Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Albert Joan Santacana Gabella

Titulació: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol de Treball Final de Grau: **INTERFÍCIE DE COMUNICACIÓ ENTRE UN MODEL DE XARXA NEURONAL LÒGIC PREDICTIU I UN PLC EN UN PROTOTIPUS REAL**

Director/a: **Fernando Guirado Fernandéz/ Cristian Solé Cutrona**

Presentació

Mes: Setembre

Any: 2024

ÍNDEX

1. INTRODUCCIÓ

En l'era de la nova digitalització industrial, la millora i optimització dels sistemes de control, la comunicació cada dia més eficient entre mòduls i l' automatització estan a l'ordre del dia. El PLC o controlador lògic programable és la part computacional central del procés d'automatització.

Aquest TFG troba els seus inicis en un treball d'investigació d'un nou prototipus real estudiat per la xarxa d'investigadors del SEMB(Sustainability in Energy, Machinery and Buildings) de la UdL. L'objectiu d'aquest prototipus és fonamentat en la captació d'energia solar durant el dia i el refredament natural durant la nit per mitjà del fenomen conegut com radiative cooling a distància.

Aquest prototipus està governat per un PLC el qual s'encarrega del control electromecànic dels diferents actuadors en base a una lògica programada i de les lectures dels diferents sensors juntament a un SCADA per a que l'usuari pugui modificar, visualitzar i realitzar un control del prototipus.

La programació del PLC i el SCADA es realitza des de l'entorn de desenvolupament TIA Portal de Siemens que permet crear la lògica de control a més de la interfície d'usuari podent així tindre un entorn on treballar tots els elements del projecte.

Actualment, aquest prototipus funciona baix dos tipus diferents d'operatives, manual o automàtica, malgrat estar programades en funció dels paràmetres que es llegeixen dels sensors, el circuit i la coberta que es posa durant el dia i es treu durant la nit, manca d'un comportament que permeti predir i actuar de forma automàtica decisions encarades a l'eficiència energètica i l'optimització del comportament de l' instal·lació en base a la situació actual en conjunt amb la contemplació de futurs casos en un marge de temps determinat.

Per assegurar aquest comportament és essencial l' implementació d'una interfície que comuniqui una lògica de control intel·ligent que prengui decisions de forma lògic predictiva amb el controlador (PLC) del prototipus real.

2. PROTOTIPUS RCE

El prototipus parteix d'una idea formulada en un article científic creat per l' equip d'investigació SEMB[1] de la UdL, el qual a traves d'un "proof of concept" corrobora mitjançant l'eficiència pic que es produeix en el RCE (49% durant el dia i 32% durant la nit) la demostrabilitat de l'efecte positiu que aquest pot tindre en la combinació del efecte radiative cooling nocturn i la captació energètica solar diürna.

Com s'esmenta en un altre article científic de ScienceDirect[2] la transferència de calor per radiació és un dels mètodes naturals de transport d'energia més emprats. Els objectes terrestres dissipen calor cap a l'espai exterior en forma d'ones electromagnètiques, aquest fenomen s'anomena "radiative cooling" o refredament radiatiu el qual es pot aprofitar per a conformar sistemes de refredament natural sense cap font d'excitació energètica d'entrada.

El RCE aprofita aquest efecte i prova que és factible en una temporada on les condicions de dissipació de calor són mínimes, demostrant així, que pot utilitzar el fred com a font de refredament directa en altres tecnologies de refrigeració mitjançant l'ús d'energies renovables i en condicions adverses. Com es pot observar en la Figura 1, es pot trobar el diagrama de l' instal·lació que visualitza les parts més importants d' actuadors i de tancs.

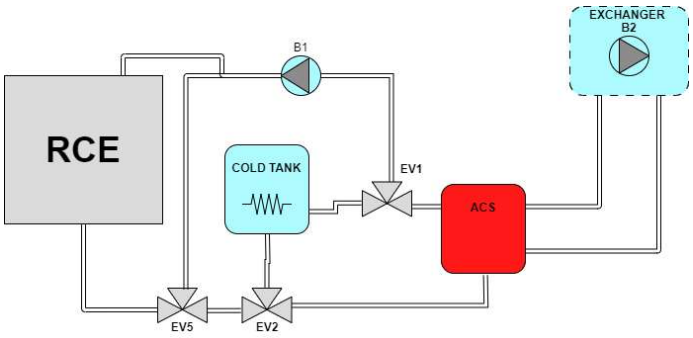

Figura 1.Diagrama general de l'instal•lació per a la visualització de les parts importants i el funcionament del circuit.

Aquest consta físicament d'un tanc d'aigua freda (COLD TANK), un tanc d'aigua calenta (ACS), un col·lector solar amb una coberta mòbil (RCE) i un bescanviador (EXCHANGER). Durant el dia es calenta l'aigua que hi ha en el tanc calent, el qual es troba a temperatura ambient. Durant la nit, es refreda l'aigua que es troba al tanc fred. En la Taula1 es pot observar la funció de cada part del circuit i la distribució d'actuació de dia i de nit.

Taula 1. Diferents modes dels actuadors i parts del circuit en funció de la zona de les 24h diàries.

ACTUADOR	ESTATS	DIA	NIT
Coberta vidre RCE	Sí/No	Sí	No
Bomba B1	ON/OFF	ON	ON
Bomba B2	ON/OFF	OFF	ON
EV1	Solar Mode/ Cold Mode	Solar Mode	Cold Mode
EV2	Solar Mode/ Cold Mode	Solar Mode	Cold Mode
EV5	Bypass ON/OFF	Depèn	Depèn
Cold Tank	Refredament de l'aigua	No s'utilitza	Refredament de l'aigua
ACS	Escalfament de l'aigua	Escalfament de l'aigua	No s'utilitza

Al llarg del circuit es poden trobar diferents sensors que capten variables com la temperatura, la radiació o el vent que en aquella situació concreta hi ha, la EV5 que regula el cabal que passa pel RCE en funció de que es vulgui o altres elements agregats com el ventilador del bescanviador.

3. OBJECTIU

L'objectiu que persegueix el treball és implementar una interfície que permeti, mitjançant l'ús de diferents tècniques i protocols de comunicació, la transmissió efectiva de dades entre el PLC i una lògica de control intel·ligent, afegint així, un comportament predictiu,

una millora i una optimització de la resposta que ofereix l' instal·lació davant qualsevol possible situació meteorològica.

4. ESTRUCTURA DEL TREBALL

La memòria d'aquest treball final de grau consta de quatre parts que estan ben estructurades :

1) Marc Teòric. Introdueix les diferents tecnologies utilitzades per a la realització de l'objectiu, afegint així, el marc contextual en el que el treball es desenvolupa.
2) Metodologia. Es dona en detall de l' implementació de la solució que es dona per assolir l'objectiu proposat.
3) Resultats. Es presenten els resultats i el seu posterior anàlisi en blocs.
4) Conclusions. Es discuteixen les conclusions i les propostes de millora segons escalabilitat i/o altres paràmetres d'estudi que puguin ser imprescindibles.

5. PLANIFICACIÓ I ASSOLIMENT DE TASQUES

En un projecte com aquest sempre es bo tenir una diagrama que permeti definir la constància del seguiment setmana rere setmana que s'ha portat a terme. Per això, s'inclou un diagrama de Gantt amb l'objectiu d'observar la continuació des de la fase d'inici passant per la fase de desenvolupament i finalitzant amb les proves i validació de l'aplicació de forma aproximada, el qual es pot observar en la Figura 2.

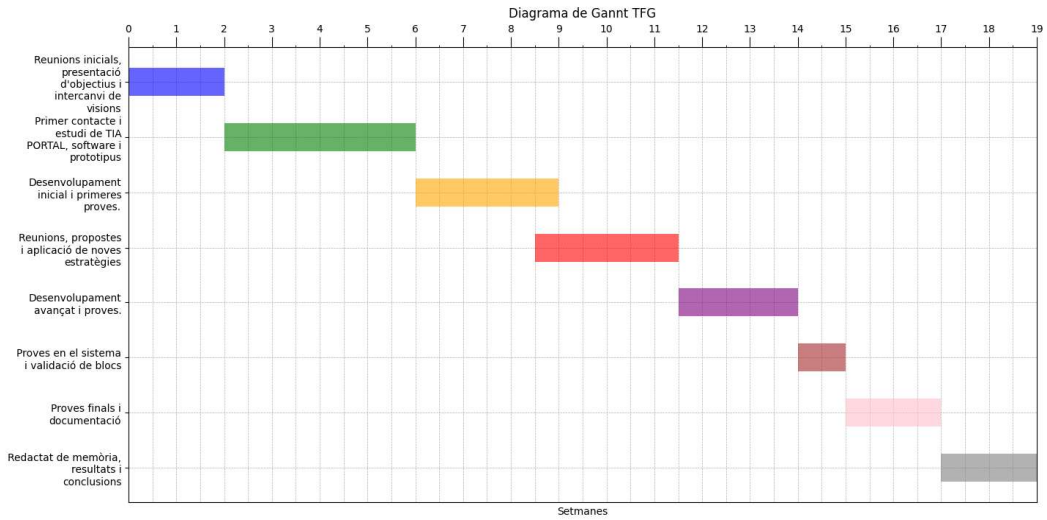

Figura 2. Diagrama de Gantt del TFG des de la fase d'inici, desenvolupament, execució i final aproximat.

Aquesta aproximació de temps per setmanes es dona ja que no hi ha unes pautes temporals definides donada la variabilitat de la consecució dels objectius marcats. En el transcurs del projecte en conseqüència a la complexitat i temps d'anàlisi requerit per a trobar una resposta adequada al sistema(a vegades canviant i amb necessitat de nous temps d'estudi) s'ha seguit una planificació no lineal i s'ha requerit d'una aproximació.

6. ACRÒNIMS I SIGLES

SCADA: Supervisory Control and Data Adquisition

Sistema de Control i Adquisició de Dades que visualitza les entrades i sortides del sistema permetent a l'usuari el control des d'una pantalla.

PLC: Programmable Logic Controller

Controlador de la lògica de recepció de dades de sensors i l'enviament de senyals de control als diferents actuadors programable.

TIA PORTAL: Totally Integrated Automation Portal

Software de Siemens per a l'accés, implementació i control dels diferents sistemes que envolten el PLC i la comunicació amb els seus diferents perifèrics, el qual permet la integració en un entorn de totes les parts del projecte d'automatització.

HMI: Human-Machine Interface

En TIA PORTAL quan es pretén incloure un SCADA es diu que, de forma general, es una interfície entre l'operari extern i la màquina entesa com el PLC.

7. MARC TEÒRIC

7.1. PLC

Un PLC (Programmable Logic Controller) és un dispositiu electrònic utilitzat en el món de la indústria, específicament de l'automatització, per al control i actuació prèviament programada de les senyals d'actuadors en base a la lectura d'unes senyals d'entrada.

Com es pot observar en la Figura 3 aquests poden rebre diferents tipus de senyals de sensors, emmagatzemar en memòria interna variables, i de la mateixa forma, exercir de centre de distribució i transmissió de senyals, informació a SCADA, bombes, electrovàlvules...

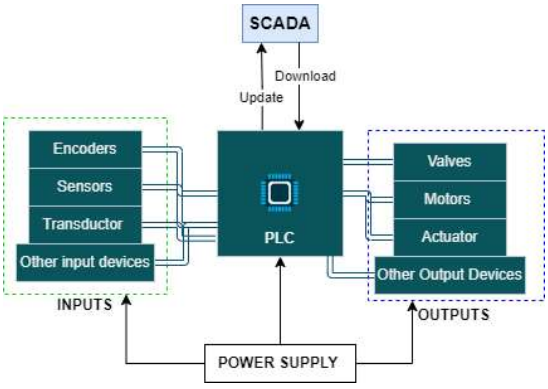

Figura 3.Diagrama il·lustratiu de la distribució general d'entrades, sortides i altres elements amb els que es comunica un PLC

Un PLC es compon generalment de:

1. CPU: on s'executa la lògica del PLC prèviament programada.
2. Memòria: lloc d'emmagatzemament de variables internes, temporals, numèriques...
3. Mòduls I/O: on es connecta el PLC amb els diferents dispositius d'entrada i sortida.
4. Font d'alimentació: Proporciona la energia al PLC per al seu correcte funcionament.
5. Mòduls de comunicació: permeten la connexió del PLC amb altres dispositius o sistemes .

El PLC funciona d'acord a un cicle d'operació en el qual es completa un procés seqüencial precís en el que es llegeixen les entrades, s'executa la lògica de processament i s'actualitzen les sortides d'acord a aquesta lògica. De forma recurrent, i per assegurar el bon funcionament mantenint les bones pràctiques, es realitzen sessions de manteniment i prevenció.

Les entrades i les sortides poden ser digitals o analògiques, i representatives de qualsevol tipus de procés automatitzable. Per exemple, en el circuit, els diferents sensors que s'utilitzen per a la mesura de temperatura PT100 són analògics o en el cas de les electrovàlvules , específicament en les senyals digitals suficients per a engegar-les o apagar-les.

Per programar la lògica operativa desitjada en una estructura en que TIA Portal pugui compilar i transmetre al PLC hi ha diversos llenguatges entre els quals es pot destacar:

1. Ladder Diagram(LD): és una tipologia representativa similar a un esquema elèctric els elements gràfics dels quals es basen en contactes i blocs.
2. Function Block Diagram(FBD): utilitza blocs funcionals connectats entre si per poder representar el flux de senyals i dades.
3. Structured Text(ST): llenguatge d'alt nivell que s'utilitza per a rutines i algoritmes complexes.
4. Statement List(STL): ofereix molta precisió a l'hora d'interactuar amb el hardware, poden manipular bits i bytes, és eficient i és adequat per a tasques complexes.
5. Instruction List(IL): és el llenguatge amb el que es programen cada bloc de codi del PLC. Ofereix una estructura similar al llenguatge de baix nivell assemblador amb una gran rapidesa d' execució i tendeix a ocupar menys memòria.

En el projecte de TIA PORTAL pertinent al maneig del PLC els blocs estan codificats en IL.

7.1.1. PROTOCOLS DE COMUNICACIÓ PC-PLC

Els PLC, tal com s'ha mencionat abans, es comuniquen amb sistemes SCADA que li permeten a l'usuari mantindre un control de l' instal·lació des d'una pantalla industrial o, en el cas pertinent, un PC. Per aquest propòsit s'utilitzen protocols de comunicació imprescindibles per a poder passar la informació via Ethernet, que a la vegada compleixin els requeriments que el sistema demanda. En aquest nínxol de mercat es poden assenyalar

protocols com per exemple Modbus TCP, POWERLINK, EtherCAT... tot i la gran quantitat d'aquests, el TFG es centrarà en explicar Profinet, el qual és el que hi ha a l' instal·lació.

PROTOCOL PROFINET

El protocol PROFINET o Process Field Network, és la forma de comunicació estàndard a nivell global per a l'automatització industrial. Com s'observa en la Figura 4 una xarxa PROFINET pot combinar múltiples estacions de treball, que van de I/O, fins actuadors, PLC o SCADA en la mateixa línea amb una alta rapidesa, facilitat d'instal·lació i temps mínim de posat en marxa.

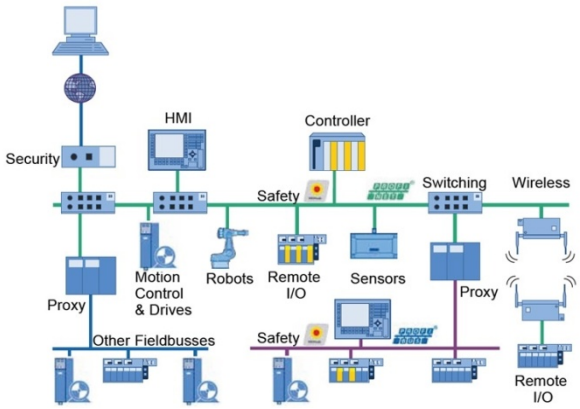

Figura 4. Exemple de protocol de comunicació PROFINET en múltiples estacions on en la part verda es visualitza la connexió.

S'adreça als requeriments de planta tant com als de la part de IT o d'oficina a través dels seus tres canals de comunicació que aborden el conjunt de protocols TCP/IP estàndard, la transmissió de dades en temps real i un canal per a la sincronització altament precisa en un rang de temps baixos el qual requereix d'un hardware addicional[3]. En l' instal·lació s'aprofita aquest seguit de avantatges esmentades per a poder establir una comunicació entre el PLC i el PC.

7.2. XARXES NEURONALS I MACHINE LEARNING

Tal com en la introducció s'ha mencionat es necessita una interfície de control intel·ligent que prengui les decisions envers a una lògica predictiva, aquesta es tracta d'una xarxa neuronal. Tot i que el TFG es centri en la comunicació més que en aquest tipus de tecnologia i algorítmica, en els següents apartats s'explicarà què són les xarxes neuronals, quin funcionament tenen les principals per a la gestió dels diferents tipus d'informació

entrant i quins algoritmes o el com es manegen per a que el lector comprengui a quin tipus de lògica s'està comunicant la decisió.

7.2.1. QUÈ SÓN LES XARXES NEURONALS?

Les xarxes neuronals, tal com el seu nom indica, són models computacionals que han revolucionat àrees de la ciència i la tecnologia basades conceptualment en les funcionalitats biològiques que exerceix el sistema nerviós humà.

S'han convertit en eines de tractament i resolució afinada de problemes complexos que tenen la capacitat d'adaptació a noves entrades, aquesta és essencial en el retorn de la resposta a l' interfície. Les unitats de processament o neurones artificials s'organitzen en capes en les quals sol haver-hi tres tipus de classe/s de capa: d'entrada, de sortida i l'invisible/oculta. Les unitats es connecten amb forces de connexió variables o ponderacions i la millora de predicció evoluciona amb la reiteració d'errors.

7.2.2. XARXES NEURONALS FEEDFORWARD

Les neurones mantenen una organització de connexions seqüencial, per tant, es va de l' inici de la capa d'entrada a la de sortida de forma lineal. Les seves aplicacions solen estar en processament d'imatges, reconeixement de patrons, predicció de dades.

7.2.3. XARXES NEURONALS PERCEPTRON MULTICAPA

Tipus de xarxa neuronal amb múltiples capes. És interessant la capacitat d'aprenentatge de funcions no lineals que aquestes tenen. Es solen aplicar per exemple en problemes de classificació , reconeixement de veu i detecció de frau. Com la Figura 5 reflexa, aquestes basen el seu funcionament en l'algoritme de Backpropagation el qual ajuda a incrementar la precisió de la sortida reduint l'error entre la sortida predita i sortida real[4].

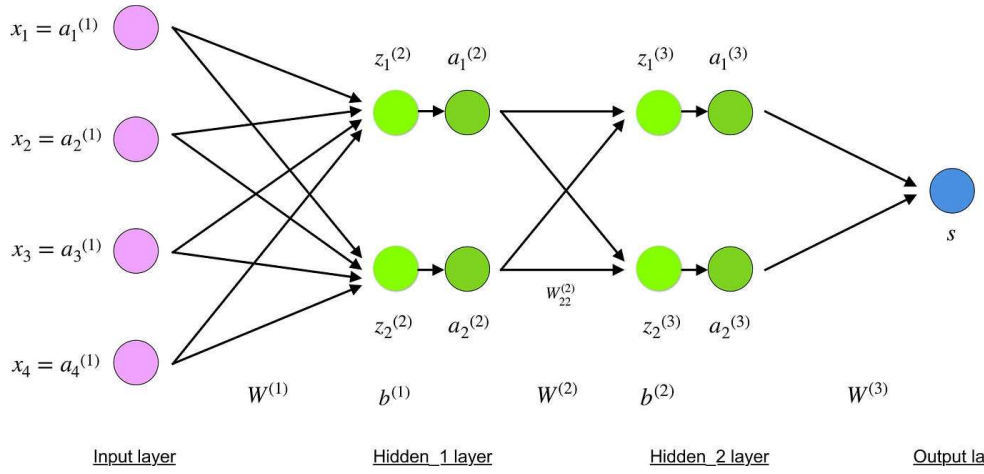

Figura 5. Capes d'estructuració i variables en l'algoritme de Backpropagation aplicat en una xarxa neuronal Perceptron Multicapa

7.2.4. XARXES NEURONALS CONVOLUCIONALS

Les xarxes neuronals convolucionals contenen tres tipus de capes: la capa convolucionar, la capa d'agrupació(polling layer) i la capa completament conectada(fully-connected layer). Aquestes tres es poden visualitzar de forma clara en la Figura 6, exemple que utilitza l'entrada d'una imatge per a generar una sortida que reconeix el tipus d'animal que és.

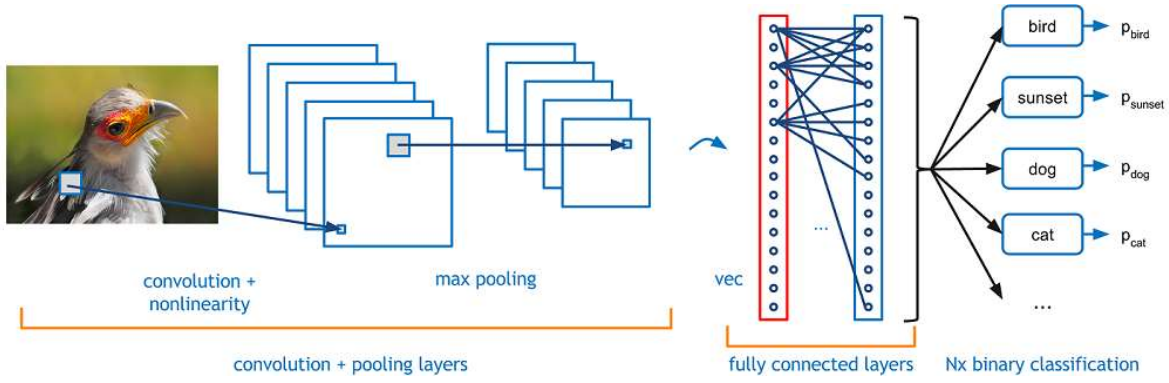

Figura 6. Diagrama de funcionament d'una CNN

La capa convolucionar, o la convolució, és l'aplicació iterativa d'una funció que es mou per les diferents regions de la imatge d'entrada aplicant filtres de la mateix mida que reconeixen un patró específic de la imatge. La capa d'agrupació, redueix la mida del mapa d'informació que es rep aplicant filtres de nou. Aquesta deixa caracteritzada de forma matricial cada característica de la imatge.

Seguidament, les capes completament connectades basen la seva aplicació en la reducció a paràmetres unidimensionals del resultat de la capa d'agrupació i l'aplicació d'unes combinacions lineals seguit d'unes funcions d'activació[5]. Finalment, el valor de sortida és el que ofereix el màxim rang de possibilitats.

7.2.5. XARXES NEURONALS RECURRENTS

Les xarxes neuronals recurrents són aquelles que utilitzen dades seqüencials o series de dades temporals[6] , de manera que són una opció tàcita per la traducció de llenguatge, el reconeixement de veu o la subtitulació. La diferència entre aquest tipus i altres, és que reutilitzen les entrades o sortides prèvies per a alimentar i influenciar les noves entrades o sortides, cosa que es pot veure en la Figura 7.

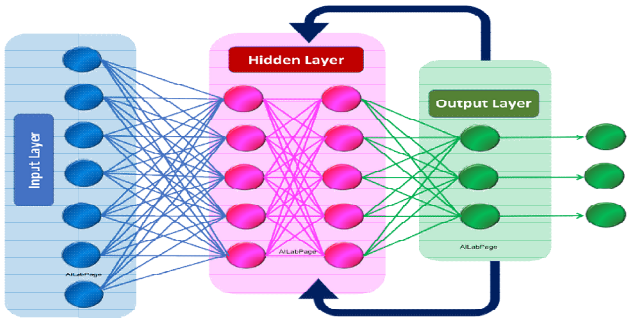

Figura 7. Diagrama de funcionament de les RNN

7.3. PROPOSTA D'IMPLEMENTACIÓ

En el desenvolupament de l' interfície s'han trobat diferents reptes en la manera en que s' aplica l' integració de diferents tècniques de caire informàtic, ús de diferents protocols i comunicació efectiva. Uns dels factors més remarcables a l'hora d'enfrontar aquest procés de desenvolupament podrien ser:

- La complexitat de la Tasca
  - Anàlisi de requisits.
  - Disseny del sistema: l'estructuració del software per a la gestió de tasques complexes de manera eficient i d'una manera òptima.
- Capacitat interoperativa.
  - La comunicació eficient amb altres sistemes i l'experiència d'interfície d'usuari.
- Correcta optimització de la gestió de recursos i del codi.
- Escalabilitat del projecte.

- El disseny modular ajuda a entendre el projecte i fragmentar en parts que ajudaran a que aquest sigui escalable.
- Ha de permetre actualitzacions i millores constants sense interrompre el funcionament del sistema.

7.3.1. LLENGUATGE DE PROGRAMACIÓ PYTHON

Un dels primers passos alhora d' implementar una interfície és escollir el llenguatge en la que aquesta es programa. És important considerar l'elecció en base a l'escalabilitat, flexibilitat i l'adaptabilitat del protocol al llenguatge. Les 4 propostes de llenguatge eren les que suportava el mateix protocol que es veurà i justificarà snap7. Aquests eren C/C++, Python, Pascal i C#.

C/C++ : llenguatges amb complexitat elevada i un alt rendiment en aplicacions en temps real. Poden ser molt dependents del sistema operatiu i del hardware que s'utilitza, i si la gestió de memòria en bits o bytes ja la fa el protocol snap7 no són tant justificables en compensació a la complexitat requerida.

Python : llenguatge d'alt nivell amigable amb una semàntica dinàmica integrada, i un rendiment mig en aplicacions de temps real. Suporta l'ús de mòduls i paquets i es de propòsit general. Adaptabilitat perfecta a snap7 i a l' idea d'interfície que es vol.

Pascal : llenguatge d'alt nivell amb una sintaxis clara, ben organitzat i eficaç a nivell de velocitats d'execució i ús eficient de recursos. No té una biblioteca snap7 tan ben suportada com en el cas de C/C++ o Python i no s'utilitza tant en l'entorn industrial modern.

C# : llenguatge amb sintaxis moderna i fàcil d'entendre. És menys eficient que C/C++, requereix d' instal·lació de .NET, mida i gestió de recursos no tan eficient i està molt subjecte al hardware i sistema operatiu de partida.

Per tant, la tria s'ha fet en una vista al present en termes de: adaptabilitat, relació complexitat – integració de protocol i compatibilitat amb altres sistemes. També s'ha mirat al futur en termes de: escalabilitat, sintaxis clara, programació funcional i disseny que no involucri per l'usuari més del requerit.

Aquest llenguatge és de propòsit general i existeixen moltes llibreries de tercers tal com una comunitat molt gran darrere. La interfície es programa en Python i a continuació,

s'expliquen les diferents metodologies informàtiques utilitzades per arribar a assolir la seva implementació.

7.3.2. BLOCS ALGORÍTMICS

7.3.2.1. SINCRONICITAT COMUNICATIVA

Un dels aspectes fonamentals a l'hora de dissenyar una interfície que es comuniqui amb altres softwares es la manera en que gestiona les sol·licituds. Elegir el sincronisme amb la que es gestionen els recursos ajudarà a l'execució eficient. Per tant, podem separar entre:

- La programació síncrona. És seqüencial, significa que espera a que la primera o una tasca s'acabi per a començar a executar la següent com es pot veure en la Figura 8.

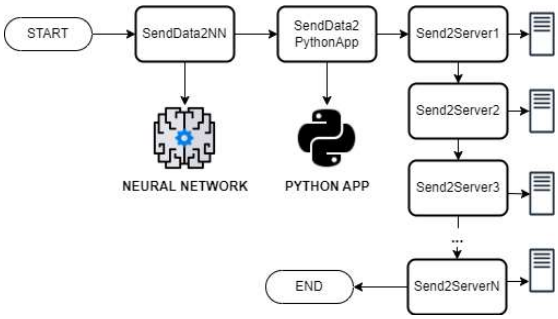

Figura 8. Diagrama de flux d'algorítmica d'enviament de dades seqüencial a diferents servidors.

Existeixen dos tipus de programació asíncrona: la concurrència i el paral·lelisme. En la Figura 9 la principal diferència en relació execució de tasques – recursos utilitzats.

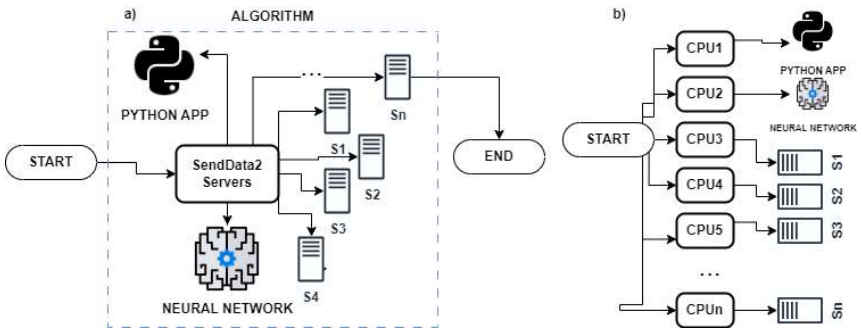

Figura 9. a) Diagrama de flux d'enviament de dades a servidors de manera concurrent b) Diagrama de flux d'enviament de dades a diferents servidors per mitjà del paral·lelisme

Les principals diferències entre ambdós són:

- La concurrència, es comunica de forma paral·lela d'un mateix client a múltiples servidors amb sol l'ús d'una unitat de processament[7], per tant

l'opció elegida. No hi ha simultaneïtat, de forma que les instàncies es produeixen en línies de temps diferents.

- El paral·lelisme, estableix a través de múltiples unitats de processament a cada servidor la comunicació adient.

7.3.2.2. COMUNICACIÓ BASADA EN SOLICITUDS

La comunicació http(protocol de comunicació d'hipertext) és un conjunt de regles atribuïdes a la tipologia client servidor importants en la programació de l' interfície que ens permet mitjançant un seguit de sol·licituds i respostes establir una comunicació entre un client i un servidor web, o inclusive, en un servidor local.

Si es vol fer una sol·licitud de lectura el client envia un HTTP GET, a l'inversa, si es pretén enviar informació s'utilitza el HTTP POST. D'aquesta manera es fan els servidors que ajuden a enviar una resposta de nou a l' interfície dels quals se'n parlaran a les següents seccions.

El servidor HTTP té unes tres tipologies de respostes per definició, aquestes són protocol·làriament importants per saber si una sol·licitud s'ha executat i interpretat amb èxit:

1) 200-OK(Sol·licitud acceptada)
2) 400-Bad request(Sol·licitud incorrecta)
3) 404-Resources not found(Recursos no trobats)

En el context de l' interfície implementada s'utilitza AIOHTTP el qual un framework específic de client/servidor HTTP asíncron que ens permet la comunicació no bloquejant entre múltiples servidors en temps real. Aquesta és essencial per a la comunicació en la qual l' interfície envia les dades en els corresponents servidors o punts de comunicació.

7.4. INTERFÍCIES GRÀFIQUES

Les interfícies gràfiques són els mitjans pels quals l'usuari té el poder d'interactuar, gestionar i visualitzar per mitjà de widgets informàtics com per exemple botons, sliders, caixes de text...un cert tipus de lògica programada.

En la creació d'una interfície, és important tindre en compte la facilitat d'ús i l'experiència interactiva que aquesta ofereix a l'usuari tant com la relació que aquesta tingui amb les dades i els programes que s'executen.

En l' interfície dissenyada s'ha utilitzat una llibreria la qual s'anomena tkinter, específicament una extensió d'aquesta que es diu tkinterbootstrap. Aquesta extensió ofereix una experiència visual més agradable i moderna per a l'usuari.

7.5. TIA PORTAL

Un dels grans blocs d'aquest TFG es centra en el software que ofereix la possibilitat de programar l'entorn de comunicació entre el SCADA i el PLC. TIA Portal és un software que permet enllaçar els PLC de la marca SIEMENS amb diferents aplicacions, SCADA, clients i altres alternatives de comunicació. Aquest interactua mitjançant el tipus de protocol que és seleccionat per l'usuari entre una diversa llista com Profinet, Profibus o Modbus entre d'altres intercanviant les dades i l' informació necessària sempre baix la lògica de programació que s'hagi estipulat.

7.5.1. WINCC

És l'eina de visualització, control de processos i supervisió de SIEMENS integrada dins l'entorn de TIA PORTAL el qual proporciona un entorn gràfic intuïtiu on l'usuari pot crear d'una forma interactiva interfícies d'usuari. És compatible amb una gran varietat de dispositius i hardware de Siemens. La versió utilitzada específicament en el projecte és SIMATIC WinCC Runtime Advanced el qual entre d'altres característiques permet:

- Solució de control i monitorització basat en PC.
- Pack bàsic per a la visualització i generació d'informes.
- Pot integrar-se en solucions basades en TCP/IP.

7.5.2. PRINCIPALS PROTOCOLS SUPORTATS

L'elecció del protocol de comunicació informàtics entre els diversos que en aquests apartats es presenten és difícil donada l'àmplia varietat de maneres que ofereix Siemens de comunicar TIA Portal. El protocol elegit és snap7, en els següents apartats s'explicaran les diverses opcions que existeixen i s'establirà una comparativa entre els quatre, justificant així la tria.

7.5.2.1. SNAP7

La suite de comunicació multi plataforma Ethernet que permet crear una interfície nativament amb els PLC de Siemens S7, i per tant, la seleccionada per l'aplicació Python

amb TIA Portal, és snap7. Aquesta permet el control i l' integració de control a PLC des del protocol S7.

Les funcions que snap7, com és el cas, pot fer com a client són:

- Connexió al PLC.
- Lectura i escriptura de dades a àrees de memòria específiques del PLC, com per exemple, les Data Blocks o les etiquetes del PLC que corresponen a entrades, sortides o memòria interna.
- Diagnòstic d'estats i errors en el PLC.

7.5.2.2. OPC UA SERVER

Com estàndard de comunicació independent orientat a serveis, OPC UA, proporciona una arquitectura unificada per la comunicació i l' intercanvi segur d'informació. Per a poder realitzar aquesta comunicació és necessària una configuració prèvia, aquesta es basa en l'ajust dels certificats que utilitzen TIA PORTAL i OPC UA per donar una capa de seguritat extra a la transmissió d'informació.

Com a servidor, pot rebre múltiples peticions de clients els quals poden ser aplicacions externes, SCADA... i generar una resposta que el PLC pot executar.

7.5.2.3. MQTT / HTTP

Aquests protocols són àmpliament utilitzats en la comunicació d'entorns en l' Internet de les Coses. El protocol de missatgeria MQTT(Message Queueing Transport Telemetry) esta dissenyat per entorns amb una alta latència i un ample de banda limitat[8] ideal per a sistemes amb recursos restringits que necessiten una transmissió de dades.

Com s'ha explicat abans HTTP està dissenyat per a la comunicació i integració de plataformes web o inclusive servidors locals els quals sol necessitin una xarxa intercomunicativa entre sistemes.

Es necessita tindre integrades certes biblioteques i blocs de funcions instal·lades per a poder exercir aquests tipus de comunicació en TIA Portal, les biblioteques LQMTT i LHTTP respectivament.

7.5.2.4. COMPARATIVA

En aquest apartat es justificarà la decisió que s'ha pres en funció de quatre paràmetres, ús per al qual va destinat, comunicació utilitzada, complexitat i escalabilitat.

La complexitat pot ésser alta, moderada o baixa-moderada, essent "alta": una gran quantitat d'esforços, temps dedicat i una gran densitat de compartició de recursos necessaris. Per lo contrari "baixa-moderada" significa una baixa exigència al respecte de compartició de recursos, temps dedicat per assimilació i un esforç moderat. En la Taula 2 es pot veure de forma més clara la comparació:

Taula 2. Comparativa d'ús, comunicació, complexitat i escalabilitat de les quatre opcions de comunicació més vàlides per a la gestió de la solució en TIA Portal.

PROTOCOL	ÚS	COMUNICACIÓ	COMPLEXITAT	ESCALABILITAT
SNAP7	Comunicació amb PLC Siemens S7	Accés directe a àrees de memòria del PLC	Moderada: segons l'ús	Molt escalable en aplicacions d'automatització industrial
OPC UA SERVER	Servidor entremig de sistemes informàtics i traspàs de dades	Comunica SCADA, aplicacions externes amb el PLC i exposa les dades en nodes per a la seva utilització externa si l' usuari o desitja.	Alta	Escalabilitat encarada a la comunicació múltiple de diferents sistemes
MQTT	Comunicació no periòdica màquina a màquina(M2M) en un espai amb limitacions físiques.	Es basa en el mètode publicació subscripció en el que un client publica una informació que la filtra i la distribueix adequadament el agent i un altre client es subscriu al tema per a rebre la informació d'aquest.	Moderada	Altament escalable per aplicacions pròpies en condicions de comunicació precàries
HTTP	Intercanvi de dades entre clients i servidors webs	Mètode client servidor.	Baixa - moderada	Moderadament escalable per a servidors web

L'elecció proposada per a l' interfície es basa en snap7 no sol per la seva relació complexitat - escalabilitat si no també per la seva capacitat, a diferència dels altres

protocols o mètodes de comunicació entre TIA Portal i una aplicació externa, de llegir o escriure a nivell baix, en les àrees de memòria en les que hi ha capes de bytes i de bits dels Data Blocks, Function Blocks o Functions que es descriuran en els següents apartats.

7.5.3. ESTRUCTURES I BLOCS DE DADES DE TIA PORTAL

L' importància de les estructures i correlacions que les dades mantenen en TIA PORTAL ha esmentat crucial per assolir una solució òptima i adaptativa. Entendre aquesta comunicació, permet emprar l'entorn en la flexibilitat i la solidesa demandades juntament amb el protocol establert.

7.5.3.1. ORGANIZATION BLOCK

Els blocs d'organització són la capa més pròxima entre el programa de l'usuari i el mateix sistema operatiu i són els que defineixen l'estructura bàsica del programa[9], aquests apareixen en TIA PORTAL tal com es veu en la Figura 10. Es poden tindre varies opcions al respecte de l'ús d'un determinat OB, entre d'altres l'execució cíclica, la posada en marxa o el bloc de diagnòstic d'errors.

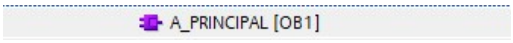

Figura 10.Visió d'un OB en TIA PORTAL.

En la majoria de casos, salvant excepcions, s'utilitzen per cridar a blocs de funcions (FB) o funcions(FC), aquestes FB a la vegada s'associen a les Data Blocks(DB) per emmagatzemar i cridar variables quan sigui necessari, en la Figura 11 podem observar la manera en què les OB criden als diferents blocs o funcions.

Figura 11. Crida de programa des d'un organization block(OB)

7.5.3.2. FUNCTION BLOCK

Els blocs de funció, són parts que emmagatzemen el codi principal del programa, i utilitzen estructures de crida, condicionals, de carga i de transferència, com a sistema de programació d'objectes, en aquest cas la Figura 12 n'és un exemple. Aquestes, a diferència de les funcions, si contenen memòria emmagatzemada específica.

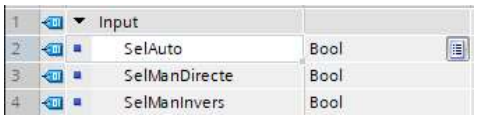

Figura 12. Visió de varies variables d'entrada d'un Function Block

7.5.3.3. DATA BLOCK

Els blocs de dades d' emmagatzematge, es visualitzen en TIA PORTAL com es mostra en la Figura 13, es poden separar en:

- Global DBs. Memòria que pot ser cridada en qualsevol part del programa
- Instance DBs. Memòria que es genera per l'associació d'un Function Block a una àrea de memòria específica(DB assignada de forma manual o automàtica).

Figura 13.DataBlock d'exemple

7.5.3.4. TAGS

Una etiqueta o tag, es un identificador assignat a una variable, de manera que, es pot associar a una acció del SCADA , podem enviar aquesta variable o associar aquesta a una data block(DB), una entrada(I:Input) , una sortida(O:sortides) o com a memòria de programa(M: memòria interna). Aquests ajuden a l'usuari en l'elecció de si es vol que s'utilitzi un mode en que la xarxa neuronal operi la lògica de circuit o si pel contrari es vol fer un ús més manual.

8. METODOLOGIA

8.1. REPTES I DIFICULTATS

Per explicar quina metodologia cal tindre present quin és l'objectiu de partida. Aquest, baix definició, és la creació d'una interfície que ofereix una comunicació entre un sistema PLC/SCADA gestionat per l'entorn TIA Portal de Siemens i una lògica de control intel·ligent que permeti preveure de manera meteorològica i energètica les actuacions que s'haurien de tindre.

El primer repte ha esmentat el treball de documentació que s'ha donat per aconseguir, estudiant els diferents sistemes de l'entorn, la tecnologia existent, el circuit del prototipus i la casuística necessària, adaptar una solució òptima, precisa i tàcita per a la situació de partida donada.

Seguidament l' implementació adequada dels diferents blocs a més de l' interfície que involucren els servidors externs, el maneig de respostes d'aquests, la interpretació de casos, la traducció d'aquest context i solucions en el programa de l' interfície ha esdevingut fonamental en la resolució.

Finalment, l' ajust de l' interfície i servidors, últims tests i corroboració final en funció dels resultats obtinguts, han sigut claus per a donar un context adequat als resultats obtinguts.

8.2. ESTUDI PRÈVI

Es requereix un mapa visual i teòric profund del sistema per a poder crear una algorítmica efectiva de les alternatives de resolució que es proposin. Els reptes en l' inici del projecte han escaigut en forma de pregunta.

- Quins valors llegir i quins escriure per a la millor implementació possible?
- Quins casos tindré en compte i com TIA PORTAL entén aquests casos?
- Interferiran les ordres que el SCADA envia a les DB a l'escriptura feta per l'aplicació?

En els següents apartats es presenten les diferents eines i metodologies que s'han seguit amb l'ajut de la tecnologia necessària per portar a terme la solució que es presenta, responent aquestes preguntes.

8.3. RESOLUCIÓ

8.3.1. TIA PORTAL

El primer plantejament, i l' inici de l' implementació passa per comprendre com es comunica el SCADA amb el PLC i com les dades són interpretades en cada nivell de les lectures i escriptures, aquesta comprensió es torna més visual en la Figura 14.

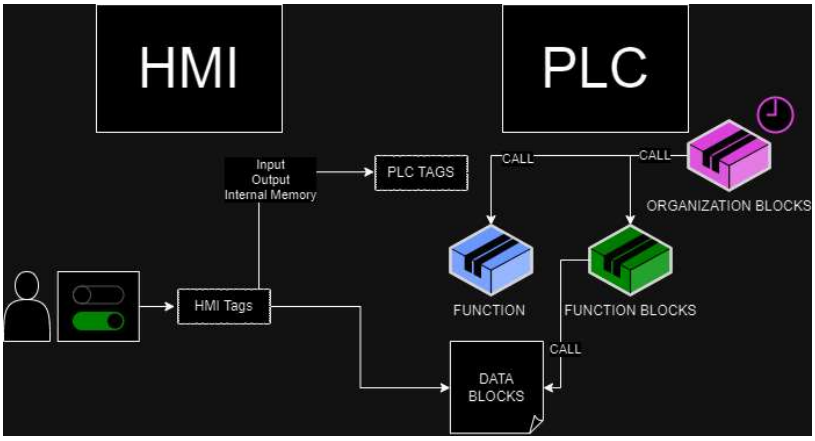

Figura 14. Esquema de comunicació d'accionament HMI a execució de variables i funcions al PLC.

La comunicació entre les diferents parts d'execució i memòria de l'entorn SCADA/PLC es dona mitjançant els diferents blocs de memòria i execució mencionats abans(DB,FB,FC..). En la figura 15 es pot observar d'una forma més il·lustrativa quan es crea una etiqueta/identificador, en aquest cas l'usuari indica: el nom d'aquesta, el tipus de dades que serà(bool, int, Time_Of_Day...), la connexió establerta amb el PLC('Conexión_1'), l'adreça de memòria, el mode d'accés i el cicle d'adquisició en el que es desitja que es capturin aquestes dades. Tant pot estar associada a una DataBlock, com pot ésser una entrada, sortida o un marcador.

Figura 15. Assignació de variable a DataBlock(DB), específicant el byte i bit de memòria d'aquella àrea específica d'exemple.

Com es pot observar en el menú d'identificador, com a exemple, es pot assignar una adreça i especificar el lloc de memòria que es pretén per aquell identificador en el DB. Per tant, DB8.DBX0.0 significa emmagatzemament en el Data Block número 8, en el byte i bits de dades 0,0, respectivament.

Part de la solució ha passat per la creació, amb les tècniques abans comentades, d'un botó al SCADA que li dona la possibilitat a l'usuari d'utilitzar la xarxa neuronal per a que es manegin les electrovàlvules, bombes i la tapa del RCE de manera adequada. Aquest botó emmagatzema com a Merker(marcador) un bit de memòria. Els Merker són utilitzats per l'emmagatzematge temporal o intermig de condicions lògiques necessàries per al control de l' interfície.

Donat que s'està treballant en una interfície que es aplicada en un sistema que varia en marges regularment baixos (el temps meteorològic), no es necessita una resposta en marges de segons, per tant, s'ha optat per enviar una sol·licitud cada 15 minuts de l'hora del rellotge, siguin i 15, i 30, 45 o en punt, mentrestant el botó de xarxa neuronal està actiu. Aquest botó de Neural Network Mode s'observa en la Figura 16 i 17 en aspecte, tal com en la pantalla del PC i en el SCADA.

Figura 16. Botó de neural Network Mode activat.

Figura 17. Botó de Neural Network Mode desactivat

Cal comentar que el sistema RCE, tal com en la resistència del tanc E2 i el bescanviador, contenen un control horari el qual s'utilitza per al canvi d'estat automàtic del circuit en funció de les hores d'activació/desactivació indicades per l'usuari. Aquestes hores es posen en un panell que apareix cada cop que s'entra en uns requadres del SCADA. En aquests panells es designa el dia, l'hora en la que es vol activar i l'hora en la que es vol desactivar el mode nocturn i diürn del RCE.

Com la resistència del tanc fred i el bescanviador són paràmetres que s'activen en un horari diferent i en dependència a variables exteriors, s'ha decidit no donar un control estricte a aquests quan s'activa la xarxa neuronal en fi de donar una independència beneficiosa que pot ajudar a que la lògica general de circuit s'executi mentrestant aquests tinguin un comportament independent. Per tant, l' interfície controla els principals actuadors del circuit: les bombes, electrovàlvules i la coberta del RCE, que tot i no tindre

una part mecànica implementada per a la seva execució, té una part associada en TIA Portal en forma de variable.

8.3.2. ESTRUCTURA DE BLOCS DE RESOLUCIÓ

La resolució del problema tant com el resultat final estan estructurats per blocs, aquests estan dividits en: explicació del funcionament general del bloc acompanyat d'un diagrama il·lustratiu, llibreries principals utilitzades a cada bloc, variables inicialitzades i funcions utilitzades en detall. També s'acaba finalitzant el bloc amb un diagrama de flux que respon a l'estructuració d'aquest.

En la Taula 3 es pot observar de forma més desglossada cada contingut del bloc de la resolució final.

Taula 3. Explicació dels diferents blocs de resolució.

BLOCS	FUNCIÓ	CONTINGUT
BLOC I	Lectura de valors dels sensors.	Relació TIA Portal – interfície en la lectura i explicació del codi.
BLOC II	Distribució de dades.	Relació interfície – servidors i explicació del codi d'ambdós.
BLOC III	Reconeixement i lògica d'estats.	Relació resposta del servidor i explicació del codi.
BLOC IV	Escriptura	Relació interfície – TIA Portal en l'escriptura i explicació del codi final.
BLOC V	Connexió i desconnexió	Quin funcionament té el botó Connectar, la qual assembla els 4 blocs anteriors i com funciona Desconnectar.
BLOC VI	Interfície gràfica	Part gràfica de l' interfície feta en Tkinter/ttkbootstrap i visualització d'aquesta

Cal afegir que en les parts de “EXPLICACIÓ GENERAL” les il·lustracions que es mostren són per millorar la comprensió del lector al respecte del funcionament del bloc. En la part “EXPLICACIÓ DEL CODI” es mostra el diagrama de flux fidel.

8.3.3. BLOC DE RESOLUCIÓ I: LECTURES

EXPLICACIÓ GENERAL

La primera part de l'execució de l' interfície ve de la mà de la lectura de valors. No tant sols dels sensors que ens atenyen i que hem d'enviar, si no també de l'estat dels actuadors que ens informen del mode del circuit juntament amb la lectura del estat del botó del SCADA. Aquesta fase funciona tal com es pot il·lustrar en la Figura 18:

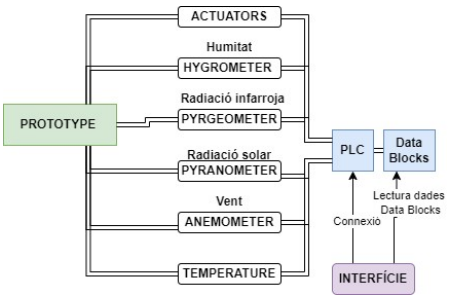

Figura 18. Diagrama il·lustratiu de la lectura del PLC.

Per a la connexió d'un PLC Siemens amb snap7 primer s'inicia un client. Quan es vol llegir informació, aquest client la sol·licita al PLC sempre i quan s'hagi establert una connexió amb èxit de forma prèvia. Es crida a aquest client quan es vol efectuar la lectura indicada juntament amb el bloc(DB, FB,FC,Tag...) i la àrea de memòria a la que es vol accedir.

EXPLICACIÓ DEL CODI

En la Taula 5 podem observar les llibreries i els mòduls utilitzats.

Taula 4. Llibreries amb els mòduls i la funció utilitzada en cada mòdul.

Llibreries	Mòduls	Funció
Snap7		Connexió del PLC i inicialització del client
Snap7.util	get_bool	Capta el valor booleà dintre d'un bytearray específic.
	get_dword	Capta el valor d'una double word(4 bytes) dintre d'un bytearray específic.
	db_read	Llegeix el bytearray d'un DB específic.
	read_area	Llegeix dades d'una àrea específica de memòria.
Snap7.types	Areas	Definició de la localització que es busca, en aquest cas àrea potser en un DB, Input, Output o Marcador.

No s'inicialitzen variables en l'entrada del codi.

1. **Funció readbooldb(**client, numeroDB,direccion,bit**)**

És la funció general que ens permet llegir en els bits que volem dels diferents bytes del bytearray dintre de el DataBlock especificat, aquesta retorna el valor de lectura.

Paràmetres d'entrada:

- client: el client inicialitzat per part de Snap7 per la connexió amb el PLC.
- numeroDB: el número de DataBlock específic
- direccion: byte de memòria on s'emmagatzema la dada que es vol llegir.
- bit: bit de memòria on s'emmagatzema la dada que es vol llegir.

Paràmetre de sortida

- valor: valor de la lectura

2. **Funció Lectura(**dispositivo**)**

És la funció que estableix dos diccionaris, un serà SensorData i l'altre EstadoActuadores. Aquests contindran la lectura dels sensors i la lectura dels estats dels actuadors, respectivament, també llegeix el valor de una etiqueta que correspon al valor booleà del estat del botó de xarxa neuronal. Per llegir les dades dels actuadors crida a la funció readbooldb().

Paràmetres d'entrada:

- dispositivo: client inicialitzat per part de Snap7 per la connexió amb el PLC.

Paràmetres de sortida:

- SensorData: diccionari amb totes les lectures de les cadenes, aquest conté una clau(key) que es el nombre del sensor que s'està llegint i un valor que és una string que conté el valor llegit del sensor.
- EstadoActuadores: diccionari amb la mateixa distribució que SensorData.
- ButtonRNState: variable que retorna el estat del botó que mostra si es vol la xarxa neuronal o no. No és un diccionari, és una variable.

3. **Funció readHour(**client, numeroDB, direccion**):**

Funció que permet llegir la hora emmagatzemada en els diferents DataBlocks que indiquen el control horari nocturn i diürn del RCE.

Paràmetres d'entrada:

- client: client inicialitzat per part de Snap7 per la connexió amb el PLC.
- numeroDB: el número de DataBlock específic
- direccion: byte de memòria on s'emmagatzema la dada que es vol llegir.

Paràmetres de sortida:

- tiempo: retorna el temps que s'ha extret de la memòria del DB

8.3.4. BLOC DE RESOLUCIÓ II:DISTRIBUCIÓ DE DADES

EXPLICACIÓ GENERAL

El segon bloc d'execució està inevitablement unit al primer ja que un cop es reben les dades es distribueixen als punts de comunicació desitjats. Aquesta distribució com hem comentat en algunes ocasions, és asíncrona i concurrent, brindant al sistema la capacitat d'enviar les dades d'una manera ràpida i eficient a diversos punts, com en el cas, seran els servidors. En la Figura 19 es pot observar un diagrama il·lustratiu del procés general del bloc de resolució II.

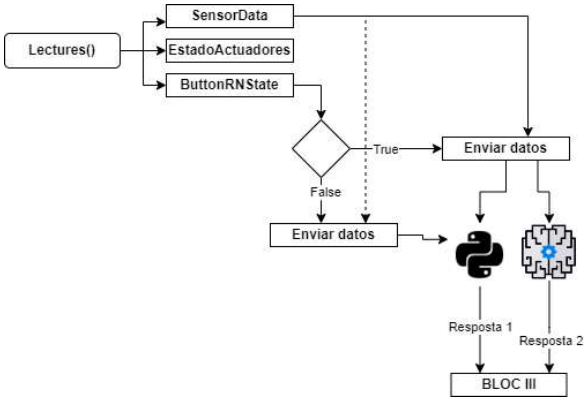

Figura 19. Diagrama il·lustratiu del funcionament general del Bloc II

EXPLICACIÓ DEL SERVIDORS

Les llibreries utilitzades en els servidors es poden observar en la Taula :

Taula 5. Llibreries del servidors corresponents a l'aplicació de Python i la Xarxa neuronal.

Servidor	**Llibreries**	**Mòduls**	**Funció**
Servidor xarxa neuronal/Servidor aplicació Python	**Http.server**	BaseHTTPRequestHandler	Classe base que proporciona mètodes per a gestionar els diferents tipus de sol·licitud.
		HTTPServer	Classe que gestiona les connexions i gestiona les peticions d'entrada i sortida.
Servidor xarxa neuronal/Servidor aplicació Python	**json**		Permet manipular dades en el format lleuger json.
Servidor xarxa neuronal	**random**		Permet aleatoritzar les respostes que s'envien per part del servidor.

En aquest servidor no s'inicialitzen variables.

Les funcions i classes de les quals consta són:

1. **Class RequestHandler(**BaseHTTPRequestHandler**):**

La classe heretada de la classe BaseHTTPRequestHandler, RequestHandler. BaseHTTPRequestHandle conté el mètode do_POST() el qual permet enviar un missatge de volta al client i manejar les sol·licituds POST. El mètode es utilitzat per la classe creada.

   a. **Funció do_POST():**

      Aquesta funció segueix aquests passos en l'ordre que es comenta:

      - Obté la longitud de la sol·licitud i llegeix les dades.
      - Descodifica les dades, si aquestes no estan en JSON entrarà en una excepció.
      - Prepara la resposta i la envia al client de volta.

2. **Funció Ejecucion(**ClaseServidor, ClaseHandler,port**):**

La funció Ejecucion defineix i inicialitza el servidor.

Paràmetres d'entrada:

- ClaseServidor: indica la classe que s'utilitza com a servidor la qual serà HTTPServer.
- ClaseHandler: indica la classe que es farà servir per gestionar les sol·licituds que arribin.
- port: defineix el port al que es comunica.

L'única diferència entre el servidor python i el servidor de la xarxa neuronal, és el quina resposta es dona. L'aplicació python sempre tornarà un "Data transaction has been successfull " i la xarxa neuronal tornarà, en el context del nostre servidor, de forma completament aleatòria un si, no o rain/wind. A continuació expliquem en quina lògica es tradueixen:

   - SI: canvi d'estat del circuit.
   - NO: el circuit es manté en l'estat que es vol.
   - RAIN/WIND: es posa el circuit en parada per condicions meteorològiques o altres causes que provoquin una casuística desfavorable, sigui per funcionament o consum energètic.

EXPLICACIÓ DEL CODI

Havent explicat com es construeixen els servidors cal assemblar les peces i veure com funciona la distribució per part del client.

Les llibreries utilitzades s'expliquen en la Taula 6:

Taula 6. Llibreries utilitzades en la distribució de dades

Llibreries	Mòduls	Funció
aiohttp		Llibreria que permet la comunicació asíncrona entre webs o servidors locals
json		Permet manipular dades en el format lleuger json.
asyncio		S'utilitza per escriure codi concurrent utilitzant async/await

En aquest bloc no s'inicialitzen variables.

Les funcions implementades per a la correcta distribució són:

1. **Funció SendjsonData2NeuralNetwork(**session,datos):

Aquesta funció prepara les dades de SensorData, envia una sol·licitud POST HTTP asíncrona i retorna la resposta que es rep a la xarxa neuronal.

Paràmetres d'entrada:

- session: objecte que manté obertes les connexions sense tindre que comunicar-se de forma intermitent.
- datos: entrada de dades que s'envien.

Paràmetres de sortida:

- NNresponse.text: retorna el text de la resposta, si no s'ha trobat un recurs llavors es retorna un None.

2. **Funció SendjsonData2PythonApp**(session,datos):

Aquesta funció prepara les dades de SensorData, envia una sol·licitud POST HTTP asíncrona i retorna la resposta que es rep a l'aplicació Python.

Paràmetres d'entrada:

- session: objecte que manté obertes les connexions sense tindre que comunicar-se de forma intermitent.
- datos: entrada de dades que s'envien.

Paràmetres de sortida:

- NNresponse.text: retorna el text de la resposta, si no s'ha trobat un recurs llavors es retorna un None.

3. **Funció SendAllData**(datos):

Aquesta funció crea una nova sessió HTTP, envia totes les dades de manera asíncrona concurrent cridant a les funcions SendjsonData2NeuralNetwork() i SendjsonData2PythonApp() i retorna la resposta en una array de dos.

Paràmetres d'entrada:

- datos: entrada de dades que s'envien.

Paràmetres de sortida:

- results: array amb les dues respostes.

4. **Funció SendAllDataNormalMode**(datos):

Aquesta funció crea una nova sessió HTTP, envia totes les dades de manera asíncrona concurrent cridant a les funcions SendjsonData2PythonApp() i la seva resposta.

Paràmetres d'entrada:

- datos: entrada de dades que s'envien.

Paràmetres de sortida:

- results: resposta per part de l'aplicació Python.

Per mostrar l'execució es mostra en el diagrama de flux de la Figura 20 :

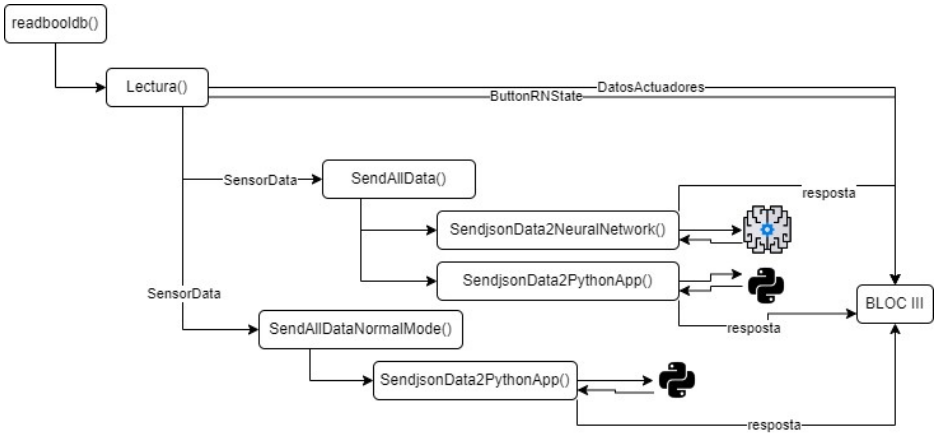

Figura 20.Diagrama de flux de la lectura i distribució de dades, la resposta que es genera per part de la xarxa neuronal es gestiona en el bloc III junt amb l'estat actual del circuit.

8.3.5. BLOC DE RESOLUCIÓ III: RECONEIXEMENT I LÒGICA D'ESTATS

EXPLICACIÓ GENERAL

Havent llegit i coneixent cadascun dels estats dels actuadors, es pot conformar una lògica d'identificació d'estat. Els diferents estats del circuit poden ser HeatMode(mode diürn), ColdMode(mode nocturn), AutomaticMode(mode automàtic) , Parada i s'adopta el valor None si ens trobem en estats que no corresponen a cap configuració en específic.

En la Taula 6, es poden observar els detalls de com es gestiona per a cada mode del circuit cada resposta de la xarxa neuronal:

Taula 7. Taula que mostra totes les possibles combinacions de nous estats que es produeixen en funció de la resposta de la xarxa neuronal.

Estat del circuit	Resposta de la xarxa neuronal	Nou estat del circuit
Cold Mode	Si	HeatMode
Heat Mode	Si	ColdMode
Automatic Mode	Si	AutomaticMode
Parada	Si	HeatMode [hora definida al SCADA com diürna]
	Si	ColdMode [hora definida al SCADA com nocturna]
	Si	Parada [hora entre HeatMode i ColdMode]
ColdMode	No	ColdMode
HeatMode	No	HeatMode
AutomaticMode	No	AutomaticMode
Parada	No	Parada
ColdMode/HeatMode AutomaticMode/Parada	rain/wind	Parada

EXPLICACIÓ DEL CODI

No es crida a cap llibreria a aquest bloc però si s'inicialitzen prèviament varis diccionaris.

El primer diccionari és SystemCase, el qual associa una tupla que combina un conjunt d'estats(clau) a un estat en específic(valor). Com a exemple representatiu:

('A', 'B', 'C', 'D',...): 'ColdMode'

El segon diccionari és ConfiguracionesActuadores, el qual associa un diccionari que associa una tupla que es una combinació lògica de dos o de tres(clau del diccionari interior) a un estat específic d'un actuador(valor del diccionari interior). Per mostrar un exemple representatiu:

```
ConfiguracionesActuadores = {
  "Actuador1": {
    (True, True): "MM",
    (True, False): "MP",
    (False, False): "A",
    (False, True): None,
  }
```

Aquests diccionaris estan associats a tots els actuadors del circuit i s'utilitzen per a poder confirmar o saber quina es la casuística d'estats en l'explicació del codi que es farà a continuació.

Les funcions que s'utilitzen en aquest codi són:

1. **Funció StateStablisher**(DatosActuadores):

Funció que se li entra les dades dels estats dels actuadors, construeix una array d'estats lògics, consulta al diccionari ConfiguracionesActuadores per associar a cada actuador el string que correspon al estat en el que està i finalment retorna una array d'estats lògics i una tupla de la combinació de tots els estats.

Paràmetres d'entrada:

- DatosActuadores: diccionari d'entrada representatiu dels estats de cada actuador del circuit de forma lògica.

Paràmetres de sortida:

- Combinacion: tupla dels strings que representen els estats dels actuadors.
- valoresBooleanos: array que correspon a cada estat de cada bit del actuador en el que s'ha d'escriure.

2. **CheckSystemType**(combination,valoresBooleanos):

La funció que intenta veure si la tupla Combinacion que rep de StateStablisher() s'associa a algún valor del diccionari SystemCase()

Paràmetres d'entrada:

- combination: tupla dels strings que representen els estats dels actuadors.
- valoresBooleanos: array que correspon a cada estat de cada bit del actuador en el que s'ha d'escriure.

Paràmetres de sortida:

- Estado: retorna l'estat del circuit, si no existeix en cap combinació es retorna un None que també serà interpretat en les següents funcions.

3. **DecisionNoneorParada**(client):

Aquesta funció esta escrita per quan la resposta es "si" però l'estat del circuit es None o Parada, llavors és necessari que anem a revisar al control horari de les configuracions diürnes i nocturnes que l'usuari ha imposat.

Si estem entre les hores diürnes, es retorna un string "HeatMode1" ,si estem en hores nocturnes, es retorna un string tal com "ColdMode1" i si no estem en cap dels tres, llavors es retorna un string "Parada".

Paràmetres d'entrada:

- client: client inicialitzat per snap7.

Paràmetres de sortida:

- retorna el nou estat del circuit en aquesta casuística específica.

4. **<u>StateDecisionChangerYES</u>**(estado):

Aquesta funció està escrita per quan la resposta és “si”. Aquesta extreu un NouEstat en funció de l’estat del circuit que li entra. Segueix la lògica que ja s’ha vist en la Taula 7. Quan la resposta és si i l’estat és None o Parada crida la funció que hem vist abans.

Paràmetres d’entrada:

- client: client inicialitzat per snap7.
- estado: estat actual del circuit.

Paràmetres de sortida:

- NuevoEstado: retorna el nou estat del circuit en aquesta casuística específica.

5. **<u>StateDecisionChangerNO</u>**(estado):

Aquesta funció esta escrita per quan la resposta és “no”. Retorna com a nou estat l’estat que li entra.

6. **<u>StateDecisionnChangerRAINWIND</u>**(estado):

Aquesta funció esta escrita per quan la resposta és “rain/wind”. Retorna com a nou estat “Parada”entri quin li entri.

A continuació en la Figura 21, es pot visualitzar el diagrama de flux que correspon a aquest bloc.

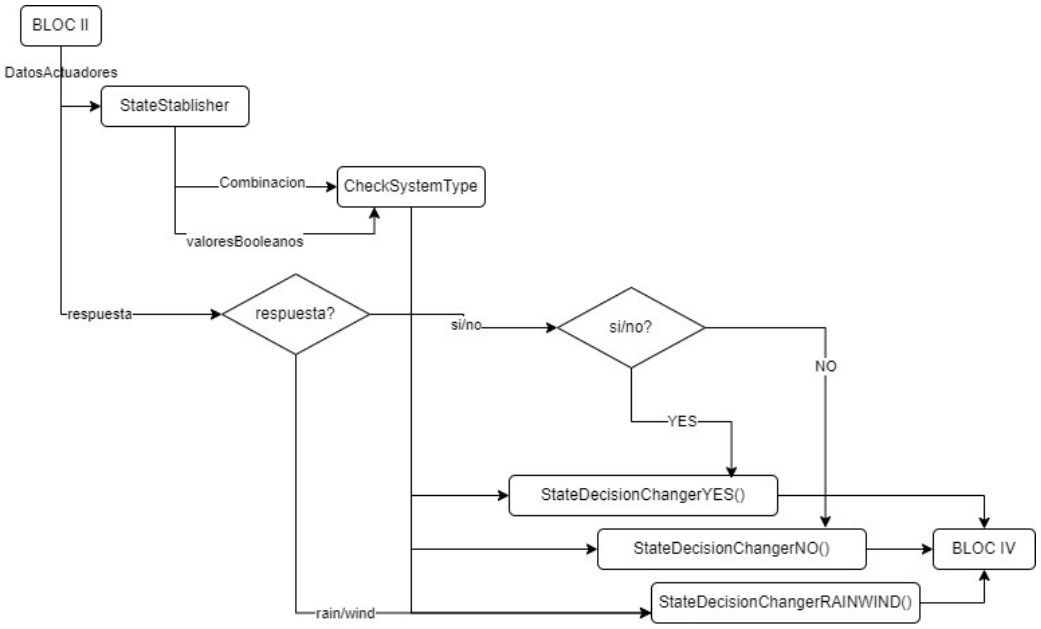

Figura 21. Diagrama de flux de la relació entre funcions del bloc III

8.3.6. BLOC DE RESOLUCIÓ IV:ESCRIPTURA

EXPLICACIÓ GENERAL

En aquest bloc es rep el nou estat del circuit i s'escriu en les àrees de memòria de les DataBlock adequades. En la Taula 8 es reflexa tots els bits que es volen portar sota control en el cas que sigui una bomba, una electrovàlvula o una tapa.

Taula 8. Taula que mostra els diferents bits en el que s'escriu.

Actuador	Bit1	Significat	Bit2	Significat	Bit3	Significat
Bomba	ActManB1	Marxa Manual/Automàtica	ActManMarxaB1	Marxa Manual/Paro Manual		
Electrovàlvula	ActManEV1	Marxa Manual/Automàtica	ActManMarxaEV1	Marxa Manual	ActManStopEV1	Paro Manual
Tapa	ActManTH/TV	Marxa Manual/Automàtica	ActManMarxaTH/TV	Marxa Manual/Paro Manual		

De forma representativa i com, en el diagrama de la Figura 22, el bloc IV actua de tal forma.

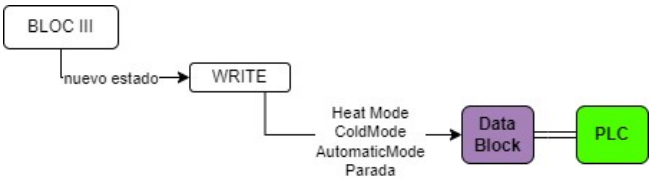

Figura 22. Diagrama que mostra l'actuació general del bloc d'escriptura.

EXPLICACIÓ DEL CODI

La llibreria utilitzada es *Snap7.util* i el mòdul específic *set_bool* el qual estableix un valor booleà en una localització d'un bytearray específic.

No s'inicialitzen variables.

Les funcions implementades són:

1. **Funció writebooldb**(client,numeroDB,direccion,bit,value):

Funció que escriu en un bit d'un bytearray determinat True o False, segons el "value" que s'estableixi.

Paràmetres d'entrada:

- client: client inicialitzat per snap7.
- numeroDB: número que correspon al DB que es vol accedir.
- direccion: byte al que es vol accedir.
- bit: bit al que es vol accedir.
- value: valor booleà que es desitja escriure en el bit del byte específic.

2. **Funció WriteColdMode**(dispositivo):

Funció que crida la funció writebooldb() per escriure en les àrees de memòria desitjades. S'utilitza per escriure la combinació corresponent a ColdMode. Com hi ha dos ColdMode1 o 2, segons si la bomba B1 esta en Automàtic o ManualMarxa, s'ha decidit escriure ColdMode1.

Paràmetres d'entrada:

- dispositivo: client inicialitzat per snap7.

3. **Funció WriteHeatMode**(dispositivo):

Funció que crida la funció writebooldb() per escriure en les àrees de memòria desitjades. S'utilitza per escriure la combinació corresponent a HeatMode. Com hi ha dos HeatMode1 o 2, segons si la bomba B1 esta en Automàtic o ManualMarxa, s'ha decidit escriure HeatMode1.

Paràmetres d'entrada:

- dispositivo: client inicialitzat per snap7.

4. **Funció WriteAutomaticMode**(dispositivo):

Funció que crida la funció writebooldb() per escriure en les àrees de memòria desitjades. S'utilitza per escriure la combinació corresponent a AutomaticMode.

Paràmetres d'entrada:

- dispositivo: client inicialitzat per snap7.

5. **Funció WriteParadaMode**(dispositivo):

Funció que crida la funció writebooldb() per escriure en les àrees de memòria desitjades. S'utilitza per escriure la combinació corresponent a Parada.

Paràmetres d'entrada:

- dispositivo: client inicialitzat per snap7.

6. **Funció WriteNoneMode**(dispositivo,combinaciones):

Funció que crida la funció writebooldb() per escriure en les àrees de memòria desitjades. S'utilitza per escriure la combinació que inicialment surt de la funció StateStablisher com a valoresBooleanos, així assignant cada estat al estat d'entrada fent que None es perpetuí.

Paràmetres d'entrada:

- dispositivo: client inicialitzat per snap7.

- combinaciones: array que correspon a cada estat de cada bit del actuador en el que s'ha d'escriure.

7. **<u>Funció FinalWritetoPLCNNmode</u>**(dispositivo,estado,combinaciones):

Funció que crida segons el cas, les funcions WriteColdMode(), WriteHeatMode(),WriteNoneMode(),WriteParadaMode() o WriteAutomaticMode() segons el Nou estat del circuit.

Paràmetres d'entrada:

- dispositivo: client inicialitzat per snap7.
- estado: nou estat del circuit
- combinaciones: array que correspon a cada estat de cada bit del actuador en el que s'ha d'escriure.

El diagrama de flux d'aquest bloc es pot veure en la Figura 23:

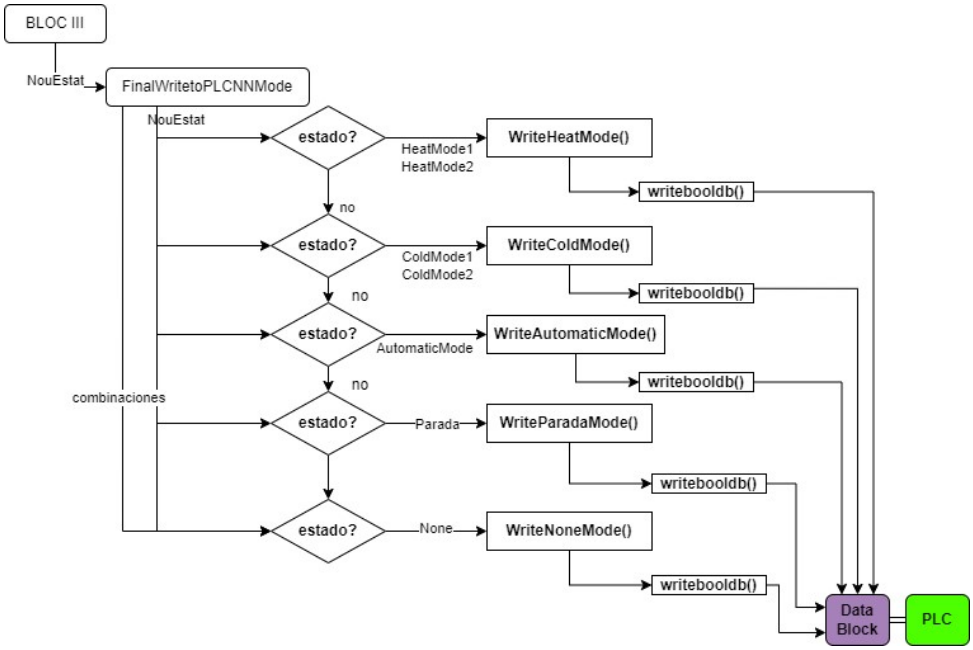

Figura 23. Diagrama de flux del bloc IV.

8.3.7. BLOC DE RESOLUCIÓ V: CONNEXIÓ I DESCONNEXIÓ

Quan s'obra l' interfície, la primera pantalla que apareix es la que es presenta en la figura 24:

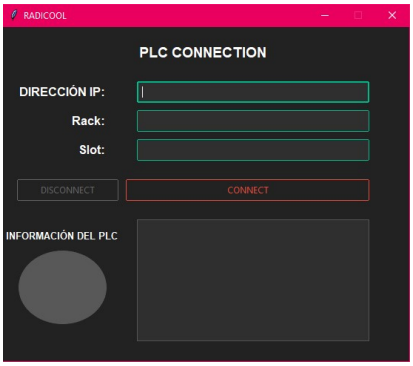

Figura 24. Pantalla inicial interfície

Com es pot observar, hi ha dos grans botons, els quals són "CONNECT" i "DISCONNECT". El botó que correspon a "CONNECT" se n'encarrega d'executar els 4 blocs que s'han comentat anteriorment de la forma adequada el qual entra en un bucle mentrestant no es prem "DISCONNECT". Per altra banda, "DISCONNECT" se'n encarrega d'aturar el procés i desconnectar-se del PLC actual.

Com s'ha esmentat en els blocs anteriors la distribució de dades és asíncrona, això vol dir que les funcions que corresponen a la distribució i la funció Conectar (la qual pertany al botó "CONNECT") també han de ser asíncrones. El benefici a obtindre d'aquesta conclusió és que pot augmentar l'alimentació de comunicació de dades en altres servidors d'una manera més fàcil, el preu a pagar són dues funcions més de codi que el que fan es executar la funció Connectar en un fil(*Thread*) diferent del principal. Una manera de visualitzar això seria l' il·lustració de la Figura 25 que ens mostra totes les funcions per les que es passa en Connectar i els Threads que la mateixa interfície té. En aquesta la funció Connectar va des de "Try Connection to PLC" a quan es desconnecta aquest.

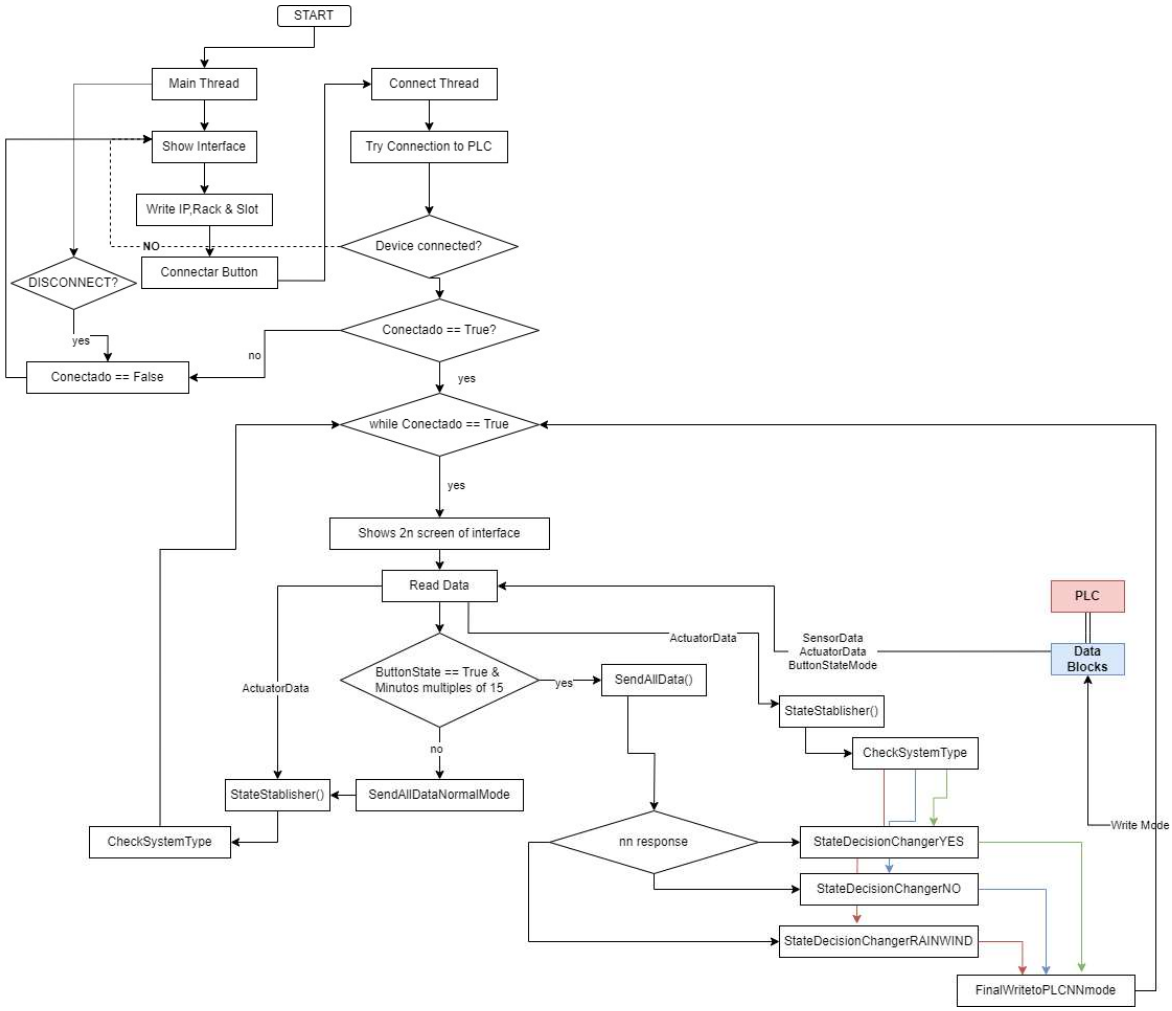

Figura 25. Diagrama de flux que mostra les funcions per les que es passa en l' interfície i específicament en les funcions de Connexió i Desconnexió

Com s'ha pogut observar quan s'escriu per part de l'usuari la IP,Rack i el slot i es prem el botó de connexió aquest crida una funció que es diu ConectarButton i acte seguit una funció que es diu ConectarThread.

La funció ConectarButton està associada al Button, quan es crida aquesta posa la variable Conectado en True i inicia un Thread que específica la funció a executar. Aquesta funció és ConectarThread.

ConectarThread a la vegada és una funció que crea un nou esdeveniment de bucle els qual no pararà fins que es completi la funció Connectar(). Acte seguit de l'execució de ConectarThread, inicia Connectar.

8.3.8. BLOC VI: INTERFÍCIE GRÀFICA

En l'anterior bloc, en el diagrama, es pot observar que hi ha diferents pantalles que s'intercanvien.

9. RESULTATS

A continuació, s'explicaran els resultats adquirits de les diferents proves realitzades a cada bloc, aquestes constaràn d'una resolució independent, i finalment d'una integració conjunta.

Per a fer el control i progrés de l'aplicació, s'ha utilitzat un software extern que es diu S7-PLC DB Simulator en un inici, el qual dona en un entorn realista i precís de l' interacció amb DataBlocks que pot oferir TIA PORTAL en el protocol S7 utilitzat. Això permet verificar i corroborar la lògica del programa sense cap necessitat d'accés a hardware físic com es pot observar en la Figura 16. Permet ajudar en millorar la manera en la que s'exerceix de manera esglaonada una escalabilitat més controlada i segura.

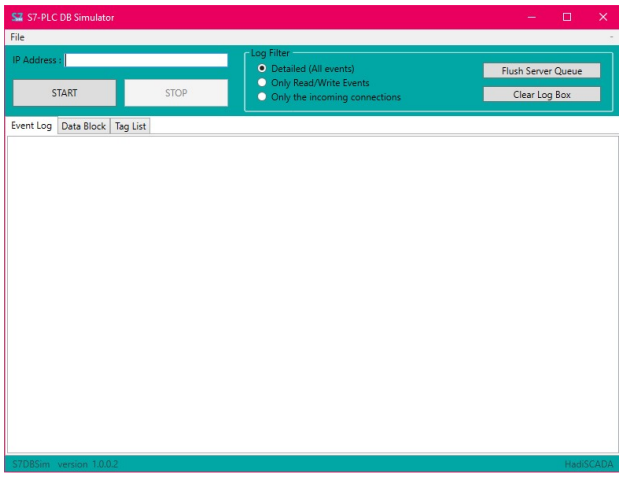

Figura 26.Aplicació S7-PLC DB Simulator

Aquest permet crear un "PLC" que existeix en la IP determinada, en un rack i un slot específic. Dins d'aquest, es permet la creació de DB i l'enllaç de Tags que s'associen als DB corresponents, podent elegir el tipus de canvi amb el temps o si romanen estables en un format específic. Havent incidit en l'entorn de testeig i desenvolupament, els resultats es basen, com es mencionarà en els següents apartats, en l' interacció real amb TIA PORTAL i s'han determinat en l'entorn d'execució del mateix prototipus.

9.1. INTERFÍCIE

L' interfície ofereix de forma interactiva una manera de connectar i realitzar la lògica de control pertinent. Consta de varies fases, una d'elles com es veu en la següent figura, visualitza la connexió i els diferents estats, en la caixa de text. En les Figures 27 i 28 es poden visualitzar les dues etapes de connexió i desconnexió d'aquesta interfície.

En aquesta s'introdueix la IP, el rack i el slot pertinents al nostre PLC i es prem Connect. Quan es prem, segueix per ordre aquesta manera d'actuar:

1) Lectura de les dades dels sensors i dels estats dels actuadors.
2) Distribució de les dades dels sensors als diferents servidors que ho esperen, la xarxa neuronal i l'aplicació en Python.
3) Anàlisis del estat actual del circuit en funció de l'estat dels actuadors, i casuística del nou circuit segons resposta de la xarxa neuronal.
4) Finalment, l'escriptura en els diferents bits de les DataBlock corresponent als actuadors.
5) Inicia de nou la lectura.

La següent figura reflexa l'estat d'entrada i l'estat de connexió:

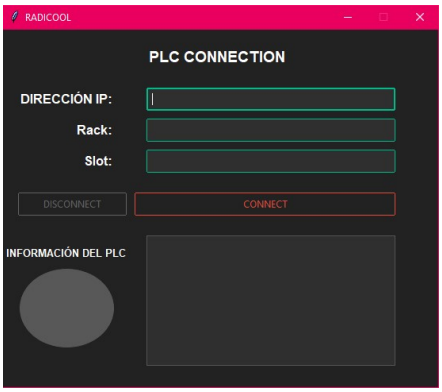

Figura 27. Entrada de l'interfície.

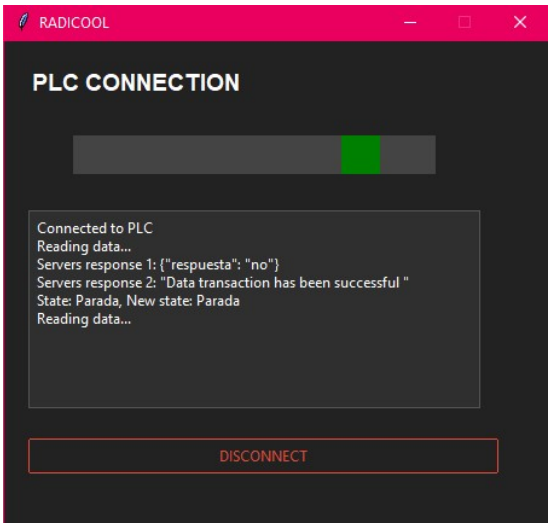

Figura 28. Interfície connectada.

Quan es desconnecta després d'haver sigut connectat ja un cop apareix la pantalla de la Figura 29 de nou:

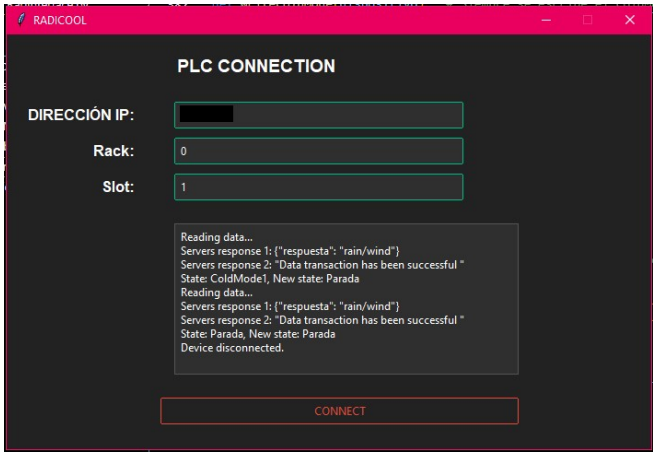

Figura 29. Pantalla quan s'ha desconnectat

Com podem observar, aquest és el patró de pregunta resposta demandat. L'algoritme del sistema va tal que:

1. Usuari prem Connect
2. El sistema utilitza la lògica comentada amb anterioritat al respecte de la relació amb el PLC.
3. Torna a iniciar el bucle fins que s'ha premut Disconnect o s'ha tancat la finestra de forma manual. Tal com es pot observar en la següent imatge la distribució i per tant la lectura de paràmetres s'implementa.

9.2. RESULTATS DEL BLOC I, II

En el bloc de lectura i distribució s'han assolit els objectius , com es pot observar en les diferents captures de pantalla, es llegeix i envia la xarxa neuronal.

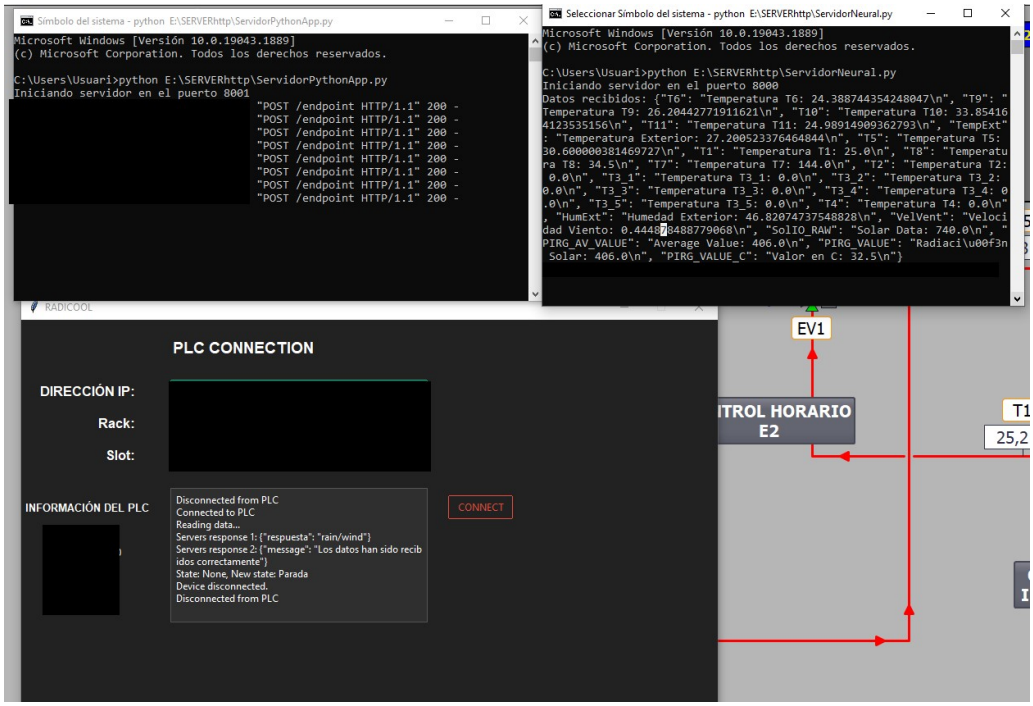

Figura 30. Demostració de lectura i distribució de l'aplicació.

9.3. RESULTATS DEL BLOC III

El bloc III corrobora la casuística de casos i el seu anàlisi, aquests s'executen en l'aplicació de la manera correcta, ja que l'escriptura correcta demostra com a símptoma de la bona lògica d'anàlisis de casos.

9.4. RESULTATS DEL BLOC IV

En el bloc IV, es mostra el resultat que una operació d'escriptura definida del caire de la funció implementada en l' interfície opera de forma correcta. En l' arxiu "PruebasEscripturaLectura.py" s'intenta escriure en una bomba que inicialment està en Automàtic per a que operi en mode Manual Marxa, les funcions definides en l'arxiu estan en l' interfície. L'execució de l' interfície també funciona aplicant al TIA PORTAL en comptes de S8-PLC DB Simulator, però intentar implementar una execució d'un servidor amb resposta aleatòria no es una bona pràctica. Com podem observar a les Figures 20 y 21, l'escriptura individual de paràmetres i a la vegada la demostració que funciona.

Figura 31.Estat del selector per pantalla.

Després de l'escriptura de bits i de paràmetres la resposta generada es tal que:

Figura 21. Estat del selector després de l'escriptura.

Per tant, en aquest Bloc final es demostra que l' interfície té totes les característiques demandades i cobreix tots els possibles casos.

9.5. ESCALABILITAT DE L' INTERFÍCIE

Pensar en un futur i una visió realista de la millora d'un sistema sempre és una tasca tant premiable com la d'optimitzar i fer més eficient aquest. Mostra que en aquest projecte s'ha pensat en aquests aspectes són l'ús d'aquestes tecnologies o softwares:

- S7-PLC DB Simulator no tant sols és un emulador de l' interacció entre les data blocks característiques de TIA PORTAL i el protocol de comunicació S7 obra un camí a l'escalabilitat sense hardware en noves implementacions de sistemes o nous projectes que emprin programari de Siemens.
- Snap7 és una suite de comunicació de codi obert que permet la comunicació en PLC de Siemens utilitzant el protocol S7, cosa que a més de significar una accessibilitat i un cost zero, té una gran flexibilitat i integració en projectes d'aquest caire. És compatible amb la gran majoria de PLC de la gama Siemens S7.
- Aiohttp és una biblioteca de codi obert que es pot utilitzar com a client o servidor en comunicacions HTTP, pot manejar de manera eficient gestions de sol·licituds de

manera concurrent i asíncrona. Això no tant sols permet la millora i eficiència del sistema si no l'adaptació senzilla per codi de noves comunicacions concurrents.

Com a proposicions per a la millora de l' interfície es podria discutir:

1. Comunicació amb una base de dades. Gestió de les dades dels sensors, i repartició en cas de que els servidors actuals ho demandin.
2. Augment i millora de les capacitats actuals de l' interfície.
3. Aprofitament de la seva capacitat de manejar dades de forma concurrent per establir connexions amb més servidors i establir un sistema de control més enriquit i precís.

10. CONCLUSIONS

Finalment, havent vist prèviament els blocs de resolució i els resultats finals, es conclou que s'ha aconseguit l'objectiu, el qual era implementar una interfície que connecti una xarxa neuronal i un sistema PLC/SCADA programat en l'entorn TIA Portal, no tant sols això, a més s'ha donat la capacitat a l'usuari d'elegir d'una forma fàcil i senzilla si el vol utilitzar o no.

He de mencionar que a l' interfície encara té un llarg camí per davant, com a mostra d'això i per falta de temps, cal aclarir que no he pogut acabar d'implantar totes les respostes a excepcions que m'haguessin agradat, no he pogut donar una connexió amb una base de dades i tampoc implementar un sistema de logging. Amb el temps anirà obtenint més robustesa i millorarà les bases que aquí estan assentades.

Tal com hem mencionat la part negativa cal mencionar que en la consecució d'aquests objectius he aprés força al respecte de la comunicació de PLC en un entorn com TIA Portal, la comunicació de servidors, nous protocols existents i la programació d'aquests essent un TFG que m'ha brindat molt aprenentatge en integració i comunicació de sistemes d'automatització amb aplicacions de tercers o interfícies externes.

11. ANÀLISI DE COSTOS

Donat que les biblioteques i tots els programes són de codi obert, el càlcul de costos no té un caràcter econòmic i es basa en el temps que s'ha tardat per a la realització en cada fase important en la persecució dels objectius.

Cal dir que la variabilitat d'hores, la complexitat, el càlcul de temps i la persecució dels objectius, fan que sigui difícil el càlcul d'hores implicades. Per tant, l' anàlisi de costos temporals es presenta d'una forma estimada en la següent Taula:

Taula 9. Taula que reflexa el temps invertit en l'interfície.

TASCA	TEMPS
Reunions i inversió de temps en investigació de solucions	40
Lectura i documentació	60
Estudi de projecte TIA PORTAL i aprenentatge de tecnologia.	60
Desenvolupament de l' interfície, servidors i proves	120
Proves finals, fer testos, correcció d'errors i recopilació d'errors	50-70
Memòria, recopilació i anàlisi de resultats	15-25
Total d'hores aproximades	**345-375**

El temps d'investigació, documentació i reunions juntament amb aprenentatge es va allargar més del compte ja que TIA Portal era una tecnologia desconeguda per a mi. De la mateixa forma, les solucions aportades depenien del programa.

En primera instància, les solucions que es van provar i proposar donaven més complexitat al sistema, ja que una d'elles com a exemple, era el canvi de direcció de les Ip's per a que tota la informació passés del SCADA a l' interfície i d'aquesta al PLC. El problema era que si al prototipus li poses una connexió PROFINET i estableixes una lògica a TIA Portal, ara cal tornar a canviar la configuració prèvia, preparar una interfície que gestioni les dades com TIA Portal i reenviar les dades al SCADA. L' incompatibilitat que es proposa en certes àrees de la comunicació esmentada fa que aquesta solució no sigui viable ni eficient, tant pel sistema del PLC com per la complexitat de la interfície i passant per l'enteniment del sistema.

La millor solució passava per interferir de la forma adequada amb el sistema brindant a l'usuari de la llibertat d'escollir la interferència de la xarxa neuronal, entenent, que per part

de l'usuari quan es fa l'elecció de la xarxa neuronal no s'interferirà de forma Manual al sistema, de la mateixa manera quan la elecció sigui sense xarxa neuronal, aquesta no interferirà ni escriurà al sistema.

Un cop estudiada la solució, el desenvolupament de l' interfície, junt amb les hores de proves i errors han esdevingut essencials i com a resultat es el marc d'hores més gran en total. Finalment, el menor temps ha estat dedicat a rebre i notificar el context dels resultats.

12. REFERÈNCIES

BLOC INTRODUCCIÓ

[1] Sergi Vall Aubets, Marc Medrano Martorell, Cristian Solé Cutrona, Albert Castells.(2020).Combined Radiative Cooling and Solar Thermal Collection: Experimental Proof of Concept: https://repositori.udl.cat/items/bd06a437-74a2-42b7-907f-f6a26ba4f2db

[2] Bin Zhao, Mingke Hu, Xianze Ao, Nuo Chen, Gang Pei. (2018). Radiative cooling: A review of fundamentals, materials, applications, and prospects: https://www.sciencedirect.com/science/article/abs/pii/S0306261918318373

BLOC MARC TEÒRIC

[3] PROFINET.(s.f.). PROFINET: https://www.profinet.com/profinet-explained/technology-description

[4] Shiksha Online. (2024, Gener 23). Understanding Multilayer Perceptron (MLP) Neural Networks: https://www.shiksha.com/online-courses/articles/understanding-multilayer-perceptron-mlp-neural-networks/

[5] DataScientest. (s.f.). Convolutional Neural Network: Everything You Need to Know. *DataScientest*: https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know

[6] Medium. (2020,Setembre 3). Simple explanation of Recurrent Neural Network(RNN): https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363

[7] KeepCoding. (s.f). Diferencias entre paralelismo y concurrencia computacional: https://keepcoding.io/blog/paralelismo-y-concurrencia-computacional/

[8] Amazon Web Services. (s.f.). What is MQTT? *Amazon Web Services*. https://aws.amazon.com/es/what-is/mqtt/

[9] SIEMENS. (2013, Abril 29). Which organization blocks can you use in STEP7(TIA PORTAL): https://support.industry.siemens.com/cs/document/40654862/which-organization-blocks-can-you-use-in-step-7-(tia-portal)-?dti=0&dl=en&lc=es-CR

BLOC METODOLOGIA:

[10] Python.(s.f).http.server-HTTP servers: https://docs.python.org/3/library/http.server.html

ANNEX DE TAULES

ANNEX DE FIGURES