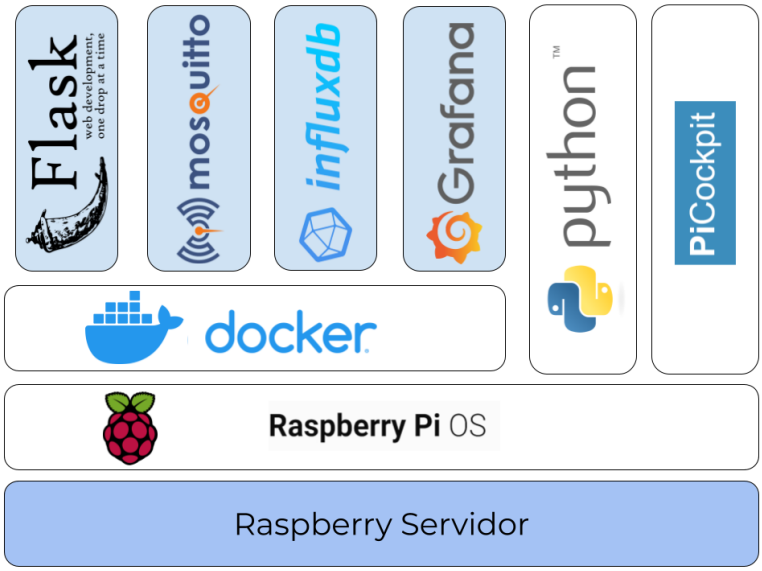

Figure 3.12: Nou *Stack Software* dels nodes Client

Capítol 4

Implementació de les Accions

Aquest capítol se centra en el procés d'implementació de les accions del Sistema d'Alarmes i s'aborda detalladament com s'ha desenvolupat cada acció, des dels seus primers passos amb l'ajuda de programes de prova fins a la seva implementació final dins del programa del node Client.

El primer tema que es tracta és l'activació i la desactivació dels actuadors. En la secció corresponent es discuteix com es controla i es manipula la bomba peristàltica *Kamoer* i com aquesta es pot utilitzar per a satisfer les necessitats del sistema. Després, es tracta l'acció de prendre una fotografia, on s'explica l'acció i el programa específic dissenyat per a realitzar la fotografia i l'anàlisi de la imatge.

Finalment, es mostra la implementació de l'acció d'enviar un correu electrònic. S'exposen les diferents plantilles que s'han creat per a enviar les mesures i es discuteix la possibilitat d'enviar també per correu la imatge capturada i el resum amb l'anàlisi de les imatges.

4.1 Activar i desactivar actuador

La bomba peristàltica *KCM Kamoer*, utilitzada com actuador en aquest projecte per afegir additius a les mescles, pot ser controlada manualment a través dels comandaments que es poden observar a la Figura 4.1. El botó verd és l'encarregat d'engegar i apagar la bomba. El botó vermell permet canviar el sentit de gir d'aquesta i el potenciòmetre permet regular la velocitat de la bomba.

Figure 4.1: Botons de control de la bomba peristàltica

A banda dels controls manuals, també existeix una interfície *Modbus* (*RS-485*) que permet controlar la bomba peristàltica remotament (En el nostre cas, a través de la *Raspberry Pi* Client).

El fabricant proporciona informació suficient per llegir i escriure als registres corresponents per engegar la bomba, aturar-la, canviar la velocitat, etc.

En la Figura 4.2 es pot veure un recull d'aquestes funcions, amb els registres adients i els valors a escriure.

Modbus address composition

Type of data	Modbus address	Read function code	write function code
Digital output (coil)	0x1001-0x1008	0x01	0x05, 0x0f
holding register	0x3001-0x3006	0x03	0x06, 0x10

Digital output (coil) registers

Register address number	Definition (corresponding function)	illustrate
0x1001	start and stop	stop(0)/start(1)
0x1002	empty	close(0)/start(1)
0x1003	direction	Positive (0) / Negative (1)
0x1004	485 control enable bit	enable(1)/disable(0)
0x1005	Analog calibration enable bit	enable(1)/disable(0)
0x1006	Analog single point data saving	save(1)
0x1007	Analog calibration completed	Calibration complete (1)
0x1008	Analog type	20 mA (1) / 5 V (0)

Holding register

Register address number	Definition (corresponding parameter)	read/write
0x3001	Speed (single-precision floating-point number) high 16 bits	R/W
0x3002	Speed (single-precision floating-point number) low 16 bits	R/W
0x3003	Analog (single-precision floating-point) value is 16 bits higher	R/W
0x3004	16 bits of the analog (single-precision floating-point) value	R/W
0x3005	Real-time speed (single-precision floating-point number) high 16 bits	R/W
0x3006	Real-time speed (single-precision floating-point number) low 16 bits	R/W

Figure 4.2: Interfície *Modbus* de la bomba peristàltica

Tal com es pot observar, la bomba es pot engegar o apagar modificant el valor del registre *0x1001*. Per engegar la bomba s'ha d'escriure un *0xFF00* en aquesta direcció a través de la funció 0x05 (Funció Write Coil del protocol *Modbus*). Així i tot, abans de començar a enviar instruccions a la bomba és indispensable activar la comunicació a través de la primera crida *Modbus* de la Figura 4.3(*C0* correspon a l'adreça per defecte de la bomba):

Enable 485 communication (set first after power-on)

Device No (1 Byte)	Function code (1 Byte)	Coil address (2 Byte)	Output value (2 Byte)	CRC check (2 Byte)
C0	05	10 04	FF 00	D9 EA

Start the pump

Device No (1 Byte)	Function code (1 Byte)	Coil address (2 Byte)	Output value (2 Byte)	CRC check (2 Byte)
C0	05	10 01	FF 00	C9 EB

Figure 4.3: Instrucció per activar la comunicació i engegar la bomba

Per tal d'aprofundir en el coneixement les diverses possibilitats que ofereix la bomba, es va desenvolupar un programa de prova que permet controlar-la de manera senzilla (Figura 4.4).

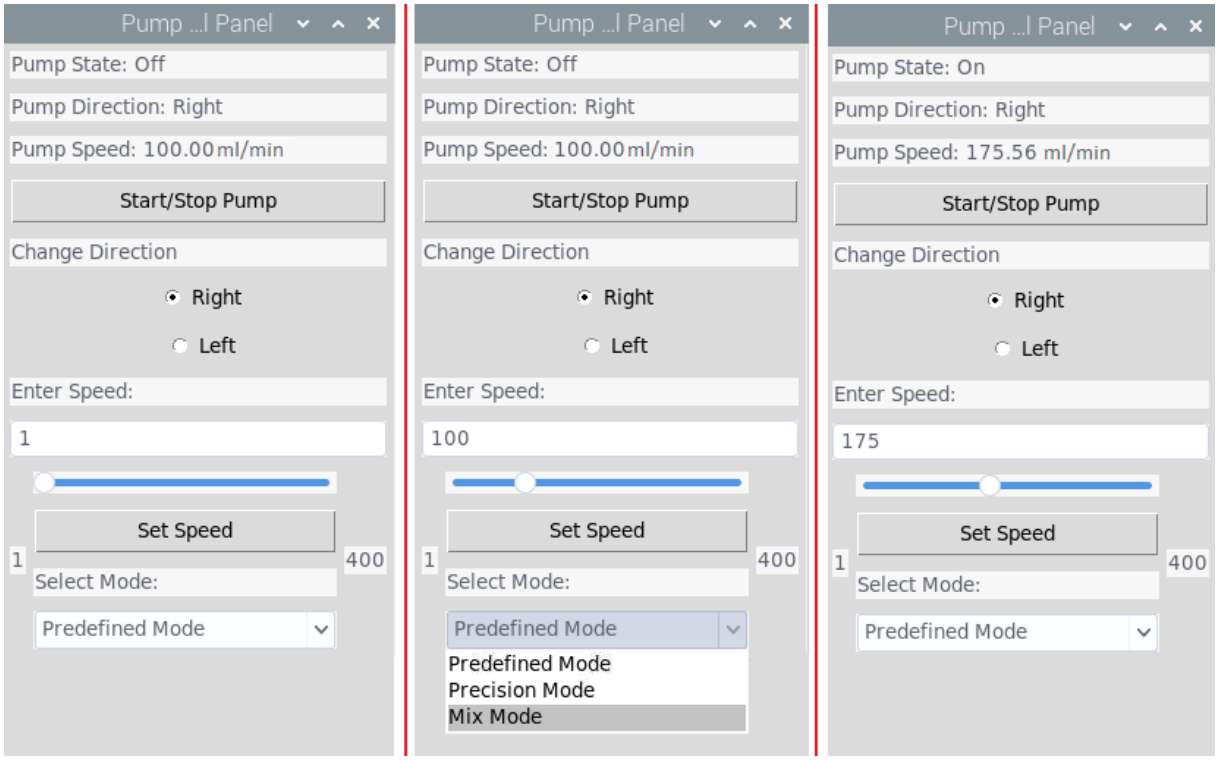

Figure 4.4: Programa de prova de la bomba peristàltica

Aquest programa, dissenyat per facilitar la comprensió del funcionament de la bomba, inclou una interfície gràfica que permet la gestió de la bomba.

Com es pot constatar, aquesta disposa d'una secció que mostra l'estat del motor (estat de la bomba, direcció de rotació, velocitat), a més de diversos elements de control. Aquests inclouen un botó per encendre o apagar el motor, un camp per introduir la velocitat desitjada (hi ha també un lliscador que compleix la mateixa funció), i un menú desplegable per seleccionar el mode de funcionament de la bomba (mode predeterminat, mode mescla i mode precisió).

La imatge central de la Figura 4.4 mostra com es pot canviar el mode de funcionament de la bomba. El mode predeterminat fa que la bomba funcioni de manera contínua fins que es rep una nova instrucció. El mode de precisió permet a l'usuari afegir additius de manera exacta i controlada. El mode mescla alterna la direcció de rotació de la bomba i també activa i desactiva la bomba per generar turbulències que ajuden a barrejar el líquid de la mescla.

Finalment, la tercera imatge de la Figura 4.4 mostra com es pot ajustar la velocitat de la bomba. En la Figura 4.5 es pot veure l'actuador en funcionament.

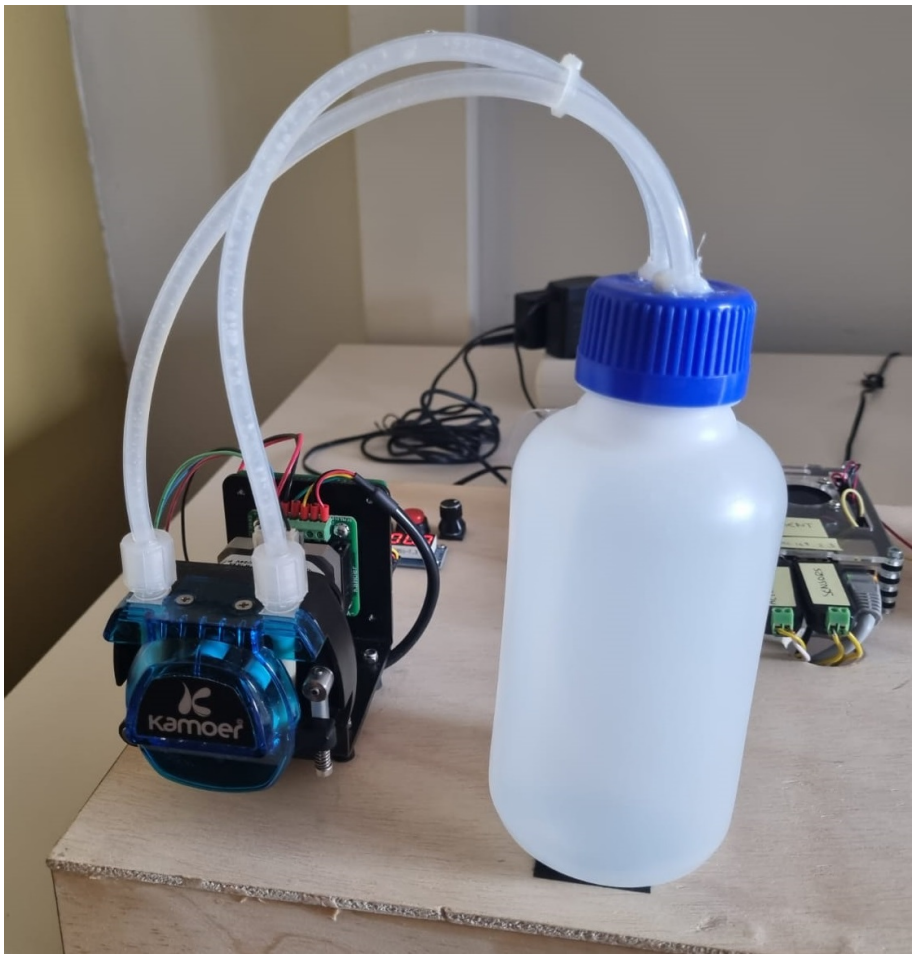

Figure 4.5: Imatge de la bomba en funcionament

Implementació dins del sistema

Tal com s'ha descrit al capítol 3, existeixen dos tipus de tasques: una tasca que dispara les alarmes si el valor del sensor llegit fa complir la condició de l'alarma i una tasca on l'alarma es dispara de manera recurrent. En ambdós casos, quan l'alarma necessita ser disparada es crida a una funció del codi *Python* nomenada "*fire_alarm*".

En aquesta funció, es comprova l'acció associada a l'alarma i depenent de l'acció es realitza un esdeveniment en específic. En aquest cas, si l'alarma té associada l'acció "Activar actuador" o "Desactivar actuador", es crida a la funció "*handle_actuator_action*".

Aquesta funció recorre tots els actuadors associats amb l'alarma i els activa/desactiva a través d'una instrucció *Modbus* que es guarda dins de la classe de l'actuador. Per tal de modul·laritzar el codi i permetre una configuració més còmoda, les instruccions es poden importar des d'un fitxer *JSON* com el de la Figura 4.6

```
{
    "Kamoer": {
        "KCM mini peristaltic pump 38~420ml/min": {
            "Default_Address": "0xC0",
            "Device_ID": "0xC0",
            "Start_Connection_Call":[
                {"Function": "5", "Address": "0x1004", "Value": "0xFF00"}
            ],
            "Start_Pump":[
                {"Function": "5", "Address": "0x1001", "Value": "0xFF00"}
            ],
            "Stop_Pump":[
                {"Function": "5", "Address": "0x1001", "Value": "0x0000"}
            ],
            "Read_Pump_State":[
                {"Function": "5", "Address": "0x1001"}
            ],
            "Set_Pump_Direction": [
                {"Function": "16", "Address": "0x3001", "Values": {"Right": "0x0000", "Left": "0xFF00"}}
            ],
            "Read_Pump_Direction":[
                {"Function": "1", "Address": "0x1003"}
            ]
        }
    }
}
```

Figure 4.6: Arxiu *JSON* per importar les instruccions *Modbus* de cada model d'actuador

4.2 Prendre una fotografia

Tal com es pot veure a la Figura 3.1 del capítol 3, un cop presa una fotografia l'usuari pot escollir si la imatge es guarda localment, si s'analitza o si s'envia per correu. En cas que l'usuari decideixi analitzar-la, es realitza la correlació de la imatge i l'informe resultant pot ser guardat localment o enviat per correu.

Abans d'implementar aquest seguit d'accions al programa principal, es va crear un programa per tal d'entendre el funcionament del mòdul de la càmera i les diferents possibilitats de configuració. La interfície gràfica d'aquest programa, es pot veure a la Figura 4.7.

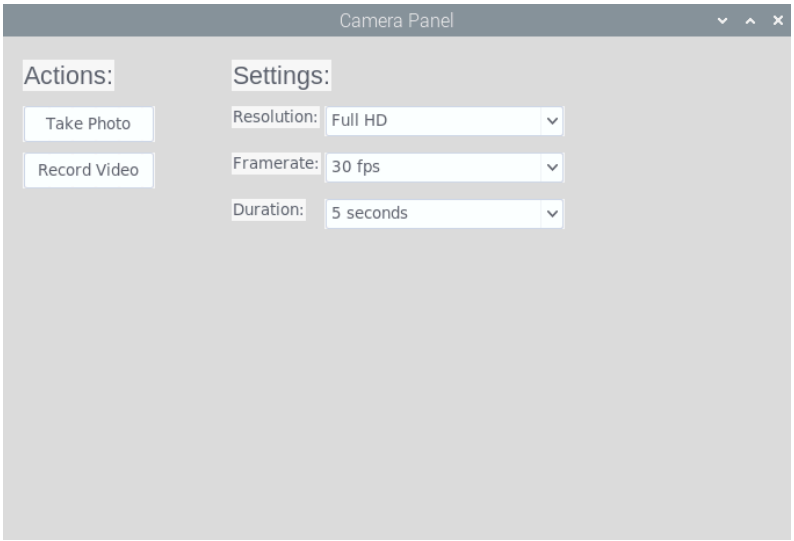

Figure 4.7: Ínterficie gràfica del programa de la càmera

La interfície permet a l'usuari o prendre una fotografia o enregistrar un vídeo. A banda de les accions, l'usuari pot escollir la resolució (*HD* o *FullHD*), el *framerate* (30 o 60 *fps*) i la duració del vídeo (5, 10 o 15 segons). Les diferents opcions es poden veure a la següent figura:

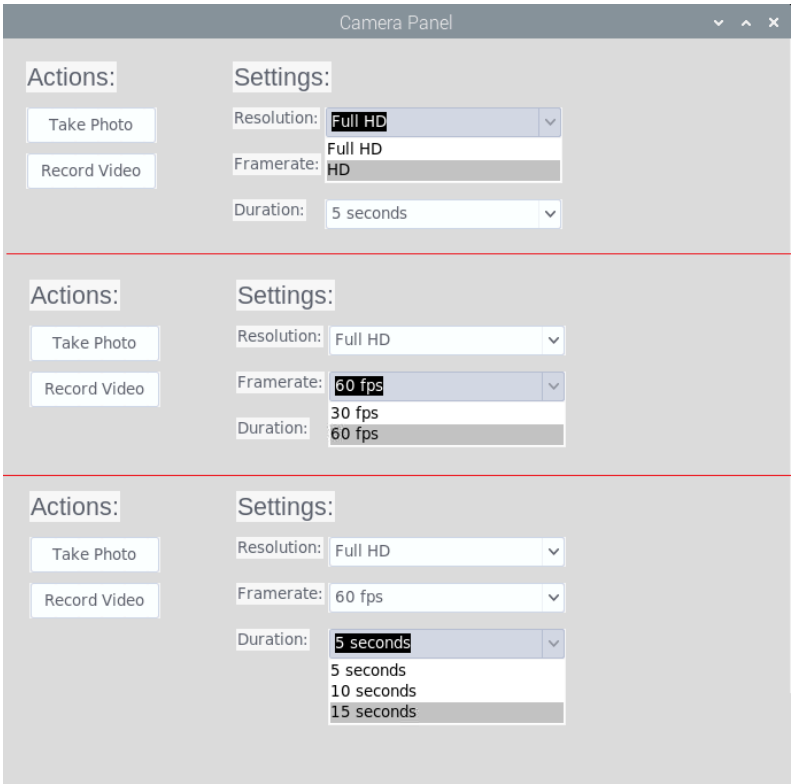

Figure 4.8: Les diferents opcions de configuració del programa de la càmera

Un cop presa la fotografia o enregistrat el vídeo, una vista prèvia de l'arxiu es pot veure per pantalla:

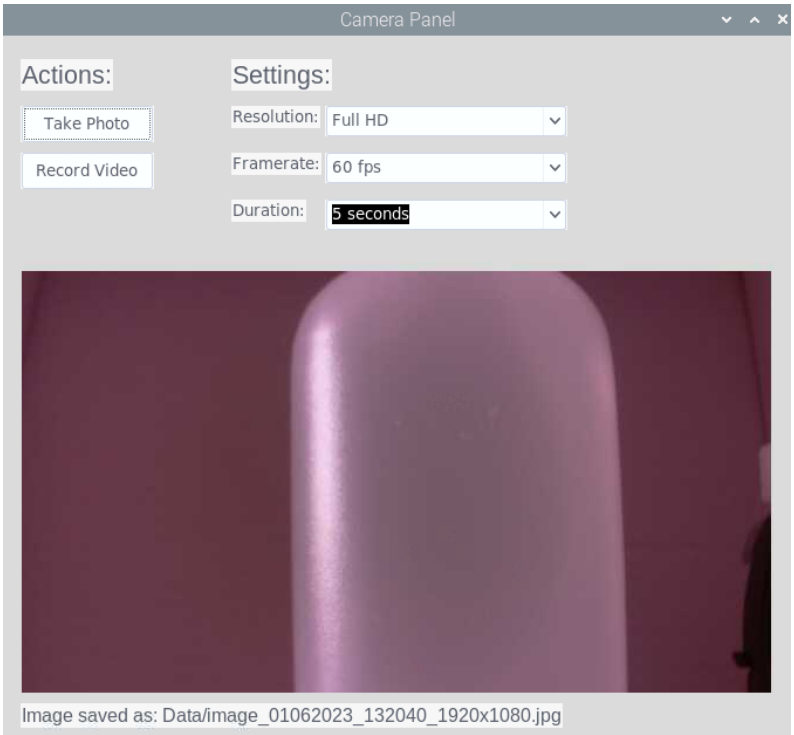

Figure 4.9: Interfície gràfica amb una imatge presa

La imatge per defecte s'emmagatzema en una carpeta nomenada "*Data*" amb un nom compost per la data del dia que va ser presa i característiques com la resolució, el *framerate* o la duració.

4.2.1 Fer correlació de la mescla

Tal com s'indica al capítol 3, una de les possibles accions a realitzar després de prendre una fotografia és analitzar-la per extreure'n informació.

Per aquest motiu, s'ha investigat diferents algorismes per tal de poder monitorar el comportament de les mescles i un dels menys exigents computacionalment és realitzar la correlació de la imatge presa amb un conjunt d'imatges de prova. A fi d'implementar aquest algorisme, es va crear un programa que té com a objectiu analitzar i comparar imatges per trobar la més semblant dins un conjunt predefinit d'estats (Figura 4.10).

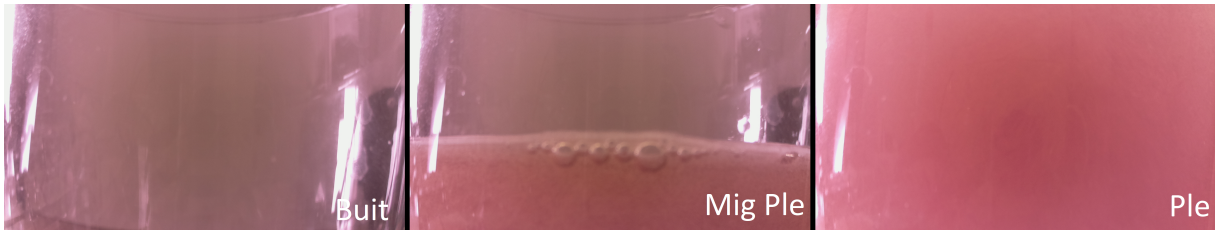

Figure 4.10: Possibles estats de la mescla

Tal com es pot veure, els estats predefinits són:

- Recipient buit.
- Recipient mig ple.
- Recipient ple.

Per tal de decidir l'estat predefinit més pròxim es converteixen totes les imatges en matrius en escala de grisos, ja que facilita el processament i la comparació. Una vegada convertides, es calcula el coeficient de correlació de Pearson ([20]) per a cada parella d'imatges, que mesura el grau de relació lineal entre les intensitats dels píxels de les dues imatges.

El programa també presenta els resultats de manera visual, amb gràfics que mostren tant la imatge capturada com l'estat d'imatge més semblant. A més a més, es mostra una gràfica de dispersió que representa la correlació entre les intensitats dels píxels de les dues imatges. Aquesta metodologia presenta un gran avantatge en termes de velocitat de processament i baixa demanda computacional, que la fa idònia per a aplicacions en temps real o en situacions on els recursos de càlcul són limitats. A continuació es mostra 3 informes generats pel programa:

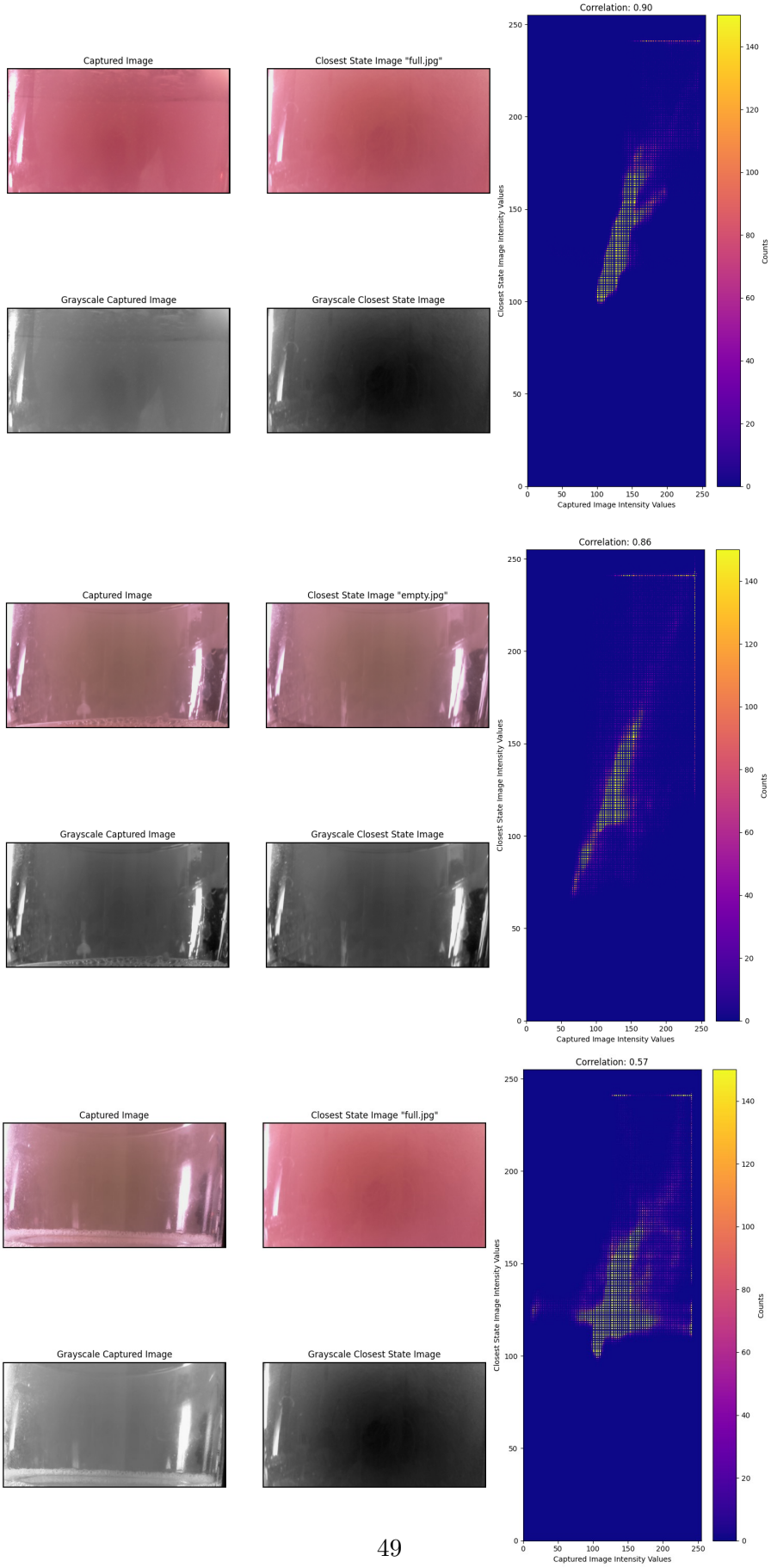

Figure 4.11: Informes generats pel programa

Com es pot observar, el càlcul de la correlació de Pearson pot ser útil per determinar l'estat de les mescles. No obstant això, l'algorisme no és infal·lible i en certes circumstàncies l'estat detectat per l'algorisme pot diferir de l'estat real capturat amb la càmera.

Encara que l'algorisme prediu correctament la primera i la segona imatge de la Figura 4.11, la tercera predicció és totalment errònia. Després d'analitzar les fotografies, s'ha arribat a la conclusió que per millorar els resultats obtinguts, el líquid ha de presentar una coloració que contrasti amb el fons de la fotografia (en el nostre cas, els dos colors són força propers) i les condicions de llum han de romandre constants. També s'ha determinat que només s'ha de considerar el resultat si la correlació és igual o superior a 0,8.

Implementació dins del sistema

En el cas de "Prendre una fotografia", la funció "*fire_alarm*" crida a la funció auxiliar "*handle_photo_action*".

En aquesta funció, es comprova que la càmera estigui disponible i llesta per prendre una fotografia. Un cop presa, es crida a la funció "*perform_postaction*" on, segons l'acció consecutiva associada amb l'alarma, es realitza una de les tres accions següents:

- S'emmagatzema la imatge capturada localment.
- S'envia la imatge per correu electrònic (acció descrita en la subsecció 4.3).
- Es fa la correlació de la fotografia per determinar l'estat predefinit més pròxim. Un cop efectuat l'anàlisi, es crida a la funció "*perform_consecutive_action*" per tal de determinar si l'informe creat s'emmagatzema localment o s'envia per correu.

4.3 Enviar un correu electrònic

La darrera acció del Sistema d'Alarmes a descriure és enviar un correu electrònic amb les lectures d'un sensor de la planta. A banda d'aquesta acció, en aquesta secció també es mostra com s'han desenvolupat i els resultats obtinguts de les accions complementàries de l'acció "Prendre una fotografia" que impliquen l'enviament de la imatge o de l'informe per correu.

D'igual manera que amb les accions anteriors, abans d'implementar aquest seguit d'accions al programa principal es va crear un programa per aprendre a enviar correus electrònics de tota mena a través de *Python*.

El programa creat es va centrar en dissenyar una plantilla per enviar la lectura d'un sensor *HTML* que inclogués la imatge corporativa de *Chemplate*, juntament amb informació de l'empresa. La plantilla, que canvia en funció del client, el sensor, el valor de la lectura, etc. resulta en un correu com el de la Figura 4.12.

Figure 4.12: Correu electrònic de prova amb informació d'un sensor fictici

Implementació dins del sistema

En el cas d'"Enviar un correu electrònic", la funció "*fire_alarm*" crida a la funció auxiliar "*handle_email_action*".

Aquesta funció comprova que el sensor associat amb l'alarma existeixi i, una vegada recuperades les últimes lectures del sensor, les envia a través de dues funcions de la llibreria "Email". Si l'alarma només ha d'enviar una lectura del sensor, el programa crida a la funció "*send_email_with_reading*". Si el nombre de lectures és superior, es crida a la funció "*send_email_with_N_readings*".

Aquestes dues funcions obtenen un fitxer de plantilla per al missatge de correu electrònic (totes les plantilles estan emmagatzemades en una carpeta nomenada "*templates*").

Un cop obtinguda la plantilla, la funció substitueix algunes paraules de la plantilla pels valors reals que se li han donat com a paràmetres a la funció (el nom del sensor, el tipus de sensor, la unitat, el valor i el nom de l'alarma).

Finalment, aquest missatge formatat s'afegeix al correu electrònic i s'envia fent servir una altra funció anomenada "*send_email*". Aquesta darrera funció és la que realment es connecta a Internet i envia el correu electrònic.

Ambdues funcions comparteixen una estructura similar. No obstant això, cada funció compta amb la seva pròpia plantilla *HTML*. En la Figura 4.13 es recull un exemple d'un correu amb una sola lectura i un correu amb múltiples valors.

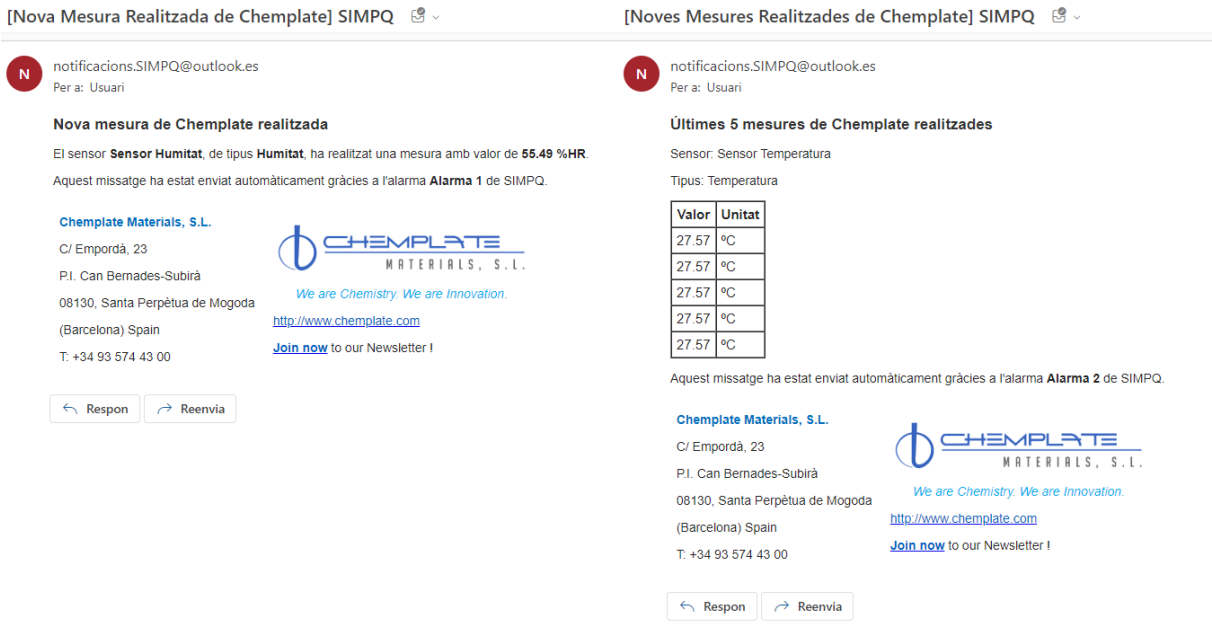

Figure 4.13: Correu electrònic amb una (esquerra) i múltiples lectures (dreta)

El fitxer "*Email*" també compta amb una funció que permet adjuntar una imatge al correu. Aquesta funció, nomenada "*send_email_with_attachment*", és la utilitzada per enviar les imatges capturades i els informes amb l'anàlisi de les imatges.

Els correus resultants d'aquestes plantilles es mostren a la Figura 4.14.

Figure 4.14: Correu electrònic amb una imatge (esquerra) i un informe adjunt (dreta).

Capítol 5

Interfície Web

En aquest capítol es presenta el desenvolupament d'una nova versió de la pàgina web de SIMPQ, la qual ha estat redissenyada i millorada per a superar totes les limitacions de la versió original. Tal com s'ha comentat anteriorment, la primera versió de la pàgina només permetia tenir un client configurat i no comptava amb una infraestructura que permetés múltiples usuaris. Aquesta limitació va limitar l'escalabilitat i la flexibilitat de la plataforma.

Amb la nova versió de la pàgina web, s'ha realitzat una important reestructuració i ampliació de les funcionalitats per tal de satisfer les necessitats de l'empresa i proporcionar una millor experiència d'usuari.

La nova versió és multiclient i permet als operaris de l'empresa, cadascun amb el seu compte d'usuari, accedir als panells de control dels clients que desitgin en funció del seu rang. Això significa que es pot gestionar i monitorar diversos clients i les seves respectives plantes pilot des d'un únic punt d'accés.

A més de l'ampliació de la funcionalitat multiclient, la pàgina web també ha sigut redissenyada per tal de millorar significativament la seva aparença i usabilitat. S'ha aplicat un enfocament modern i atractiu al disseny de la interfície, utilitzant elements visuals atractius, paletes de colors harmòniques i tipografies elegants. Aquest redisseny té com a objectiu proporcionar una experiència d'usuari intuïtiva i amigable, facilitant la navegació i la interacció amb les diferents funcionalitats de la plataforma. A continuació es fa un recorregut per aquesta nova pàgina, on també s'ha integrat el Sistema d'Alarmes donant als usuaris la possibilitat de configurar actuadors i alarmes des del panell del client.

5.1 Inici de Sessió i Selecció de Client

Per tal d'accedir al portal web, l'usuari ha d'emplenar el formulari de la Figura 5.1 amb les seves credencials d'usuari.

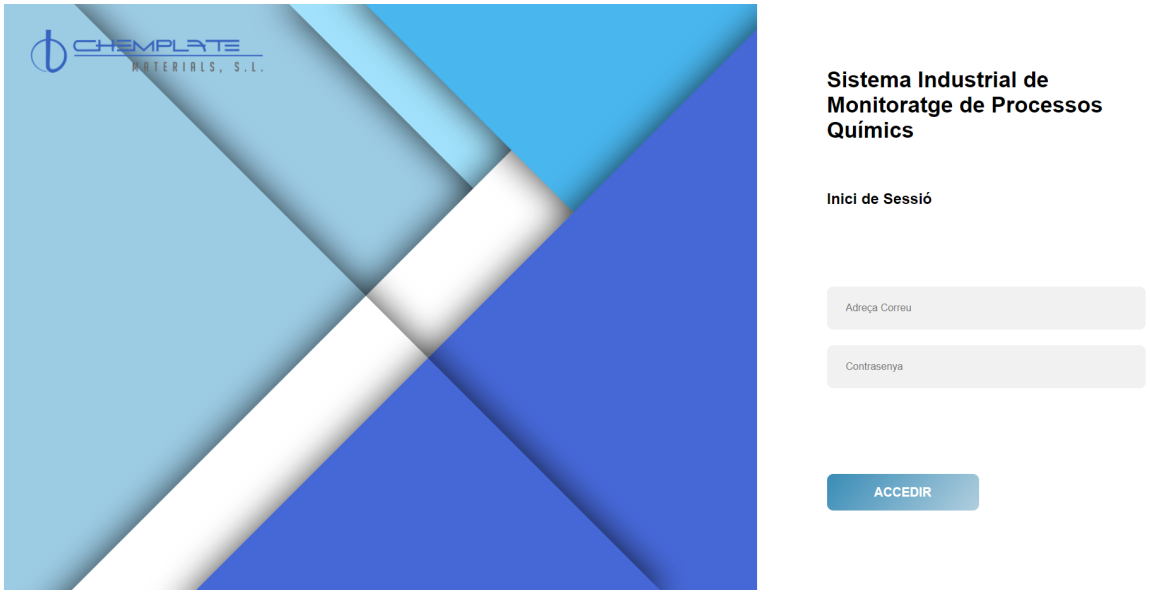

Figure 5.1: Pàgina d'inici de sessió

Després d'introduir les credencials (correu electrònic i contrasenya), la pàgina redirigeix a l'usuari al selector de clients. En el cas de la Figura 5.2, el client seleccionat és "*Chemplate*".

Figure 5.2: Selector de clients

Un cop seleccionat el client, l'usuari és redirigit al panell de control corresponent (Figura 5.3).

5.2 Panell de Control del Client

El panell de control brinda diverses funcionalitats perquè els clients puguin gestionar la seva planta pilot. A continuació es detallen les accions disponibles:

- **Lectura dels sensors**: aquest enllaç redirigeix l'usuari al panell de visualització de *Grafana*.
- **Base de dades**: és un enllaç a la base de dades *InfluxDB* que conté les lectures dels sensors del client.
- **Estat del Sistema**: redirigeix l'usuari al servei de monitoratge *PiCockpit* per a visualitzar l'estat actual del sistema.
- **Configuració dels sensors**: permet a l'usuari definir els sensors que s'utilitzen a la seva planta pilot.
- **Configuració dels actuadors**: permet a l'usuari concretar la configuració dels actuadors del seu sistema.
- **Configuració de les alarmes**: permet a l'usuari definir i configurar les alarmes per a controlar les mesures de la planta pilot.

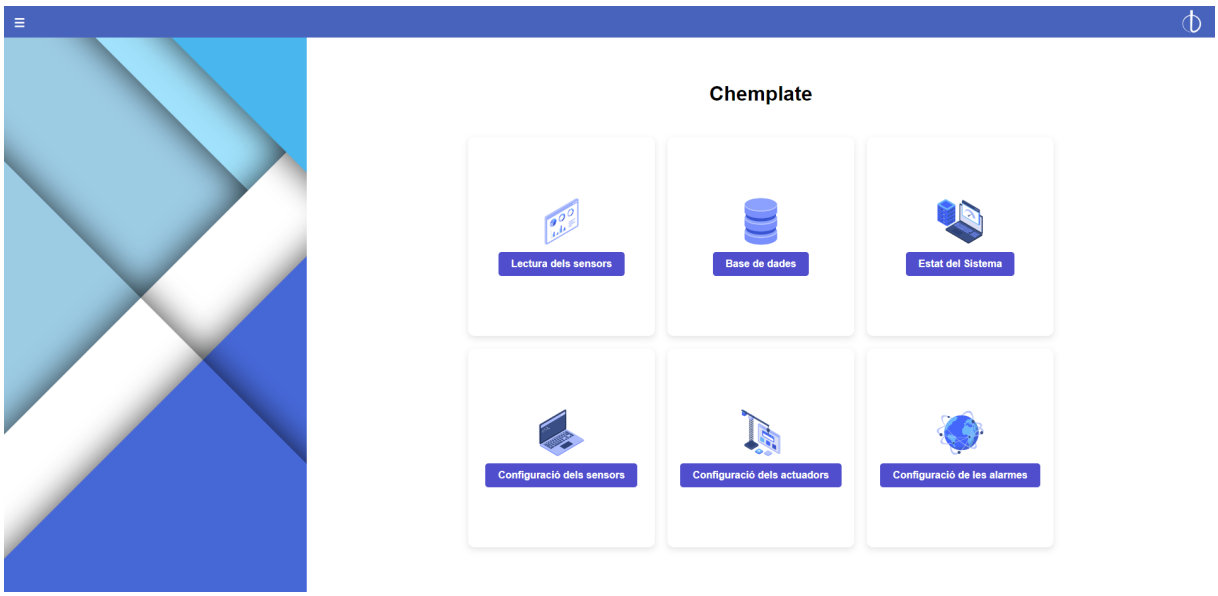

Figure 5.3: Panell de control del client

5.3 Configuració dels Sensors

En seleccionar l'opció "Configuració dels sensors", l'usuari és redirigit al formulari de la Figura 5.4, on pot introduir la informació relativa als sensors de la seva planta pilot.

Figure 5.4: Formulari de configuració dels sensors sense cap sensor configurat

Aquest formulari, similar al formulari de sensors de SIMPQ, presenta els següents camps:

- **Nom del Sensor**: Camp de text per introduir el nom del sensor. És un camp obligatori i té una longitud màxima de 30 caràcters.
- **Identificador**: Menú desplegable per seleccionar un identificador per al sensor. És obligatori i ha de ser únic. Els identificadors dels sensors són números de l'1 al 4. El valor 4 és un límit que es pot modificar adaptant-se a les necessitats del client.
- **Tipus de Sensor**: Menú desplegable per seleccionar el tipus de sensor. Està directament relacionat amb les unitats disponibles. Els tipus de sensors disponibles són: pH, conductivitat, ORP, oxigen dissolt, temperatura i humitat.
- **Unitat**: Menú desplegable per seleccionar la unitat de mesura del sensor. Les unitats corresponents als diferents tipus de sensors són: pH, µS/cm, mV, g/L, °C i %HR.
- **Interval de Temps**: Menú desplegable per seleccionar l'interval de temps de lectura del sensor. És un camp obligatori i les opcions disponibles són: cada 5 segons, cada 10 segons, cada 30 segons i cada minut.
- **Funció *Modbus* Lectura**: Menú desplegable per seleccionar la funció de lectura *Modbus* del sensor. Aquesta opció està relacionada amb les adreces disponibles.
- **Adreça**: Menú desplegable per seleccionar l'adreça *Modbus* del sensor. És un camp obligatori i les opcions disponibles són les adreces *Modbus* que s'utilitzen per als sensors de les plantes pilot dels clients de *Chemplate*.
- **Nº Registres**: Camp numèric per introduir el nombre de registres del sensor. Ha de ser un valor entre 1 i 10.

- **Comentaris Addicionals**: Camp de text per introduir comentaris addicionals o observacions sobre el sensor. Té una longitud màxima de 150 caràcters.

Tots els camps tret del camp "Comentaris Addicionals" són obligatoris. En la Figura 5.5 es poden veure 4 sensors: un de pH, un de Conductivitat, un d'ORP i un d'Oxigen Dissolt.

Figure 5.5: Pàgina per configurar els sensors amb 4 sensors configurats

5.4 Configuració dels Actuadors

En seleccionar l'opció "Configuració dels actuadors", l'usuari és redirigit al formulari de la Figura 5.6, on pot introduir la informació relativa als actuadors utilitzats a la planta pilot.

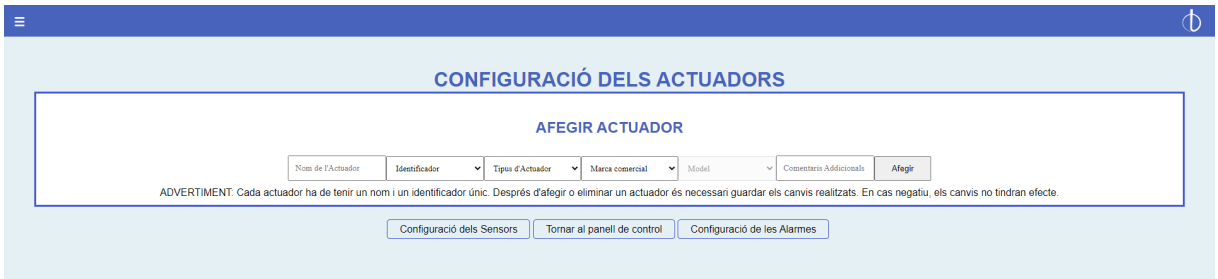

Figure 5.6: Formulari de configuració dels actuadors sense cap actuador configurat

Aquest formulari presenta els següents camps:

- **Nom de l'Actuador**: Camp de text per introduir el nom de l'actuador. És un camp obligatori i té una longitud màxima de 30 caràcters.

- **Identificador**: Menú desplegable per seleccionar un identificador per a l'actuador. És obligatori i ha de ser únic. Els identificadors dels actuadors també són números de l'1 al 4. El valor 4 és un límit que es pot modificar adaptant-se a les necessitats del client.
- **Tipus d'Actuador**: Menú desplegable per seleccionar el tipus d'actuador.
- **Marca comercial**: Menú desplegable per seleccionar la marca comercial de l'actuador. Aquesta selecció està relacionada amb els models disponibles per a aquesta marca.
- **Model**: Menú desplegable per seleccionar el model de l'actuador. És obligatori i depèn de la marca comercial seleccionada.
- **Comentaris Addicionals**: Camp de text per introduir comentaris addicionals sobre l'actuador. Té una longitud màxima de 150 caràcters.

Tots els camps tret del camp "Comentaris Addicionals" són obligatoris. En la Figura 5.7 es poden veure 2 actuadors, ambdós bombes peristàltiques.

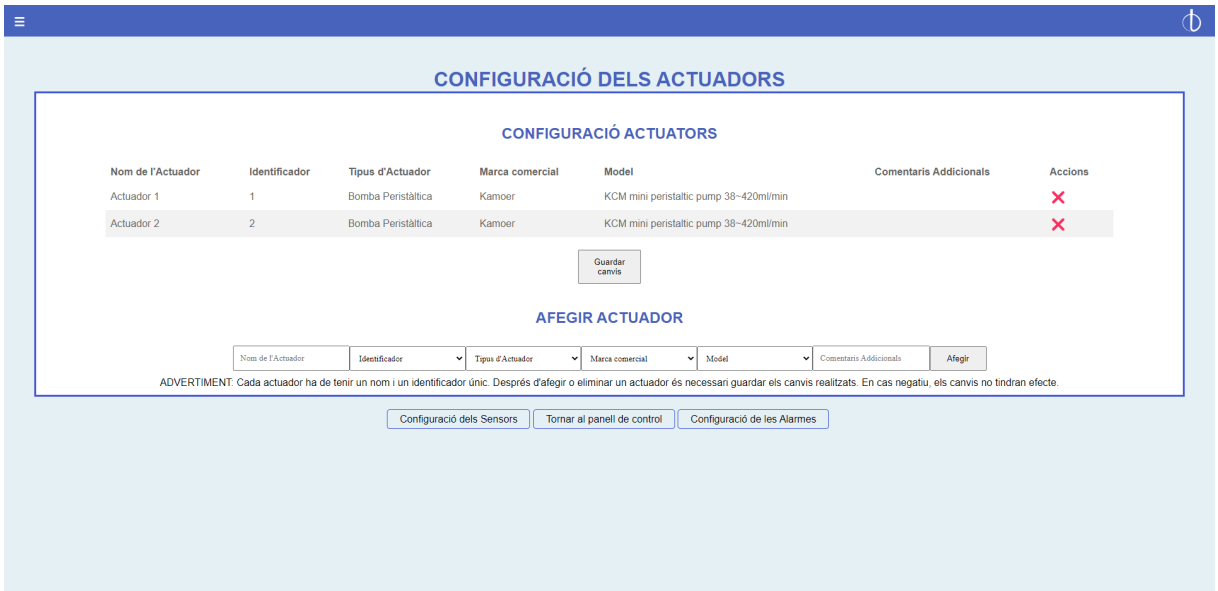

Figure 5.7: Pàgina per configurar els actuadors amb 2 actuadors configurats

5.5 Configuració de les Alarmes

En seleccionar l'opció "Configuració de les alarmes", l'usuari accedeix al formulari de la Figura 5.8, on pot introduir la informació relativa a les alarmes del sistema.

Figure 5.8: Formulari de configuració de les alarmes sense cap alarma configurada

Aquest formulari presenta un comportament dinàmic on, en funció de l'acció a realitzar per l'alarma i el desencadenant d'aquesta es mostren i s'oculten diferents camps. El formulari de les alarmes pot presentar alguns dels següents camps:

- **Nom de l'Alarma**: Camp de text per introduir el nom de l'alarma. És un camp obligatori i té una longitud màxima de 30 caràcters.
- **Identificador**: Menú desplegable per seleccionar un identificador únic per a l'alarma. És un camp obligatori. Els identificadors de les alarmes són números de l'1 al 5. Aquest límit també es pot modificar.
- **Acció**: Menú desplegable per seleccionar l'acció associada a l'alarma.
- **Actuador Associat**: Menú desplegable per seleccionar l'actuador o actuadors associats amb l'alarma. Aquest camp només es mostra si s'ha seleccionat una acció relacionada amb un actuador.
- **Mode de Funcionament**: Menú desplegable per seleccionar el mode de funcionament de l'actuador o actuadors associats amb l'alarma. Aquest camp només es mostra si l'acció és "Activar un actuador" i les opcions disponibles defineixen el comportament dels actuadors: "Mode Predeterminat", "Mode Precisió" o "Mode Mescla".
- **Acció Posterior**: Menú desplegable per seleccionar l'acció posterior associada a l'alarma. Aquest camp només es mostra si l'acció seleccionada és "Prendre una fotografia".
- **Acció Consecutiva**: Menú desplegable per seleccionar l'acció consecutiva associada a l'alarma. Aquest camp només es mostra en certs casos, com quan s'ha seleccionat l'acció posterior "Fer correlació de la mescla" i pot incloure opcions com "Guardar informe correlació localment" o "Enviar informe per correu electrònic".
- **Sensor a Llegir**: Menú desplegable per seleccionar el sensor associat a l'alarma en el cas de l'acció posterior "Enviar per correu electrònic". Aquest camp només es mostra si s'ha seleccionat aquesta opció.

- **Nº Mostres**: Camp numèric per introduir el nombre de mostres relacionades amb l'acció posterior "Enviar per correu electrònic". Aquest camp només es mostra si s'ha seleccionat aquesta opció i permet un valor entre 1 i 10.
- **Desencadenant**: Menú desplegable per seleccionar el tipus de desencadenant de l'alarma. Les opcions inclouen "Alarma recurrent" i "Quan es compleixi la condició".
- **Interval Comprovació (seg)**: Camp numèric per introduir l'interval de comprovació de l'alarma en segons. Aquest camp només es mostra si s'ha seleccionat l'opció de desencadenant "Quan es compleixi la condició" i permet valors entre 5 i 600 segons.
- **Sensor Associat**: Menú desplegable per seleccionar el sensor associat a l'alarma en el cas de desencadenant "Quan es compleixi la condició". Aquest camp només es mostra si s'ha seleccionat aquesta opció.
- **Operació**: Menú desplegable per seleccionar l'operació de comparació per tal de definir la condició a complir. Les opcions inclouen "Major que", "Menor que" i "Igual a". Aquest camp només es mostra si s'ha seleccionat l'opció de desencadenant "Quan es compleixi la condició"
- **Valor Llindar**: Camp de text per introduir el valor llindar de la condició. Aquest camp només es mostra si s'ha seleccionat l'opció de desencadenant "Quan es compleixi la condició".
- **Nº Cicles**: Camp numèric per introduir el nombre de cicles relacionats amb l'alarma. Aquest camp només es mostra si s'ha seleccionat l'opció de desencadenant "Quan es compleixi la condició" i permet valors entre 1 i 50.
- **Comentaris Addicionals**: Camp de text per introduir comentaris addicionals sobre l'alarma. Té una longitud màxima de 150 caràcters i és un camp opcional.

En les següents Figures (5.9 i 5.10) es troba descrit el procés per tal d'afegir fins a 5 alarmes.

Figure 5.9: Pàgina per configurar les alarmes on el formulari està sent emplenat

Figure 5.10: Pàgina per configurar les alarmes amb 4 alarmes configurades

5.6 Funcionalitats Addicionals

A banda dels apartats mencionats anteriorment, la pàgina web també compta amb seccions que milloren l'experiència d'usuari i la navegació d'aquest per la pàgina.

La barra lateral que es mostra a la Figura 5.11 proporciona informació sobre l'usuari i conté diversos enllaços per redirigir-lo a diferents parts de la pàgina. Aquesta barra lateral és

un component de navegació que permet a l'usuari accedir de manera ràpida i fàcil a diferents seccions importants del lloc web. Els enllaços de la barra lateral són:

- **Inici**: Aquest enllaç redirigeix a l'usuari a la pàgina principal.
- **Selector de Clients**: Aquest enllaç redirigeix l'usuari a una pàgina on pot seleccionar o canviar el client al qual està associat.
- **Perfil d'Usuari**: Aquest enllaç permet a l'usuari accedir a la pàgina de configuració del seu perfil, on pot visualitzar i modificar la seva informació personal com el nom d'usuari, l'adreça de correu electrònic, la foto de perfil, etc.
- **Tancar Sessió**: Aquest enllaç permet a l'usuari tancar la seva sessió actual i sortir del compte.

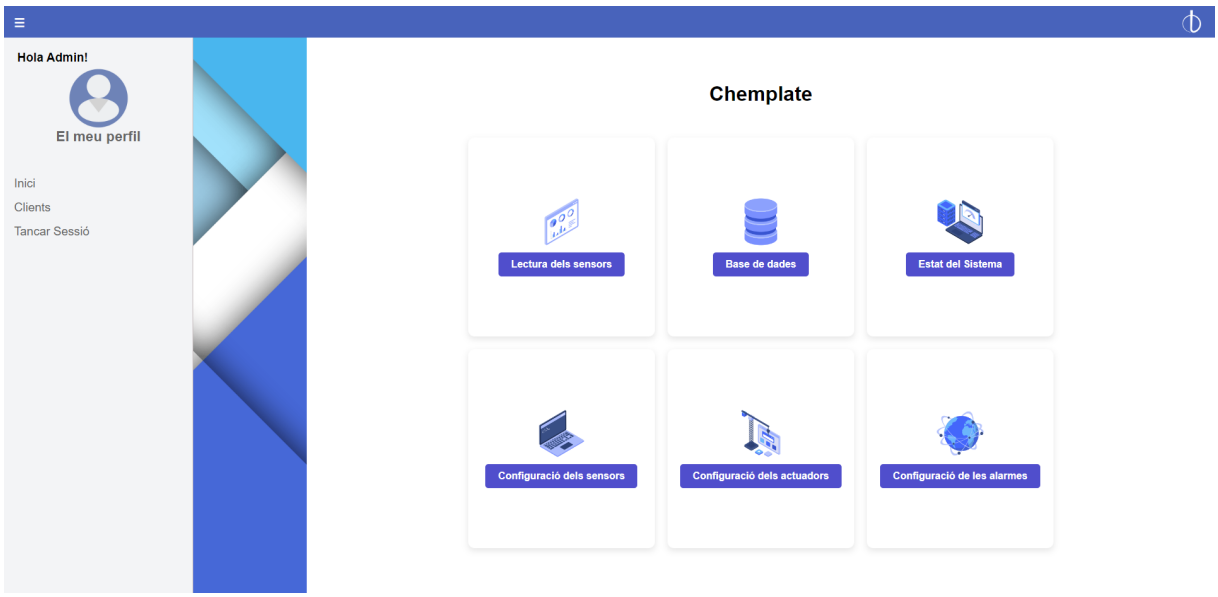

Figure 5.11: Barra de navegació lateral

La Figura 5.12 mostra la pàgina on l'usuari pot modificar la seva informació personal, com el nom d'usuari, l'adreça de correu electrònic i la contrasenya. Aquesta pàgina és útil perquè els usuaris puguin actualitzar les seves dades personals de manera senzilla i còmoda. L'usuari pot introduir les noves dades als camps corresponents i, generalment, haurà de validar la informació abans de desar els canvis.

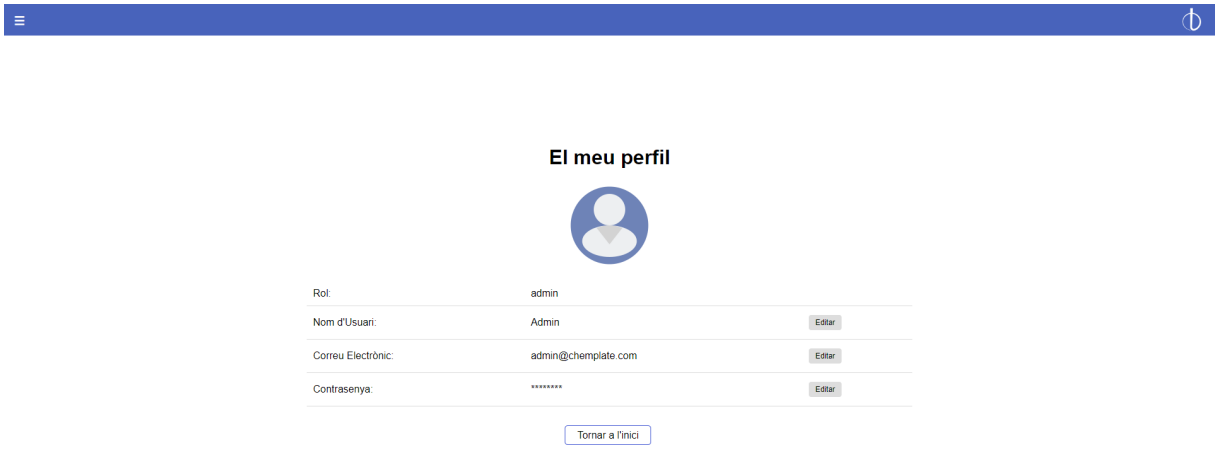

Figure 5.12: Panell lateral amb informació de l'usuari

La Figura 5.13 mostra dues pantalles diferents. En primer lloc, ens presenta la pàgina de tancament de sessió, que s'activa després que l'usuari hagi tancat la seva sessió. En segon lloc, mostra la pàgina d'errors 404, que s'activa quan un usuari intenta accedir a una pàgina que no existeix. Aquesta pàgina és útil per informar a l'usuari que la pàgina sol·licitada no es pot trobar. En el nostre cas, la pàgina indica a l'usuari que la pàgina no existeix i l'invita a tornar a la pàgina d'inici.

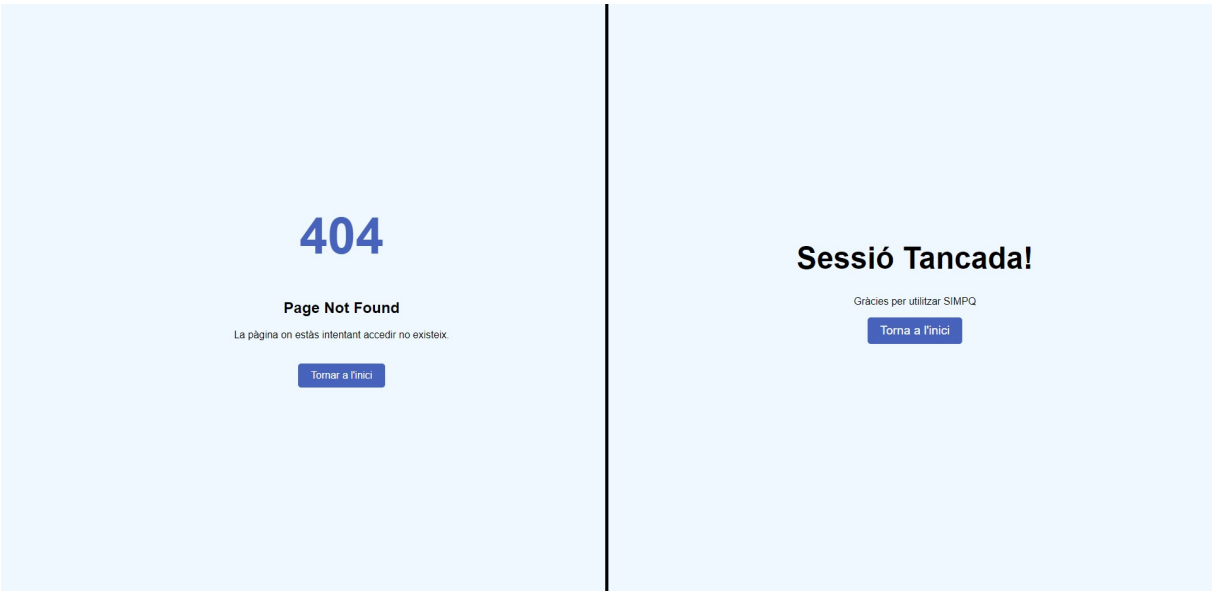

Figure 5.13: Pàgina de tancament de sessió

5.7 Tecnologies utilitzades

Per al desenvolupament de la nova versió de la pàgina web, s'han utilitzat tecnologies que ja s'havien emprat durant la primera versió de la pàgina juntament amb algunes de noves. A continuació, es presenten aquestes tecnologies i el seu propòsit.

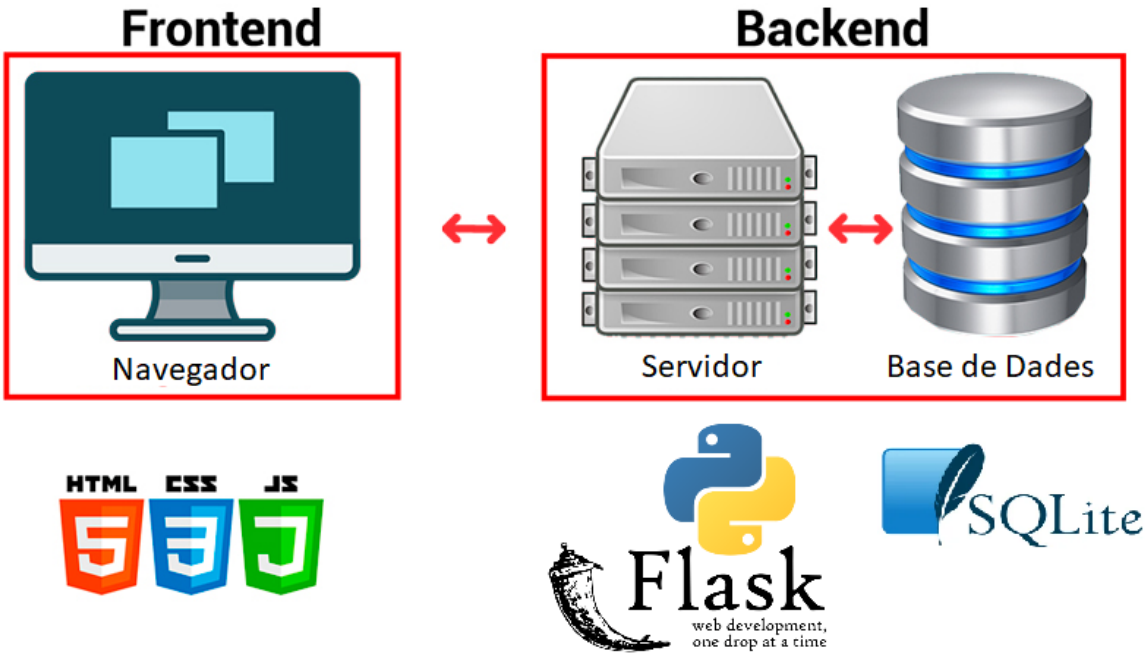

Figure 5.14: Stack Web

- ***HTML (HyperText Markup Language)***: és l'utilitzat per crear l'estructura i el contingut d'una pàgina web. Defineix els elements i l'organització del contingut, com ara encapçalaments, paràgrafs, llistes, enllaços, imatges, taules, formularis, entre d'altres.

- ***JavaScript***: és el que permet afegir interactivitat i funcionalitat dinàmica a les pàgines web. Permet també la manipulació del contingut i la interacció amb l'usuari. S'utilitza per realitzar accions en resposta a esdeveniments, validar formularis, crear efectes visuals, comunicar-se amb el servidor i moltes altres funcionalitats interactives. Una breu descripció del codi *Javascript* de la pàgina es pot consultar a l'Apèndix A.1.

- ***CSS (Cascading Style Sheets)***: és el que s'utilitza per controlar l'aparença i el disseny d'una pàgina web. Permet definir estils pels elements *HTML*, com ara colors, fonts, marges, mides, posició i animacions. Ajuda a separar la presentació del contingut, la qual cosa facilita el manteniment i la coherència visual d'un lloc web.

- ***Python Flask***: és l'encarregat de proporcionar eines i estructures per gestionar sol·licituds

i respostes *HTTP*, encaminament d'*URL*, gestió de sessions, integració de bases de dades, autenticació i moltes altres característiques comunes de les aplicacions web. Per obtenir una descripció més detallada del codi *Python* utilitzat per implementar la pàgina web es pot consultar l'Apèndix A.2.

- ***SQLite***: és un sistema de gestió de bases de dades relacional que s'utilitza per emmagatzemar i administrar dades en aplicacions web. És una base de dades lleugera que permet la creació, lectura, actualització i eliminació de dades de manera eficient. En aquesta nova versió de la pàgina web els models utilitzats són els següents (Figura 5.15 i Taula 5.1) :

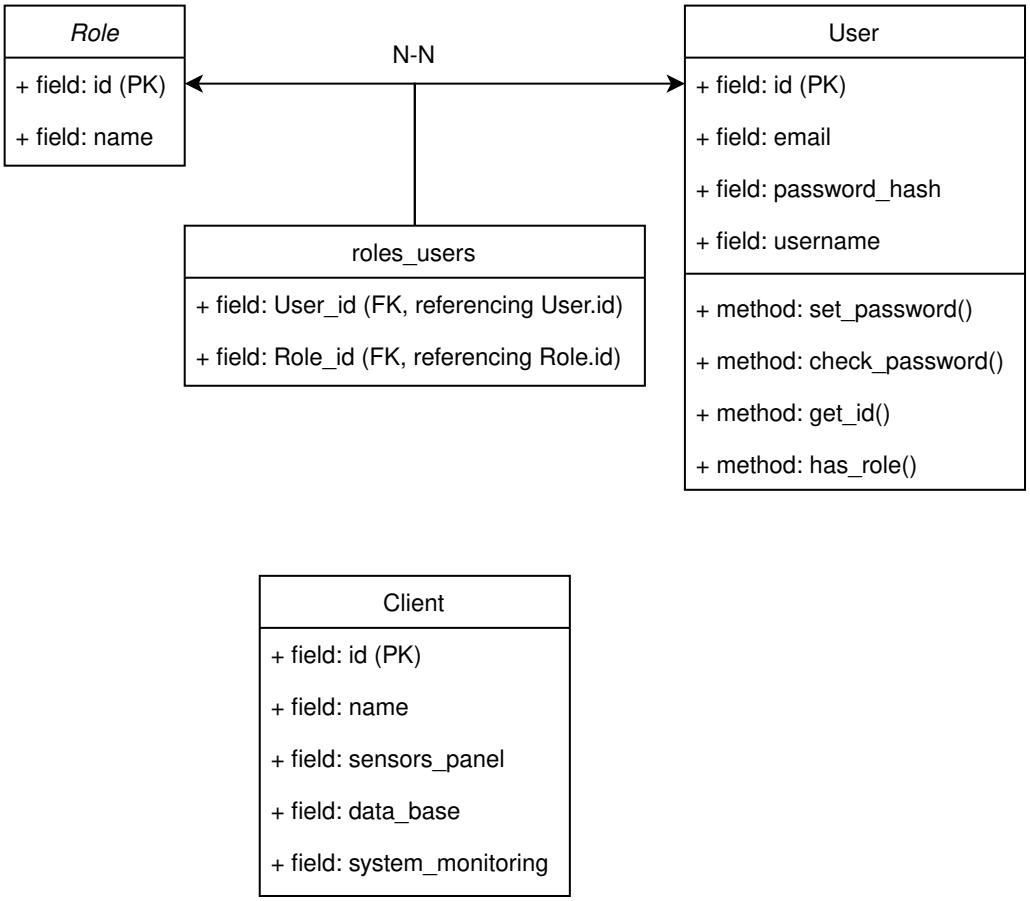

Figure 5.15: Models de la pàgina Web

Aquests models permeten definir clients i usuaris i establir els permisos d'aquests últims en funció del rol. Per tal d'afegir clients i usuaris a la plataforma s'ha creat 2 programes *Python* que es poden utilitzar per visualitzar, afegir, modificar o eliminar usuaris i clients (Apèndix A.3).

Model	**Atribut**	**Descripció**
Role	id (PK)	Clau primària, identificador únic del rol
	name	Nom del rol, no pot ser nul i ha de ser únic
User	id (PK)	Clau primària, identificador únic de l'usuari
	email	Adreça de correu electrònic de l'usuari, única i no pot ser nul·la
	password_hash	Hash de la contrasenya de l'usuari, no pot ser nul·la
	username	Nom d'usuari, únic i no pot ser nul
roles_users	User_id (FK)	Clau forana que fa referència a User.id
	Role_id (FK)	Clau forana que fa referència a Role.id
Client	id (PK)	Clau primària, identificador únic del client
	name	Nom del client, únic i no pot ser nul
	sensors_panel	Panell de sensors, no pot ser nul
	data_base	Base de dades, no pot ser nul·la
	system_monitoring	Monitoratge del sistema, no pot ser nul

Table 5.1: Descripció dels models utilitzats

Capítol 6

Resultats

En aquest capítol, s'exposa els resultats obtinguts a partir de l'execució del projecte. El desenvolupament del projecte es pot veure reflectit en l'evolució i la millora del banc de treball, que serveix com a representació a escala reduïda del sistema final que es vol desplegar a *Chemplate*. Aquest banc de treball ha sofert nombroses revisions i ajustos al llarg del projecte, fins a assolir una versió completament operativa.

Durant el procés de millora, el banc ha incorporat una *Raspberry Pi* que assumeix el rol de node Servidor. Aquest node Servidor conté l'última versió de la pàgina web, detallada al capítol 5. Addicionalment, s'ha afegit un sensor *Modbus* de temperatura i humitat, controlat per la *Raspberry* Client, que emula les funcions dels sensors de la planta pilot de *Chemplate*. Aquest sensor, nomenat "*ARCELI SHT20*" es pot veure a la Figura 6.1.

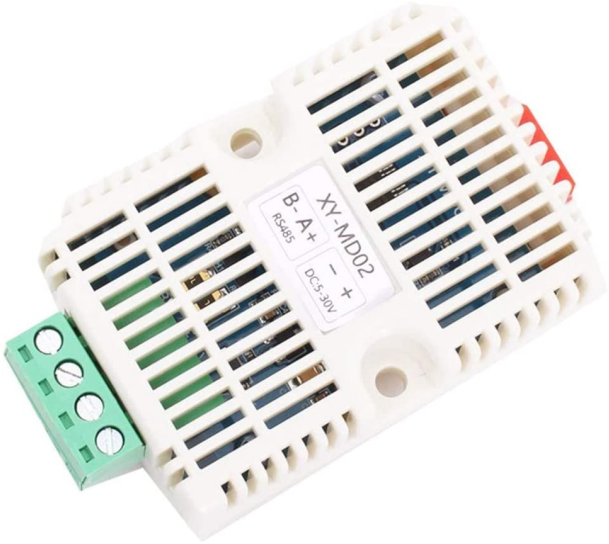

Figure 6.1: Sensor *ARCELI SHT20* utilitzat per llegir la temperatura i la humitat del laboratori

També s'ha decidit prescindir dels *LED IR* del mòdul *PiCamera*, ja que el mòdul de la càmera els activa i desactiva de forma no controlada, la qual cosa afecta negativament els resultats de la correlació descrita en capítols anteriors. Per aquesta raó, s'ha optat per utilitzar únicament la càmera amb una font de llum constant (Figura 6.2).

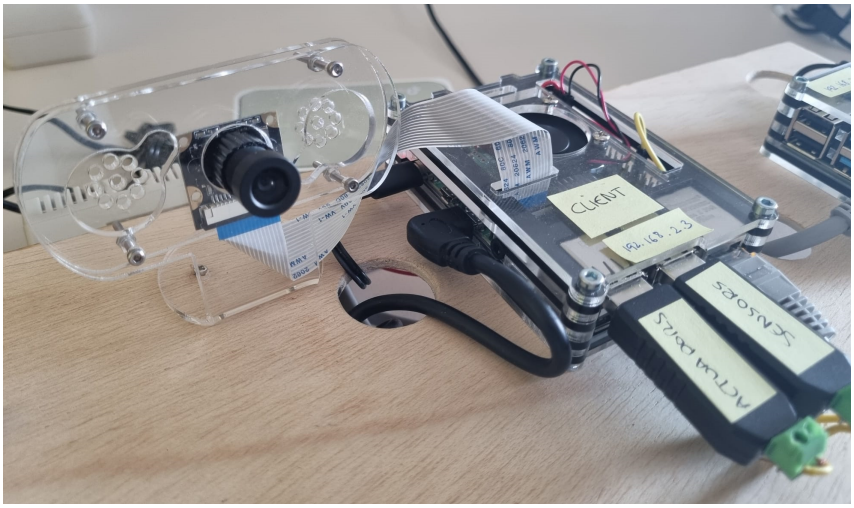

Figure 6.2: Imatge de la càmera final sense *LED IR*

A més a més, a la Figura 6.3, es mostra una imatge ampliada de la pantalla de la *Raspberry Pi* client amb el programa en execució. En aquesta imatge, es poden apreciar els *logs* del programa en funcionament, on es pot veure que el sensor està sent llegit i les dades es transmeten correctament a través de *MQTT* a la *Raspberry Pi* Servidor.

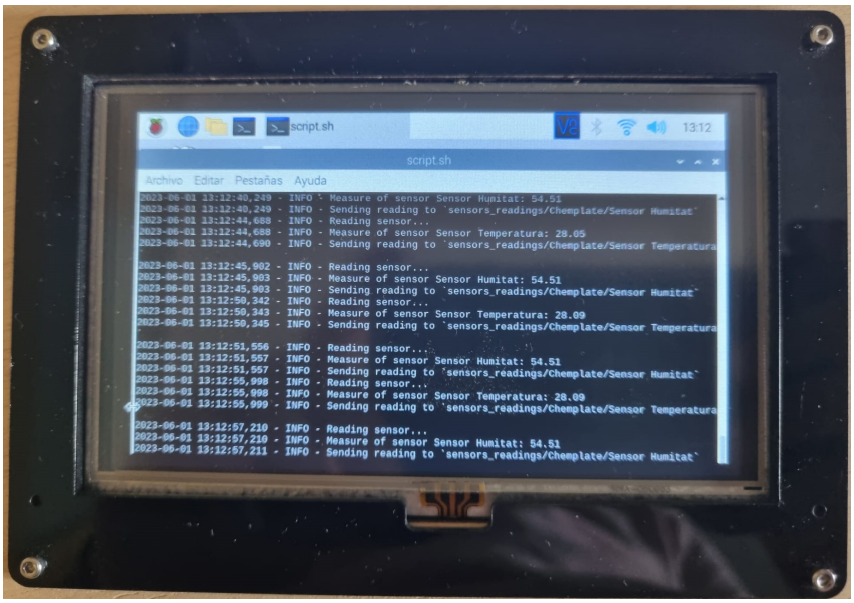

Figure 6.3: Pantalla de la Raspberry Pi client amb el programa en execució

La Figura 6.4 i 6.5 mostra una visió completa del banc de treball i el panell *Grafana* amb la informació captada pel sensor.

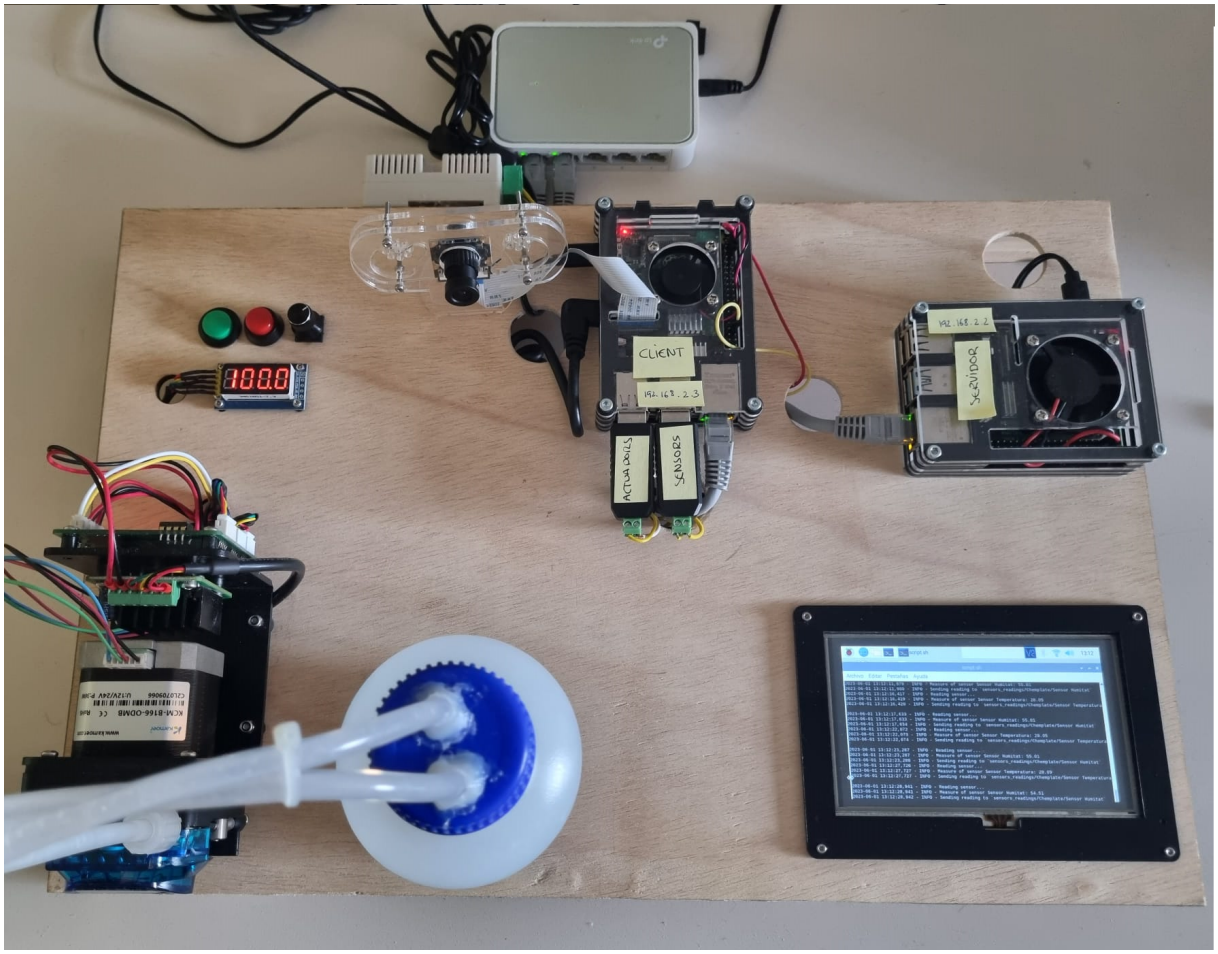

Figure 6.4: Banc de treball final

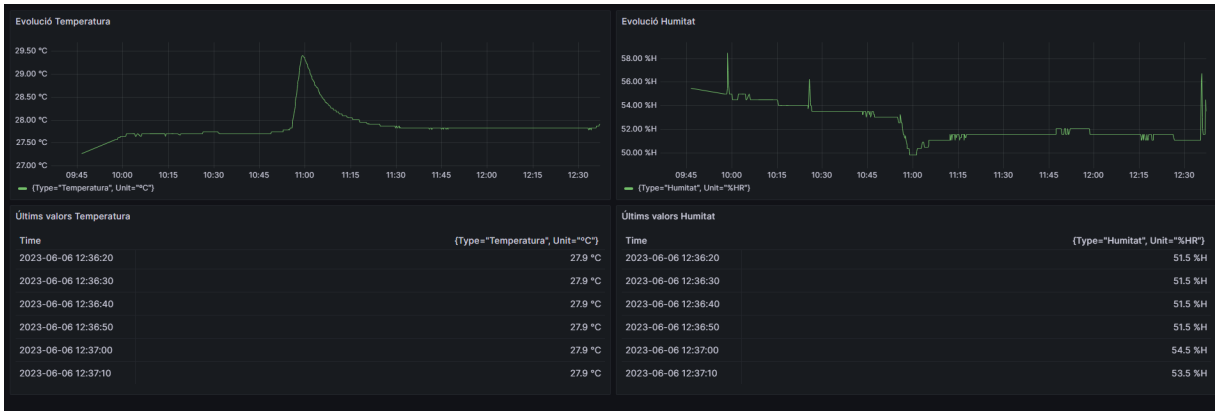

Figure 6.5: Panell de control de *Grafana* amb les lectures de temperatura i humitat del sensor

Capítol 7

Conclusions

Després d'analitzar els resultats obtinguts i el sistema desenvolupat, es pot procedir a l'extracció de conclusions i a la valoració dels objectius preestablerts.

En primer lloc, s'ha desenvolupat i implementat amb èxit un sistema d'alarma avançat que inclou diverses accions. Aquest sistema permet als operaris de *Chemplate* reaccionar ràpidament a canvis inesperats a la planta pilot. Aquest sistema inclou la generació automàtica d'informes i notificacions, a més de la capacitat d'aplicar accions correctives de manera autònoma. També s'ha incorporat un sistema d'anàlisi d'imatges, permetent als usuaris comparar les imatges capturades amb imatges d'estat predefinides. Pel que fa al sistema de visualització en temps real, s'ha integrat una càmera d'alta definició que no només analitza les mescles, sinó que també pot capturar imatges de tota la línia de producció.

Pel que fa a la interfície d'usuari, s'ha creat una plataforma intuïtiva que mostra l'estat dels sensors, actuadors i alarmes en temps real, tot això a través de la nova pàgina web.

Finalment, amb relació a l'adaptabilitat i escalabilitat del sistema, aquest projecte ha demostrat un gran potencial per a la seva futura implementació en múltiples clients de *Chemplate*.

Amb aquestes conclusions, es pot afirmar que s'han complert els objectius del projecte. Malgrat això, és important destacar que sempre hi ha espai per a la millora. Alguns aspectes que es podrien millorar són:

- Optimització del sistema de detecció d'escuma: El sistema actual de detecció d'escuma podria beneficiar-se d'algoritmes més avançats o de l'ús de càmeres de millor resolució. L'ús de l'intel·ligència artificial per a l'anàlisi de dades i la detecció anticipada de problemes o fallades en el sistema de producció podria ser una millora significativa.

- Sistema de notificació més avançat: Es podria implementar un sistema que permeti als operaris gestionar les alertes de manera més eficient, per exemple, incorporant prioritat

d'alertes, historial d'alertes o integració amb sistemes externs per a la notificació immediata als operaris.

- Ampliació d'accions del Sistema d'Alarmes: Podrien incorporar-se més accions al sistema, com ara la notificació als usuaris mitjançant missatges personalitzats o l'enviament de les notificacions a través d'una aplicació mòbil.

Apèndix

A.1 Descripció del codi *Javascript* de la pàgina web (*Frontend*)

L'estructura de carpetes del codi per a aquest projecte de pàgina web està organitzada per a facilitar la implementació i el manteniment. L'organització reflecteix la divisió de tasques del projecte, amb carpetes específiques per a configuracions de clients, instàncies de base de dades, scripts, contingut estàtic i plantilles. Aquesta estructura permet una navegació intuïtiva i eficient del codi, fomentant així una implementació i depuració eficaç.

En aquesta secció, farem una anàlisi breu del codi *JavaScript* que s'ha utilitzat per implementar la interfície d'usuari de la pàgina web. Aquest codi es troba majoritàriament a la carpeta "*static/js*", que conté una varietat de *scripts* organitzats en subcarpetes.

L'arxiu "*actuadors.js*", ubicat a la carpeta *configurations*, controla la configuració i manipulació dels actuadors en la pàgina web. Defineix les constants relacionades amb els actuadors, com els identificadors, tipus, marques i models. A més, inclou la funció "*load_models*", que permet l'actualització dinàmica de la selecció de models disponibles basada en la marca d'actuador escollida per l'usuari.

L'arxiu "*alarmes.js*", ubicat a la carpeta *configurations*, gestiona les configuracions i les funcionalitats relacionades amb les alarmes de la pàgina web. Defineix identificadors, accions i desplegables per a la interacció de l'usuari. A més, inclou diverses funcions per generar condicions d'alarma basades en l'entrada de l'usuari, actualitzar la vista en funció de les seleccions d'alarma i gestionar les interaccions d'entrada de l'usuari amb els desplegables i els camps d'entrada. L'arxiu també proporciona funcionalitats per obtenir noms de sensor i actuadors, i funcionalitats per mostrar i amagar opcions basades en les seleccions d'entrada de l'usuari.

L'arxiu "*sensors.js*" també forma part de la carpeta configurations. Aquest arxiu gestiona la configuració i la interacció de l'usuari amb els sensors en la pàgina web. Defineix les constants necessàries per als sensors, així com els identificadors, les unitats de mesura i els intervals d'actualització. També inclou un conjunt de funcions per a la lectura dels registres Modbus.

A més a més existeix un fitxer nomenat "*base.js*" que conté funcions generals que s'utilitzen

arreu de la pàgina web i un altre nomenat "*cards.js*" que gestiona tota la creació, addició i les tasques d'eliminació de les targetes que s'utilitzen per mostrar les alarmes.

A.2 Descripció del codi *Python* emprat a la pàgina web (*Back-end*)

En aquesta secció es descriu el codi *Python* que implementa la part del servidor de la pàgina web. Es troba principalment en els arxius "*app.py*" i "*models.py*". L'arxiu "*app.py*" és el responsable de gestionar tots els aspectes relacionats amb l'aplicació web, incloent-hi la configuració i la funcionalitat de l'autenticació d'usuari, la interacció amb la base de dades i la gestió de diferents vistes i rutes. Defineix una sèrie de rutes per a diferents operacions, com ara la connexió d'usuaris, la selecció de clients i la configuració de sensors, actuadors i alarmes.

A més, configura la base de dades *SQL* i la funcionalitat de l'autenticació d'usuaris amb la biblioteca *Flask-User*. També conté una sèrie de funcions auxiliars per llegir les configuracions dels fitxers *JSON*, obtenir les dades de sensors i actuadors i manipular les dades d'usuari.

L'arxiu inclou també la funcionalitat de protecció de rutes específiques per a usuaris amb privilegis d'administrador, la gestió d'errors personalitzada i la possibilitat de descarregar i actualitzar les configuracions en format *JSON*. En resum, aquest arxiu gestiona les interaccions de l'usuari amb l'aplicació web, proporcionant una interfície dinàmica i segura per a la configuració i el control del sistema.

L'arxiu "*models.py*" és el que descriu els models i l'estructura de la base de dades que s'utilitza per emmagatzemar els usuaris i els clients.

A.3 Programes per administrar els models de la pàgina web

A.3.1 Gestió d'Usuaris

Aquest programa s'usa per gestionar els usuaris de la base de dades i permet als administradors crear, eliminar, modificar i visualitzar-los. Els usuaris tenen diverses propietats com ara un correu electrònic, un nom d'usuari, una contrasenya (emmagatzemada com un *hash*) i un rol assignat. L'aplicació també gestiona els rols i presenta certa protecció davant errors humans. Si s'intenta crear un usuari amb un correu electrònic que ja existeix, o un rol que ja existeix, el programa no permet dur a terme l'operació.

```python
import os
import sys

# Afegeix el directori arrel al camí Python
root_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), '..'))
sys.path.insert(0, root_dir)

# Importa l'aplicació, la base de dades i les classes Usuari i Rol des del mòdul app
from app import app, db, User,Role

# Funció per imprimir tots els usuaris
def print_all_users():
    # Obté tots els usuaris de la base de dades
    users = User.query.all()

    # Imprimeix la informació de cada usuari
    for user in users:
        print(f"ID: {user.id}, Email: {user.email}, Username: {user.username}, Password Hash: {user.password_hash}, Roles: {[role.name for role in user.roles]}")

# Funció per afegir un nou usuari
def add_user():
    # Demana les dades del nou usuari
    email = input("Enter the email of the user: ") or 'admin@chemplate.com'
    password = input("Enter the password for the user: ") or 'admin'
    username = input("Enter the username for the user: ") or 'Admin'
    role_name = input("Enter the role for the user: ") or 'admin'

    # Verifica si ja existeix un usuari amb aquest correu electrònic
    existing_user = User.query.filter_by(email=email).first()
    if existing_user:
        print(f"A user with the email {email} already exists.")
        return

    # Verifica si ja existeix aquest rol, si no existeix el crea
    role = Role.query.filter_by(name=role_name).first()
    if not role:
        role = Role(name=role_name)
        db.session.add(role)

    # Crea el nou usuari
    new_user = User(email=email, username=username)
    new_user.set_password(password)
    new_user.roles.append(role)

    # Afegeix el nou usuari a la base de dades i confirma la transacció
    db.session.add(new_user)
    db.session.commit()
    print("New user added successfully!")
```

```
49
50 # Funció per eliminar un usuari
51 def delete_user():
52     # Demana el correu electrònic de l'usuari a eliminar
53     email = input("Enter the email of the user to delete: ")
54     # Busca l'usuari en la base de dades
55     user_to_delete = User.query.filter_by(email=email).first()
56     if user_to_delete:
57         # Elimina l'usuari de la base de dades i confirma la transacció
58         db.session.delete(user_to_delete)
59         db.session.commit()
60         print(f"User {email} has been deleted.")
61     else:
62         print(f"No user found with email {email}.")
63
64 # Funció per modificar un usuari
65 def modify_user():
66     # Demana el correu electrònic de l'usuari a modificar
67     email = input("Enter the email of the user to modify: ")
68     # Busca l'usuari en la base de dades
69     user_to_modify = User.query.filter_by(email=email).first()
70     if user_to_modify:
71         # Demana les noves dades de l'usuari
72         new_email = input(f"Enter the new email for the user (leave blank to
    keep '{user_to_modify.email}'): ")
73         new_password = input(f"Enter the new password for the user (leave blank
    to keep the current one): ")
74         new_username = input(f"Enter the new username for the user (leave blank
    to keep '{user_to_modify.username}'): ")
75         new_role_name = input(f"Enter the new role for the user (leave blank to
    keep '{[role.name for role in user_to_modify.roles]}'): ")
76
77         # Modifica les dades de l'usuari
78         if new_email:
79             user_to_modify.email = new_email
80         if new_password:
81             user_to_modify.set_password(new_password)
82         if new_username:
83             user_to_modify.username = new_username
84         if new_role_name:
85             new_role = Role.query.filter_by(name=new_role_name).first()
86             if not new_role:
87                 new_role = Role(name=new_role_name)
88                 db.session.add(new_role)
89             user_to_modify.roles.clear()
90             user_to_modify.roles.append(new_role)
91
92         # Confirma la transacció
93         db.session.commit()
94         print(f"User {email} has been modified.")
95     else:
```

```
96          print(f"No user found with email {email}.")
97
98  # Funció principal
99  def main():
100     # Inicia el context de l'aplicació
101     app.app_context().push()
102
103     # Crea totes les taules en la base de dades
104     db.create_all()
105     while 1:
106         # Demana a l'usuari quina acció vol realitzar
107         answer = input("Do you want to add a new user (a), delete a user (d),
        modify a user (m), or view all users (v)? ")
108
109         # Realitza l'acció segons la resposta de l'usuari
110         if answer.lower() == 'a':
111             add_user()
112         elif answer.lower() == 'd':
113             delete_user()
114         elif answer.lower() == 'm':
115             modify_user()
116         elif answer.lower() == 'v':
117             print_all_users()
118         else:
119             print("Invalid option selected.")
120
121 if __name__ == '__main__':
122     main()
```

A.3.2 Gestió de Clients

Aquest segon programa s'utilitza per gestionar els clients de la base de dades i permet als administradors afegir, eliminar, modificar i visualitzar-los. Cada client té diverses propietats, incloent-hi un nom i tres URL: un que especifica el panell dels sensors del client (*Grafana URL*), un que redirigeix a la base de dades dels sensors (*Sensors URL*), i l'última que redirigeix al servei extern *PiCockpit* (*System Monitoring URL*) de monitoratge. Aquest segon programa també presenta certs mecanismes de proteció davant errades.

```python
import os
import sys

# Afegeix el directori arrel al camí Python
root_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), '..'))
sys.path.insert(0, root_dir)

# Importa l'aplicació, la base de dades i la classe Client del mòdul app
from app import app, db, Client

# Funció per imprimir tots els clients
def print_all_clients():
    # Obté tots els clients de la base de dades
    clients = Client.query.all()

    # Imprimeix el nom i les URL de cada client
    for client in clients:
        print(f"Name: {client.name}, Grafana URL: {client.sensors_panel}, Sensors URL: {client.data_base}, System Monitoring URL:{client.system_monitoring}")

# Funció per afegir un nou client
def add_client():
    # Demana les dades del nou client
    name = input("Enter the name of the client: ") or 'Chemplate'
    sensors_panel = input("Enter the Grafana URL for the client (default: https://www.grafana.com): ") or 'https://www.grafana.com'
    data_base = input("Enter the sensors URL for the client (default: https://www.influxdata.com): ") or 'https://www.influxdata.com'
    system_monitoring = input("Enter the system monitoring URL (default:  https://picockpit.com/mypis/login): ") or ' https://picockpit.com/mypis/login'

    # Verifica si ja existeix un client amb aquest nom
    existing_client = Client.query.filter_by(name=name).first()
    if existing_client:
        print(f"A client with the name {name} already exists.")
        return

    # Crea el nou client
    new_client = Client(name = name, sensors_panel = sensors_panel, data_base = data_base, system_monitoring = system_monitoring)

    # Afegeix el nou client a la base de dades i confirma la transacció
    db.session.add(new_client)
    db.session.commit()
    print("New client added successfully!")

# Funció per eliminar un client
def delete_client():
    # Demana el nom del client a eliminar
    name = input("Enter the name of the client to delete: ")
```

```
46     # Busca el client en la base de dades
47     client_to_delete = Client.query.filter_by(name=name).first()
48     if client_to_delete:
49         # Elimina el client de la base de dades i confirma la transacció
50         db.session.delete(client_to_delete)
51         db.session.commit()
52         print(f"Client {name} has been deleted.")
53     else:
54         print(f"No client found with name {name}.")
55
56 # Funció per modificar un client
57 def modify_client():
58     # Demana el nom del client a modificar
59     name = input("Enter the name of the client to modify: ")
60     # Busca el client en la base de dades
61     client_to_modify = Client.query.filter_by(name=name).first()
62     if client_to_modify:
63         # Demana les noves dades del client
64         new_name = input(f"Enter the new name for the client (leave blank to
       keep '{client_to_modify.name}'): ")
65         new_sensors_panel = input(f"Enter the new Grafana URL for the client (
       leave blank to keep '{client_to_modify.sensors_panel}'): ")
66         new_data_base = input(f"Enter the new sensors URL for the client (leave
       blank to keep '{client_to_modify.data_base}'): ")
67         new_system_monitoring = input(f"Enter the new system monitoring URL for
       the client (leave blank to keep '{client_to_modify.system_monitoring}'): ")
68
69         # Modifica les dades del client
70         if new_name:
71             client_to_modify.name = new_name
72         if new_sensors_panel:
73             client_to_modify.sensors_panel = new_sensors_panel
74         if new_data_base:
75             client_to_modify.data_base = new_data_base
76         if new_system_monitoring:
77             client_to_modify.system_monitoring = new_system_monitoring
78
79         # Confirma la transacció
80         db.session.commit()
81         print(f"Client {name} has been modified.")
82     else:
83         print(f"No client found with name {name}.")
84
85 # Funció principal
86 def main():
87     # Inicia el context de l'aplicació
88     app.app_context().push()
89
90     # Crea totes les taules en la base de dades
91     db.create_all()
92
```

```
    while 1:
        # Demana a l'usuari quina acció vol realitzar
        answer = input("Do you want to add a new client (a), delete a client (d)
    , modify a client (m), or view all clients (v)? ")
        # Realitza l'acció segons la resposta de l'usuari
        if answer.lower() == 'a':
            add_client()
        elif answer.lower() == 'd':
            delete_client()
        elif answer.lower() == 'm':
            modify_client()
        elif answer.lower() == 'v':
            print_all_clients()
        else:
            print("Invalid option selected.")


if __name__ == '__main__':
    main()
```

Ambdós programes fan servir *SQLAlchemy* per interactuar amb la base de dades i comparteixen una arquitectura similar, amb una interacció d'usuari basada en el terminal per triar l'acció desitjada i introduir les dades necessàries.

Bibliografia

[1] Guillem Companys Garcia. "Sistema industrial de monitoratge de processos químics." In: (). URL: https://ddd.uab.cat/record/272824. (accessed: 25.04.2023).

[2] *¿En qué consiste la metodología Kanban y cómo utilizarla?* APD. URL: https://www.apd.es/metodologia-kanban/. (accessed: 25.04.2023).

[3] *Trello*. Trello. URL: https://trello.com/. (accessed: 25.04.2023).

[4] *Raspberry Pi*. Raspberry Pi. URL: https://www.raspberrypi.com/. (accessed: 25.04.2023).

[5] *Docker*. Docker. URL: https://www.docker.com/. (accessed: 25.04.2023).

[6] *MQTT: The Standard for IoT Messaging*. MQTT. URL: https://mqtt.org/. (accessed: 25.04.2023).

[7] *Eclipse Mosquitto™ An open source MQTT broker*. URL: www.mosquitto.org. (accessed: 25.04.2023).

[8] *Influxdata*. Influxdata. URL: https://www.influxdata.com/. (accessed: 25.04.2023).

[9] *Flux vs InfluxQL*. URL: https://docs.influxdata.com/influxdb/v1.8/flux/flux-vs-influxql/. (accessed: 25.04.2023).

[10] *The open observability platform — Grafana Labs*. Grafana. URL: https://grafana.com/. (accessed: 25.04.2023).

[11] *Node-RED: Low-code programming for event-driven applications*. Node-RED. URL: https://nodered.org/. (accessed: 25.04.2023).

[12] *The Modbus Organization*. The Modbus Organization. URL: https://www.modbus.org/. (accessed: 25.04.2023).

[13] *Welcome to Flask — Flask Documentation (2.2.x)*. Flask web development, one drop at a time. URL: https://flask.palletsprojects.com/en/2.2.x/. (accessed: 25.04.2023).

[14] *Chemitec*. Chemitec. URL: https://www.chemitec.it/en/50-series. (accessed: 25.04.2023).

[15] *Github: IzakMarais/reporter*. Github. URL: https://github.com/IzakMarais. (accessed: 25.04.2023).

[16] *PiCockPit: Monitor and Control your Raspberry Pi*. PiCockPit. URL: https://picockpit.com/. (accessed: 25.04.2023).

[17] *Kamoer Peristaltic Pump*. Kamoer. URL: https://www.kamoer.com/kcm.html. (accessed: 01.06.2023).

[18] *Kiwi Electronics*. Kiwi Electronics. URL: https://www.kiwi-electronics.com/en/raspberry-pi-night-vision-camera-2968. (accessed: 01.06.2023).

[19] *OpenLabs*. OpenLabs. URL: https://www.uab.cat/open-labs/. (accessed: 02.06.2023).

[20] *Correlación de Pearson*. Wikipedia. URL: https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson. (accessed: 08.06.2023).

[21] *Onshape — Product Development Platform*. Onshape. URL: https://cad.onshape.com. (accessed: 02.06.2023).