Sistema industrial de monitoratge de processos químics

Guillem Companys Garcia

Resum– Aquest treball presenta una proposta per a un sistema de monitoratge en temps real per a la indústria de la fabricació, específicament per a la planta pilot de Chemplate Materials S.L., una empresa especialitzada en la distribució i producció de productes químics semielaborats.
El sistema proposat pretén augmentar l'eficiència i la productivitat en el procés de fabricació llegint els sensors de la planta pilot, enviant aquesta informació a un servidor per a l'anàlisi i oferint una interfície web per visualitzar aquestes dades i alertes en cas d'errors.
El sistema permetrà el monitoratge remot, fet que permet una major flexibilitat i facilitat de control sense la necessitat de ser a l'emplaçament. El sistema es basa en diferents tecnologies com Python, Docker, Grafana, InfluxDB, Mosquitto i Node-Red per poder fer una anàlisi de les dades, la seva visualització i emmagatzematge i la comunicació entre els dispositius.

Paraules clau– Automatització, Monitorització en temps real, Sensors, Optimització dels processos de fabricació, Visualització de dades, Microcontroladors.

Abstract– This paper presents a proposal for a real-time monitoring system for Chemplate Materials S.L., a company that specializes in the distribution and production of chemical semi-finished products.
The proposed system aims to increase efficiency and productivity in the manufacturing process by reading sensors in the pilot plant, sending that information to a server for analysis, and offering a web interface for viewing that data and alerts in case of errors. This system will enable remote monitoring which allows for greater flexibility and ease of control without the need to be on site.
The system relies on different technologies such as Python, Docker, Grafana, InfluxDB, Mosquitto and Node-Red to be able to perform the data analysis, visualization, storing and messaging between the devices and systems.

Keywords– Automation, Real-time Monitoring, Sensors, Manufacturing process optimization, Data visualization, Microcontrollers.

1 Introducció - Context del treball

L'automatització ha permès dur a terme tasques, sovint repetitives, amb molt poca intervenció humana. Gràcies a l'avenç de noves tecnologies com l'Internet de les Coses (IoT), l'anàlisi Big Data, la connectivitat 5G o la intel·ligència artificial, l'automatització s'ha estès i ha permès descobrir nous camps d'aplicació [ED].

En la indústria, aquestes noves tecnologies juntament amb l'abaratiment dels microcontroladors, ha permès optimitzar els processos productius, augmentar l'eficiència operacional i millorar la productivitat de les empreses.

Processos completament manuals com el muntatge de peces en una cadena de producció, s'han substituït quasi totalment per màquines integrades dins HMI (Human Machine Interface), molt més eficients, productives i amb taxes d'error molt més baixes [AC, RE].

Una de les aplicacions dels sistemes HMI és el monitoratge remot, que permet una major flexibilitat i facilitat de control sense necessitat d'estar a la instal·lació in situ.

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 es-

• E-mail de contacte: guillem.companys@gmail.com
• Menció realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Màrius Montón Macian (Departament de Microelectrònica i Sistemes Electrònics)
• Curs 2022/23

tan equipades amb diversos tancs i diferents solucions controlades a través de sensors de tota mena. Durant el procés de fabricació, els químics controlen magnituds com el pH, la temperatura o la conductivitat i modifiquen la composició dels banys si és necessari.

Avui dia, aquest protocol és majoritàriament manual i els tècnics especialistes de Chemplate han de desplaçar-se i visitar presencialment les fàbriques dels clients per tal de recollir les mostres. Llavors s'envien les mostres al laboratori, on s'analitzen i, si és necessari, es corregeixen els banys.

El procés de desenvolupament sol durar una setmana i l'automatització d'aquestes mesures suposaria un estalvi econòmic significatiu i un augment evident de la productivitat. A més a més, es podrien millorar les plantes pilot afegint noves funcionalitats, com mostrar gràficament les dades recollides en una pàgina web o avisar automàticament als tècnics en cas d'errors en les mescles.

2 OBJECTIUS

Aquest treball té com a objectiu principal crear un sistema capaç de monitorar en temps real diversos sensors. Així i tot, a continuació es mostren propòsits específics per tal de millorar el sistema i afegir-hi noves funcionalitats:

- Llegir els sensors de la planta pilot de Chemplate i enviar la informació recollida a un servidor per al seu tractament i posterior lectura.
- Oferir a l'usuari la possibilitat de triar l'interval de temps entre la recollida de mostres de cada sensor (cada minut, cada hora, un cop al dia, un cop a la setmana...)
- Permetre a l'operari configurar i calibrar els sensors de forma fàcil i a través de plantilles preestablertes per ajudar a usuaris novells o no experts.
- Oferir una pàgina web on consultar les dades recollides pels sensors. També es podrà consultar les dades de les fàbriques dels clients per assegurar un correcte monitoratge remot i es podrà fer un històric de les dades i analitzar-les per extreure conclusions.
- Permetre a l'usuari especificar la manera de reproduir la informació desitjada a través d'una interfície amigable amb moltes possibilitats de configuració. Es podran crear gràfics, taules o figures a petició de l'empresa.
- Utilitzar sensors alternatius per fer estimacions de magnituds difícilment mesurables o que no es poden mesurar directament.
- Permetre a l'usuari aplicar correccions amb actuadors i avisar en cas de defectes en la mescla. Enviar també informes de seguiment de forma automàtica a través de correu electrònic.
- Mostrar per pantalla informació rellevant del sistema in situ com els valors de l'última mesura.

3 METODOLOGIA I PLANIFICACIÓ

Existeixen diferents tipus de metodologies de treball que es poden utilitzar pel disseny de Software. Aquestes es poden classificar en metodologies tradicionals i metodologies àgils (*agile*) [BE]. Cada una d'elles presenta els seus avantatges i inconvenients.

Pel que fa a les metodologies tradicionals, les més destacades són:

- *Waterfall* és una metodologia lineal en la qual les etapes s'organitzen de dalt a baix, en fases seqüencials que obeeixen un ordre rigorós.
- La metodologia de prototip, una metodologia iterativa basada en la construcció d'un prototip que permet als usuaris provar-lo i aportar *feedback*.
- La metodologia d'espiral, una combinació dels dos models anteriors amb algunes diferències.
- La metodologia de model incremental, caracteritzada per la construcció d'un producte final de manera progressiva.
- RAD (*Rapid Application Development*) és una metodologia que permet desenvolupar programari d'alta qualitat en un curt període de temps, però requereix una major intervenció dels usuaris.

Les metodologies *agile* més utilitzades són:

- Kanban divideix les tasques en petites porcions i les organitza en un tauler de treball per a crear un flux visual i prioritzar tasques.
- Scrum és una metodologia incremental que divideix els requisits del projecte en *sprints*, on es planifica, s'executa i s'avalua la feina feta.
- Lean s'enfoca en petits equips altament capacitats per a fer tasques ràpidament .
- XP, una metodologia que es basa en relacions interpersonals per a crear un ambient de treball en equip i tenir feedback constant del client.

Al ser aquest un projecte unipersonal i difícil de predir, el sistema de treball escollit per la realització del projecte ha estat la metodologia Kanban [AP]. Aquesta metodologia, molt flexible davant possibles canvis, consisteix en la creació de targetes,que corresponen a les tasques a realitzar, i estats per definir el progrés de cada tasca (Pendent, En procés i Fet).

L'eina utilitzada per seguir aquesta metodologia ha estat Trello [TR], una eina que permet gestionar un projecte mitjançant la creació de tasques i visualitzar el projecte a través de diagrames. En la Figura 1 s'adjunta el diagrama de Gantt resultant, on es pot apreciar la distribució de les tasques i els seus temps associats.

Tal com es mostra a la figura, les tasques s'han separat en 3 camins. El primer d'ells inclou totes les tasques que permetran crear un sistema funcional bàsic. El següent conjunt de tasques són aquelles que amplien les funcionalitats del sistema i afegeixen característiques i el milloren. Per últim, existeix un camí on es recull les tasques acadèmiques.

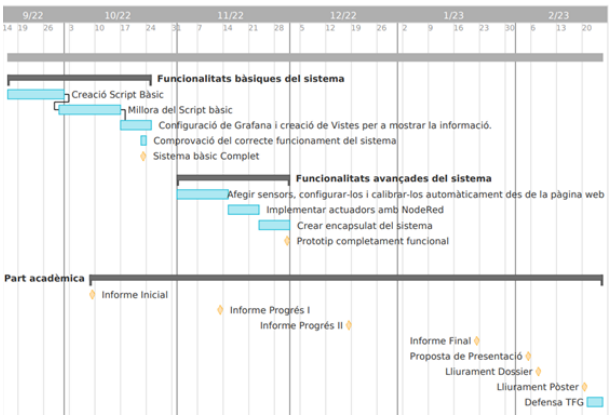

Fig. 1: Diagrama de Gantt Inicial del Projecte

4 Arquitectura del Sistema

El projecte ha estat dissenyat seguint el model de programació Client-Servidor descrit a la Figura 2. S'ha escollit aquest paradigma ja que es pretén augmentar el número de clients connectats al sistema i es busca poder accedir a la informació de cada client des d'un lloc centralitzat. També es busca delegar les tasques de recol·lecció i transformació de la informació rebuda dels sensors. Per aquest motiu, cada client controla els seus respectius sensors a través del protocol Modbus [MO] i envia la informació ja processada al servidor.

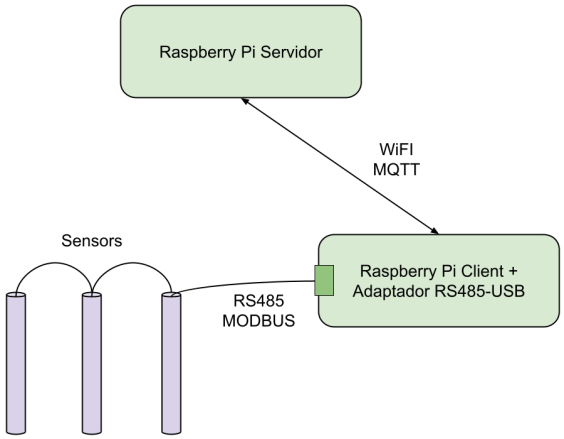

Fig. 2: Arquitectura del Sistema

L'intercanvi d'informació entre els diferents nodes del sistema es realitza amb el protocol MQTT [MQ], el qual permet una connexió física a través d' Ethernet o Wifi si es desitja una connexió sense fils.

Delegant les tasques de comunicació i recol·lecció d'informació als clients, s'aconsegueix alliberar recursos del node Servidor que es poden destinar a implementar altres funcionalitats del projecte.

Tot i que el sistema permet una relació Servidor-Client d'1-N, per a la realització pràctica d'aquest treball s'ha optat per utilitzar solament un client. En els següents apartats es descriu el Hardware i Software que presenta el sistema i les seves característiques.

5 Arquitectura Hardware

El sistema es troba desplegat sobre dos Raspberry Pi 4B, on una placa pren el paper de Servidor i l'altra el paper de Client. S'ha emprat aquestes plaques ja que la Raspberry Pi 4B és un microprocessador de baix cost relativament potent amb àmplia documentació molt recomanable com a placa de prototipatge.

Tot i així, s'ha estudiat la possibilitat de desplegar el projecte en un servidor Cloud. D'aquesta manera, només seria necesària una Raspberry Pi, que actuaria com a client. Existeixen una gran varietat d'empreses que ofereixen servidors d'aquest tipus: Microsoft, Amazon, IBM...

Un desplegament cloud del servidor aportaria resiliència al sistema i delegaria tasques de manteniment que s'hauran de dur a terme en un futur.

Els models adquirits per a la realització d'aquest projecte compten amb 4 GB de RAM i utilitzen un processador ARM Quad Core Cortex-A72 a 1.5 GHz [RA]. També disposen de connexió Wifi i Gigabit Ethernet, essencial per comunicar-se a través del protocol MQTT que detallarem més endavant. Aquests microprocessadors també compten amb un sistema operatiu molt similar a Debian, nomenat Raspbian.

Les plaques es troben encapsulades dins de carcasses protectores amb sistemes de refrigeració actius i passius per tal d'evitar el sobreescalfament. La Raspberry Servidor presenta també una pantalla de 640×480p amb connexió HDMI, on es mostra les lectures dels sensors rebudes de la Raspberry Client.

En la Figura 3 es pot observar l'adaptador Modbus-USB (DSD Tech SH-U10) juntament amb el microprocessador escollit.

Fig. 3: Raspberry Pi 4B 4GB amb l'adaptador Modbus-USB

6 Arquitectura Software

El software està disponible per a la seva visualització en el següent enllaç:

https://github.com/guillemcompanysgarcia/TFG

6.1 Servidor

La majoria de Software desplegat a la Rasbperry 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 per a que el programa s'executi.

Una de les avantatges de Docker [DO] és la seva versatilitat. Al tenir la certesa que el codi s'executarà en qualsevol entorn, el projecte és fàcilment migrable i es podria desplegar en un servei Cloud sense problemes en cas que fos necessari.

Els serveis i aplicacions necessàries pel 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 i Python. L'aplicació utilitza 5 contenidors, que es detallen a continuació:

- El contenidor InfluxDB, que és l'encarregat d'emmagatzemar les lectures dels sensors dels client.
- El contenidor Grafana, aplicació que utilitzem 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 utilitzem 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 la arquitectura del servidor (Figura 4) i les diferents capes del Stack que el formen.

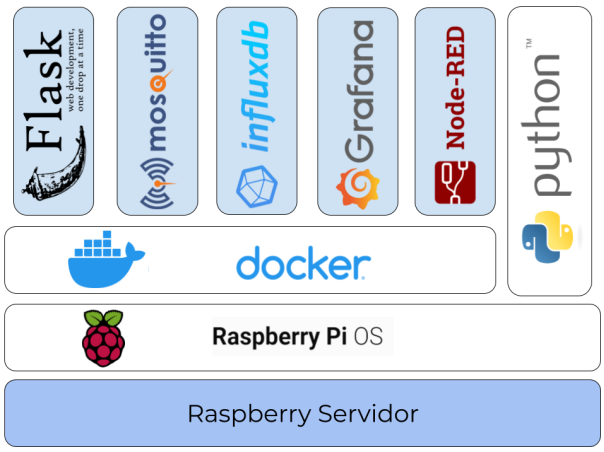

Fig. 4: Stack del Servidor

6.1.1 Servidor Web

Com a *front-end* s'utilitza el *stack* format pels llenguatges HTML, CSS i Javascript. Aquest últim llenguatge dona molta llibertat per modificar el comportament de la pàgina durant el temps d'execució.

El *back-end* de la pàgina utilitza Flask, un micro *framework* web programat en Python [FL]. Per tal d'implementar les funcionalitats necessàries per otorgar una bona experiència a l'usuari Javascript realitza múltiples crides a Flask i viceversa.

Una vista de la pàgina web principal es troba recollida a la Figura 5. Tal com es pot observar, l'usuari té la possibilitat de llegir els sensors, configurar-los, consultar la base de dades o configurar els actuadors. També s'afegeix la possibilitat d'enviar un resum per correu o consultar l'estat actual del sistema.

Fig. 5: Pàgina Web Principal

Per a la lectura de les dades s'envia a l'usuari al panell de Grafana corresponent. Les opcions 'Base de datos' i 'Configuración de los actuadores' redireccionen a l'usuari cap a la interfície web de InfluxDB i Node-RED respectivament.

La opció 'Generar Reporte del Sistema' fa una crida al contenedor auxiliar de Grafana per tal de generar l'informe desitjat i l'opció 'Estado del Sistema' redirigeix a l'usuari cap al servei de monitoratge extern.

Quan l'usuari prem 'Configuración de los sensores' l'operari es troba amb la interfície de la Figura 6.

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 la lectura o l'adreça i el número de registres a llegir.

Fig. 6: Pàgina Web Configuració

Els sensors utilitzats a Chemplate es llegeixen a través del controlador 50 Series de Chemitec [CH]. 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 utilitzar. 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.

Els sensors configurats es recullen a la taula de la part superior de la Figura 7, on es mostra la informació de cada sensor juntament amb 3 accions a realitzar sobre aquest.

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 per a que 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 7 es troba recollit una mostra d'aquest arxiu amb la configuració de tres sensors ficticis.


```
[
{
"nombre":"Sensor 1",
"tipodesensor":"Temperatura",
"intervalodetiempo":"Cada minuto",
"funciónmodbuslectura":"Function 01 (Read Coil Status)",
"dirección":"15",
"nºregistros":"2",
"comentarios":"Sensor del cubell 1"
},
{
"nombre":"Sensor 2",
"tipodesensor":"Presión",
"intervalodetiempo":"Una vez al día",
"funciónmodbuslectura":"Function 04 (Read Input Registers)",
"dirección":"18",
"nºregistros":"8",
"comentarios":"Sensor de la planta situat a la dreta del termòmetre"
},
{
"nombre":"Sensor 3",
"tipodesensor":"Otro",
"intervalodetiempo":"Cada hora",
"funciónmodbuslectura":"Function 16 (Preset Multiple Registers)",
"dirección":"20",
"nºregistros":"2",
"comentarios":"Nou Sensor de ORP"
}
]
```


Fig. 7: Arxiu JSON amb la configuració dels 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.

6.1.2 Broker MQTT

El protocol escollit per comunicar els diferents nodes del sistema és MQTT (MQ Telemetry Transport). MQTT és un protocol de xarxa lleuger, de tipus publicació-subscripció, i màquina a màquina per al servei de cues de missatges.

Està dissenyat per a connexions amb ubicacions remotes, que tenen dispositius amb restriccions de recursos o una amplada de banda de xarxa limitat. 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 enruta 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.

La informació s'organitza per *topics*. Quan un client té una nova dada que distribuir, envia un missatge al broker connectat, 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.

Diversos serveis online ofereixen brokers web gratuïtament. Tot i així, per a la realització d'aquest projecte s'ha escollit implementar el broker localment dins Docker. D'aquesta manera es té més control i possibilitats de configuració, i s'augmenta la resiliència del sistema, ja que no es depèn d'un element extern.

El broker instal·lat dins del container utilitza l'alternativa *open source* Eclipse Mosquitto [EC].

6.1.3 InfluxDB

InfluxDB és una base de dades de sèries temporals (TSDB) de codi obert ideal pel monitoratge d'operacions, de les mètriques de les aplicacions o de les dades dels sensors de la Internet de les Coses (IoT) [IN].

És una base de dades que funciona amb tres tipus d'estructures de dades: els *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 de punts agrupats per una parella clau-valor específica formen una sèrie. Un conjunt de sèries agrupades per un identificador formen part d'una mesura.

Per tal d'inserir i llegir dades, InfluxDB compta amb dos llenguatges pròpis: 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 [FL]. A més a més, compta amb múltiples biblioteques per a diferents llenguatges (Javascript, Python, Go... entre altres). Per a la lectura i escriptura d'informació a la base de dades aquest projecte utilitza aquest segon llenguatge.

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

6.1.4 Grafana

Grafana és un component popular en serveis de monitoratge, sovint utilitzat en combinació amb bases de dades de sèries temporals com Prometheus, Graphite o, en el nostre cas, InfluxDB [GR].

Grafana és una aplicació web d'anàlisi i visualització interactiva de codi obert multiplataforma que proporciona taules, gràfics i alertes. Grafana permet crear panells de control complexos utilitzant constructors de consultes interactives.

Durant les primeres fases del projecte, les dades introduïdes a la base de dades s'inserien a través d'un script Python *multithread*, on les dades seguien una distribució gaussiana amb una variança preestablerta.

La Figura 8 recull el panell generat amb les dades del script, on es mostren la temperatura, acidesa i conductància de tres cubells.

Fig. 8: Panell de Grafana

6.1.5 Node-RED

Per tal que el sistema sigui capaç de respondre a variacions i errors en les mescles dels cubells, el projecte utilitza Node-RED. Node-RED és una eina de programació visual àmpliament utilitzada per a connectar diferents dispositius amb APIs i serveis en línia [NO].

La seva interfície web ens ofereix un editor que facilita la creació de *workflows* utilitzant l'àmplia varietat d'opcions i llibreries compatibles.

El *workflow* de la Raspberry Servidor es troba recollit en la Figura 9.

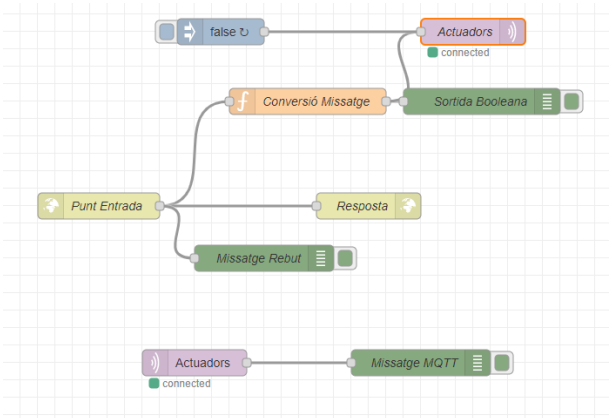

Fig. 9: Workflow del Servidor

Quan el pH de la planta pilot supera cert *threshold*, Grafana alerta automàticament a Node-RED a través d'un *Webhook* propi.

Un cop arribada l'alerta, aquesta és resposta i processada a través del bloc 'Conversió de Missatge', que analitza el contingut d'aquesta. Si és necessari, es genera una sortida booleana True que s'envia a través del protocol MQTT a la Raspberry Client, encarregada d'accionar els motors que permeten la inserció de solucions base o àcid per corregir la mescla.

Per tal de millorar la suavitat de resposta, cada 5 segons s'envia una sortida booleana False per tal de tancar el motor, ja que per limitacions del sistema aquest no pot enviar alertes en intervals inferiors a 20 segons.

6.1.6 Python

El projecte està control·lat per un script Python modularitzat en diverses llibreries pròpies, per tal de millorar la llegibilitat i reusabilitat del codi.

D'aquesta manera, la programació es pot realitzar a alt nivell i el tractament dels possibles errors i mal funcionaments es delegen a les llibreries corresponents.

En la Figura 10 es troba descrit el diagrama de flux del codi Python del servidor. Tal com es pot observar, des de l'arxiu main.py s'accedeix a les diferents llibreries de les que disposa el programa:

- Llibreria MQTT: és l'encarregada de permetre la comunicació entre la Raspberry Client i la Servidor.
- Llibreria InfluxDB: és l'utilitzada per enviar la informació dels sensors a la base de dades.
- Llibreria Email: llibreria per enviar correus electrònics que s'utilitza per enviar un resum de les lectures del dia en format PDF. Aquesta funcionalitat es troba detallada a l'apartat Grafana Reporter.

El diagrama de flux resum de la Raspberry Servidor presenta l'estructura següent:

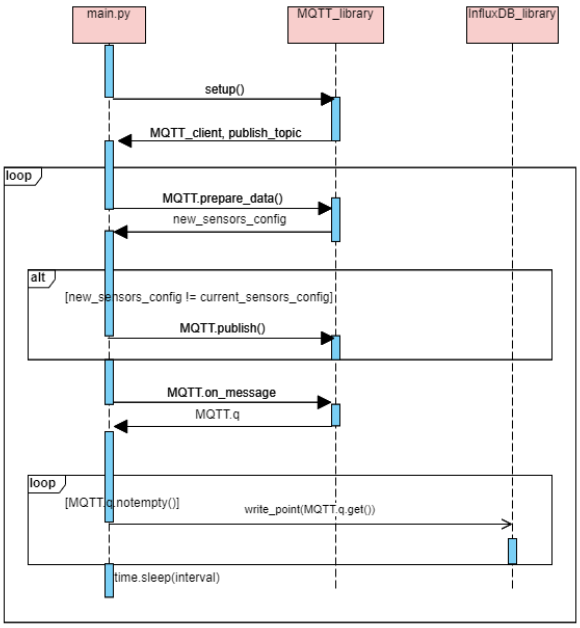

Fig. 10: Diagrama de flux del Servidor

Tal com es pot observar, en primer lloc, s'inicialitza el client MQTT per tal de poder enviar i rebre informació.

Tot seguit es llegeix l'arxiu de configuració creat per la pàgina web i, en cas que hagi patit alguna modificació, s'envia a la Raspberry Client.

Per últim, es comprova si el Client ha enviat alguna mesura i, en cas afirmatiu, s'envia la nova informació a la base de dades utilitzant la llibreria InfluxDB.

Les mesures realitzades pels sensors es guarden en *points*, on s'emmagatzema el nom del sensor, la data de la mesura i el valor d'aquesta.

6.1.7 Grafana Reporter

Un dels objectius del projecte és 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 per tal d'aconseguir-ho.

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

Dins del script Python del servidor existeix un timer 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 al usuari.

L'enviament del PDF a través del correu és automàtic i es realitza a través d'una llibreria de Python creada exclusivament per aquest projecte.

6.2 Client

La majoria de software de la Raspberry Client es troba recollit en un script de Python modul·laritzat en llibreries, que s'encarrega d'inicialitzar els sensors, calcular quan és la pròxima lectura a realitzar i d'enviar la informació corresponent a la Raspberry Servidor a través de MQTT. També es troba instal·lada Node-RED, que s'utilitza per controlar els actuadors del sistema.

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 11) .

Fig. 11: Stack del Client

6.2.1 Python

El script de Python de la Raspberry Client conté un fitxer main.py i les següents llibreries:

- Llibreria MQTT: és l'encarregada de permetre la comunicació entre la Raspberry Client i la Servidor.
- Llibreria MODBUS: és l'utilitzada per communicar-se amb els sensors a llegir.
- Llibreria de Control: és la llibreria que permet interconnectar totes les parts del sistema.

També existeix una classe de tipus Sensor on s'emmagatzema tota la informació relativa a aquest: nom, tipus, interval de lectura, funció modbus a utilitzar...entre altres.

En la Figura 12 es recull el diagrama de flux resum que segueix el programa.

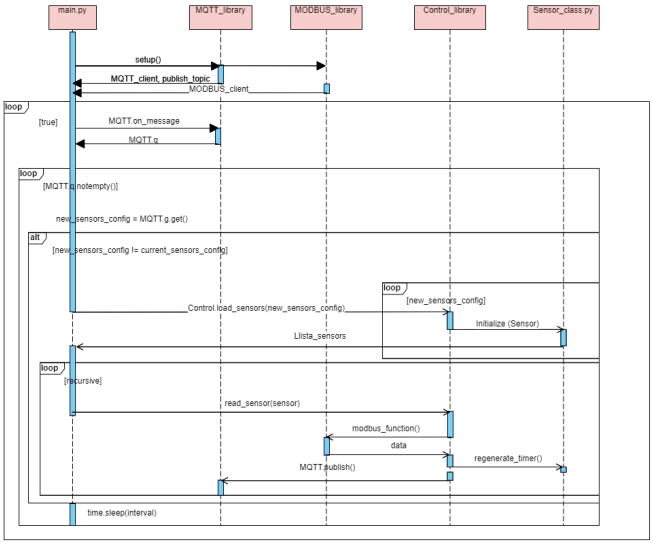

Fig. 12: Diagrama de flux del Client

En primer lloc es criden a les funcions de setup de la llibreria MQTT i MODBUS. Tot seguit, s'espera fins que arribi el fitxer de configuració des de la Raspberry Servidor. Un cop arribat, es compara amb la configuració actual i, en cas de ser diferent, s'envia a la llibreria de control.

Dins la llibreria de control es creen els objectes de la classe Sensor i es retornen al fitxer main, on s'utilitzen, juntament amb un sistema planificador, per programar les lectures dels sensors.

Quan es realitza la tasca de lectura, es crida a la funció de la llibreria de control que s'encarrega de comunicar-se amb el sensor a través de la llibreria MODBUS. També es regenera el *timer*. A continuació, s'utilitza la llibreria MQTT per traslladar la informació llegida a la Raspberry Servidor, que s'encarrega d'escriure-la a la base de dades.

Després de la lectura, el sistema entra en repòs fins que la següent tasca de lectura és cridada.

6.2.2 Node RED

El *workflow* del node client és l'encarregat de comunicar-se amb el motor.

L'esquema de la Figura mostra com es rep a través del broker situat al node Servidor les ordres de control del motor.

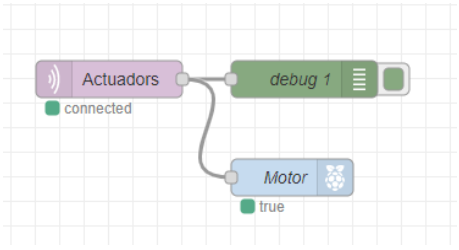

Fig. 13: Workflow del Client

Aquest motor, similar al recollit en la Figura 14, es connecta a la Raspberry Pi i es controla a través dels pins GPIO d'aquesta.

La bomba peristàltica presenta un botó per encendre, un per apagar i un altre per controlar el flux de la solució que injecta.

Fig. 14: Motor que s'utilitzarà com a actuador

6.3 Altres Funcionalitats del Sistema

6.3.1 Picockpit

Per tal de monitoritzar el sistema s'utilitza el servei *Picockpit* [PI]. Aquest servei ofereix una interfície web per monitorar les Raspberry Pi de l'usuari. També ofereix la possibilitat de controlar els dispositius i avaluar el seu estat a través d'exàmens de diagnosi.

Per una altra banda, recull estadístiques com l'ús de la CPU, la temperatura d'aquesta, la quantitat de memòria RAM disponible, etc.

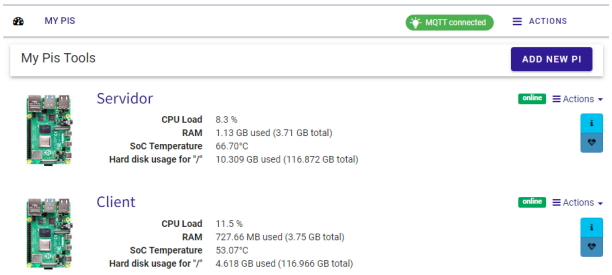

Fig. 15: Estat del Sistema

7 Dificultats trobades durant la implementació de la solució a l'empresa

En aquest apartat es fa un recull del conjunt de problemes trobats durant la realització del projecte. Alguns d'ells s'han pogut corregir, mentre que altres ens han obligat a modificar certs aspectes i característiques del projecte.

- Inicialment es pretenia llegir cada sensor de forma individual. Degut a limitacions del sensors de Chemplate (propietaris d'una empresa externa), la Raspberry Pi Client llegeix el controlador que configura i calibra els sensors. Tot i això, si s'utilitzessin sensors amb documentació oberta, el sistema seria capaç de comunicar-se amb cada sensor de forma individual per obtenir les lectures i calibrar-lo.

- El servei Grafana Reporter, utilitzat per enviar correus electrònics amb resums de les lectures dels clients, no és implementable dins de l'estructura de Chemplate. El Grafana Reporter utilitza una imatge modificada de Grafana per renderitzar les imatges del panell dels sensors.

  Degut a que només existeix aquest servei per a arquitectures x86, no és possible utilitzar-lo amb les Raspberry's que presenten una arquitectura ARM. S'ha intentat utilitzar alternatives i *workarounds* sense èxit.

- La debilitat de l'Internet de l'empresa dificulta la comunicació entre la Raspberry Client i la Servidor. Per aquest motiu s'han implementat solucions al codi per oferir tolerància a errors de connexió. Per exemple, s'ha augmentat el QoS (Quality of Service) del protocol MQTT. Aquest paràmetre indica la política d'entrega dels missatges, que ha passat de 0 (*best-effort*) a 2 (*four-part handshake*). També s'han creat scripts shell que aixequen els serveis desplegats en cas d'error.

- També s'han solucionat diversos errors (*bugs*) del codi Python. El més important d'ells provocava lectures incorrectes, ja que era degut a una interpretació errònia dels registres dels sensors.

  En aquest tipus de sensors les dades es troben en diferents registres de 16 bits (2 bytes). Normalment, per llegir el valor és necessari accedir a 2 registres i concatenar les lectures per després fer una conversió de hexadecimal a punt flotant.

  Existia un error en la concatenació que provocava que, en cas que el valor del segon registre no utilitzés 4 dígits, els dígits del primer registre fossin menys significatius. Després d'un anàlisi exhaustiu, es va implementar un sistema de mesura que té en compte tots els *edge cases* possibles i s'ha pogut solucionar el problema.

8 Desplegament Final

En la Figura 16 es pot apreciar la configuració dels sistema que permet fer la lectura de dos sensors: un sensor de pH i un de conductància.

Fig. 16: Configuració del sistema amb els 2 sensors esmentats

Els sensors de pH i conductància estan configurats per fer lectures en intervals de 10 segons. La funció Modbus utilitzada és la funció 04, encarregada de llegir els registres d'entrada.

Els dos registres del sensor de pH comencen a la direcció 00 de memòria, i són consecutius als dos registres del sensor de conductància.

En la Figura 17 es pot veure la Raspberry Client instal·lada damunt del controlador de la planta pilot. Dins del cercle vermell es pot veure una imatge ampliada de la Raspberry i l'adaptador Modbus-USB que s'utilitza per la comunicació Modbus amb el controlador.

Fig. 17: Instal·lació del projecte dins la planta pilot de Chemplate

El panell Grafana amb les lectures d'ambdós sensors es recull en la Figura 18. Tal com es pot veure, el panell està format per 4 gràfics, dos per cada sensor (un mostra l'evolució i l'altre el valor actual).

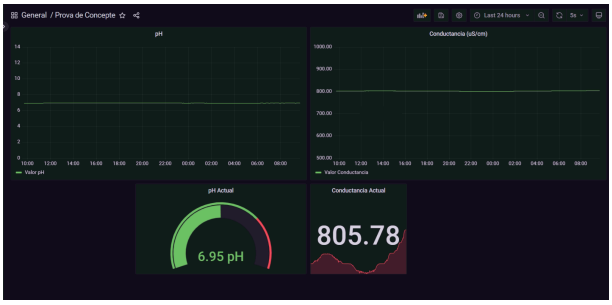

Fig. 18: Panell Grafana del sensor de pH i de conductància

9 Conclusions

Aquest treball pretenia crear un sistema capaç de monitorar diversos sensors en temps real. Aquest objectiu, juntament amb la majoria de funcionalitats que es volien afegir al sistema, han estat creades i implementades amb èxit.

Així i tot, malgrat la complexitat inicial del projecte, s'ha aconseguit crear un sistema resilient, fàcil de replicar i mantenir i àmpliament escalable que s'adapta a les necessitats del client.

No obstant això, algun objectiu específic com l'objectiu d'oferir una manera fàcil de configurar i calibrar els sensors d'una manera senzilla no s'ha pogut dur a terme, ja que la instal·lació de Chemplate no permetia la comunicació directa amb els sensors. Amb tot i això, el programa creat contempla la possibilitat de comunicar-se directament amb els sensors i només s'hauria de modificar els arxius de configuració per tal d'adaptar-se a la nova arquitectura.

Per una altra banda, s'han descartat propostes inicials com l'estimació de magnituds terciàries ja que no era una prioritat per Chemplate i també s'ha modificat l'objectiu de mostrar per pantalla informació rellevant del sistema, ja que s'ha decidit utilitzar una interfície web més còmoda i senzilla.

Chemplate Materials S.L. ja ha expressat el seu desig de continuar el projecte fora del marc establert per aquest TFG. Chemplate busca crear un sistema a mesura capaç de cobrir les seves necessitats que permeti a l'empresa recollir les dades de les plantes pilot per analitzar-les i preveure possibles tendències.

Durant el 2023, el sistema creat es traslladarà a diverses plantes pilot d'arreu el territori i es preveu tenir fins a 3 plantes pilot funcionant a la vegada.

D'aquesta manera, es podrà reaccionar amb antel·lació per evitar defectes en les mescles i, gràcies a aquest projecte, es podrà reduir el malbaratament dels productes de les plantes.

10 Possibles millores a futur

A continuació es fa un recull de possibles canvis que podrien millorar el producte:

- Modificar el codi Python de client i servidor per enviar un missatge per correu (utilitzant la llibreria ja creada) quan es produeix un error inesperat. D'aquesta manera, es podria reduir el temps de recuperació, ja que en l'actualitat la supervisió del funcionament del sistema és manual.

- Millorar certs aspectes de seguretat del sistema. Tot i que el sistema està pensat per funcionar dins de la xarxa privada de Chemplate, existeix certa informació sensible que pot ser robada per usuaris maliciosos. Ocultar aquesta informació i proveir als usuaris amb comptes sense privilegis d'administrador suposaria una disminució de riscs potencials.

- Canviar la plataforma utilitzada dels clients. Tot i que la Raspberry Pi 4B és un excel·lent microcontrolador, el seu cost actual és relativament alt. L'ús de processador i Memòria RAM del Software dels clients és del voltant del 15 % i 40% respectivament.

  Alternatives com la Raspberry Pi Zero 2W es podrien tenir en compte, tot i que seria necessària una optimització del codi.

- Crear una carcassa 3D que s'adeqüi a la perfecció a les necessitats del projecte. Durant la realització d'aquest s'ha valorat la possibilitat d'afegir pantalles OLED per mostrar l'estat del sistema però s'ha descartat la idea per manca de temps i per evitar tenir la pantalla totalment desprotegida.

  Una carcassa impressa amb impressora 3D que disposés d'un suport per a pantalles seria una bona addició.

- Crear una pàgina web on es puguin realitzar consultes utilitzant les funcions Modbus ja implementades. El producte actual només permet fer lectures dels registres del sensor i guardar-les directament a la base de dades. Permetre a l'usuari fer consultes in-situ a través

d'un formulari web ajudaria a configurar i diagnosticar el sistema.

- Proveïr als usuaris no experts d'una formació específica per ensenyar una bona configuració i ús del sistema.
- Millorar l'actuació davant dels canvis en les mescles. En el disseny actual només es té configurada una alarma que engega un motor. Fora bo posar en pràctica el sistema amb la bomba peristàltica ajudaria a demostrar l'eficàcia del prototip creat.
- Implementar el sistema damunt de clients VPN per permetre utilitzar el sistema en plantes pilot situades fora de la xarxa interna de Chemplate.
- Unificar els arxius de configuració del sistema, ja que en l'actualitat existeixen un arxiu de configuració per Raspberry i un per Docker.

AGRAÏMENTS

En primer lloc voldria agrair al meu tutor, Màrius Montón Macian la seva col·laboració i orientació durant aquest treball, així com donar-me la oportunitat de participar en el seu projecte, proporcionant-me les eines a través de les quals he pogut desenvolupar un prototip funcional. També 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, 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.

REFERÈNCIES

[ED] EDSRobotics: ¿Qué es la automatización industrial? [Online] Available: *https://www.edsrobotics.com/blog/que-es-la-automatizacion-industrial/*

[AC] Acesa: HMI,ACE, Automatisme i Control Elèctric [Online] Available: *https://www.acesa.es/cs/p1/hmi*

[BE] Santander: Metodologías de desarrollo de software: ¿qué son? [Online] Available: *https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html*

[RE] RedHat: ¿Qué es la automatización? Ventajas e importancia de automatizar. [Online] Available: *https://www.redhat.com/es/topics/automation*

[RA] Raspberry Pi. [Online] Available: *https://www.raspberrypi.com/*

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

[EC] Eclipse Mosquitto™ An open source MQTT broker [Online] Available: *www.mosquitto.org*

[IN] Influxdata [Online] Available: *https://www.influxdata.com/*

[GR] Grafana: The open observability platform — Grafana Labs [Online] Available: *https://grafana.com/*

[NO] Node-RED: Low-code programming for event-driven applications [Online] Available: *https://nodered.org/*

[MO] The Modbus Organization [Online] Available: *https://www.modbus.org/*

[MQ] MQTT: The Standard for IoT Messaging [Online] Available: *https://mqtt.org/*

[DO] Docker [Online] Available: *https://www.docker.com/*

[AP] APD: ¿En qué consiste la metodología Kanban y cómo utilizarla? [Online] Available: *https://www.apd.es/metodologia-kanban/*

[TR] Trello [Online] Available: *https://trello.com/*

[FL] Flux vs InfluxQL [Online] Available: *https://docs.influxdata.com/influxdb/v1.8/flux/flux-vs-influxql/*

[GI] Github: IzakMarais/reporter [Online] Available: *https://github.com/IzakMarais*

[PI] PiCockPit: Monitor and Control your Raspberry Pi [Online] Available: *https://picockpit.com/*

[CH] Chemitec [Online] Available: *https://www.chemitec.it/en/50-series*