Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL: Desenvolupament d'un sistema de comunicació per a instrumentació Modbus genèric

AUTOR: Soto Telles, Francisco

DATA DE PRESENTACIÓ: Juny, 2020

TITULACIÓ: Grau en Enginyeria Electrònica Industrial i Automàtica

DIRECTOR: Del Río Fernández, Joaquín

DEPARTAMENT: Enginyeria Electrònica

QUALIFICACIÓ DEL TFG

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Aquest Projecte té en compte aspectes mediambientals: ☐ Sí ☐ No

RESUM

El protocol Modus és utilitzat comunament en molts equips de mesura i control en diferents àmbits. Mitjançant aquest protocol aquests equips poden ser teleoperats remotament des d'un sistema de monitoratge i control com podria ser un PC o un PLC. La definició del protocol Modbus permet ser utilitzat sobre RS485 o Ethernet. Tot i que aquests busos de comunicació permeten operar sobre el mateix bus diferents instruments, no existeix una estandardització a nivell Modbus entre els diferents fabricants que permeti l'ús d'una aplicació genèrica per poder comunicar simultàniament amb instruments o equips diferents.

Aquest projecte té com a objectiu resoldre la problemàtica existent sobre l'estandardització mitjançant el disseny, programació i test, d'una aplicació per la connexió de diferents dispositius i de diferents fabricants mitjançant el protocol Modbus.

Primerament al projecte es realitza un estudi teòric del protocol Modbus, el seu funcionament i diferents variants.

Per poder portar endavant la programació de l'aplicació es realitza un repàs del llenguatge utilitzat, que en aquest cas és LabVIEW.

Per la comprovació del correcte funcionament de l'aplicació es crea una configuració per testejar amb 2 cabalímetres (Simex i TUF-2000M) i un Arduino Mega. A més es realitza una videoconferència amb l'empresa Sensotec Instruments, S.A pel suport a la verificació de les dades llegides dels aparells i després posteriorment una visita a l'empresa i es testeja l'aplicació al seu banc d'assajos.

Paraules clau (màxim 10):

Protocol	Modbus	LabVIEW	Aplicació
Genèric	Comunicació		

ABSTRACT

The Modus protocol is commonly used in many measurement and control equipment in different areas. Through this protocol these teams can be teleoperated remotely from a monitoring and control system such as a PC or a PLC. The definition of the Modbus protocol allows it to be used over RS485 or Ethernet. Despite the fact that these communication buses allow different instruments to operate on the same diver, there is no Modbus level standardization between the different manufacturers that allows the use of a generic application to be able to communicate simultaneously with different instruments or equipment.

This project aims to solve the existing problem of standardization through the design, programming and potting of an application by connecting different devices and from different manufacturers using the Modbus protocol.

Firstly, the project is carried out a theoretical study of the Modbus protocol, its operation and different variants.

In order to carry out the application programming, a review of the language used is performed, which in this case is LabVIEW.

By checking the correct operation of the application, a configuration is created to test with 2 flowmeters (Simex and TUFO-2000M) and an Arduino Mega. In addition, a videoconference is held with the company Sensotec Instrumentos, S.A for the support of the verification of the data read from the devices and then later a visit to the company and the application is tested in its test bench.

Keywords (10 maximum):

Protocol	Modbus	LabVIEW	App
Generic	Communication		

SUMARI

INTRODUCCIÓ **11**

1 HISTORIA MODBUS **12**

2 QUÈ ÉS MODBUS? **13**

2.1 Versions **14**

2.1.1 Enron Modbus 15

2.1.2 Pemex Modbus 15

2.1.3 Modbus ASCII 15

2.1.4 Modbus Plus 15

2.1.5 Modbus sobre UDP 15

2.1.6 Modbus sobre TCP/IP 16

2.1.7 Modbus TCP/IP 16

2.1.8 Modbus RTU 16

3 MARC GENERAL MODBUS **17**

3.1 Descripció del protocol **17**

3.2 Unitat de dades de protocol (PDU) **17**

3.3 Unitat de dades d'aplicació (ADU) **18**

3.3.1 Característiques comunes 18

3.3.2 Formats estàndards 18

3.3.3 Protocol de Control de la transmissió (TCP) 19

3.3.4 Unitat terminal remota (RTU) 22

3.3.5 Serial ASCII 24

3.4 Comprovació d'errors 26

3.4.1 Errors mètode serial 26

3.4.2 Error mètode TCP 27

3.5 Model de dades 27

3.5.1 Classificació de dades 29

3.6 Model d'adreçament Modbus 30

3.7 Funcions 31

3.7.1 Definició de funcions públiques 32

3.7.1.1 Read Coils(01) – Read Discrete Inputs (02) 33

3.7.1.2 Read Holding Registers (03) – Read Input Registers (04) 34

3.7.1.3 Write Single Coil (5) 35

3.7.1.4 Write Single Register (06) 36

4 INSTRUMENTACIÓ I LABVIEW 36

4.1 LabVIEW 36

4.2 Entorn LabVIEW 37

4.2.1 Instruments Virtuals (VIs) 37

4.2.2 Panell frontal 37

4.2.3 Diagrama de blocs 38

4.2.4 Flux de dades 39

4.3 SIMEX SRL-49 40

4.4 Arduino 41

4.5 TUF-2000M 42

4.6 Convertidor RS-485 a USB **42**

5 APLICACIÓ MODBUS **44**

5.1 Flux del programa **44**

5.2 Funcionament de l'aplicació **46**

5.2.1 Pantalla Principal 47

5.2.2 Arxiu de configuració 48

5.2.3 Creació o obertura d'un setup existent 49

5.2.4 Afegir, editar o esborrar dispositius 50

5.2.5 Interrogar dispositius 52

5.2.6 Gràfics i registre de dades 52

6 COMPROVACIÓ DE LECTURES **54**

6.1 Fase Inicial **54**

6.2 Configuració proves en bus **54**

6.3 Proves a Sensotec S.A. **57**

7 ASPECTES DE MILLORA **59**

CONLUSIONS **61**

AGRAÏMENTS **62**

BIBLIOGRAFIA **63**

ANNEXOS **65**

SUMARI DE FIGURES

Figura 1. Pila de comunicacions Modbus....14

Figura 2. Marc general Modbus....18

Figura 3. Unitat de Dades d'Aplicació TCP....22

Figura 4. Unitat de Dades d'Aplicació RTU....24

Figura 5. Unitat de Dades d'Aplicació ASCII....25

Figura 6. Transmissió de dades sense error....28

Figura 7. Transmissió de dades amb resposta d'excepció....29

Figura 8. Mapeig de dades Modbus....31

Figura 9. Categories de codis de funcions....32

Figura 10. Exemple de panell frontal buit....38

Figura 11. Exemple de diagrama de blocs buit....39

Figura 12. Pantalla frontal del SRL-49....40

Figura 13. Simulador esclau Modbus amb Arduino....41

Figura 14. TUF-2000M....42

Figura 15. Convertidor RS-485 a USB....43

Figura 16. Gestor de dispositius....43

Figura 17. Diagrama de blocs de l'aplicació....45

Figura 18. Pantalla principal de l'aplicació....47

Figura 19. Exemple d'arxiu .ini amb 2 variables definides....49

Figura 20. Apartat Configuration del menú de l'aplicació 50

Figura 21. Pestanya Instruments del menú de l'aplicació.....50

Figura 22. Finestra d'edit instruments configuration.....51

Figura 23. Menú d'interrogació 52

Figura 24. Pantalla de visualització del gràfic.....53

Figura 25. Pantalla de registre de dades 53

Figura 26. Lectura del cabalímetre Simex.....55

Figura 27. Display del cabalímetre Simex.....56

Figura 28. Descripció del registre 3 del datasheet del Simex.....56

Figura 29. Descripció del registre 4 del datasheet del Simex.....56

Figura 30. Setup de connexió en bus de l'Arduino i els cabalímetres 57

Figura 31. Prova a Sensotec S.A 59

SUMARI DE TAULES

Taula 1. Model de dades Modbus.....29

Taula 2. Funcions Modbus més utilitzades 33

Taula 3. Exemple de Read Coils - Read Discrete Inputs.....33

Taula 4. Resposta del esclau a la sol·licitud de la Taula 3 34

Taula 5. Exemple Read Holding Registers - Read Input Registers.....35

Taula 6. Exemple de la funció Write Single Coil.....35

Taula 7. Exemple de Write Single Register 36

GLOSSARI DE SIGNES, SÍMBOLS, ABREVIATURES, ACRÒNIMS I TERMES

Bit: és la unitat d'informació mínima utilitzada en Informàtica i en Teoria de la Informació. Un bit pot tenir només dos estats mútuament exclusius.

Byte: és un grup de 8 bits.

DCE: Data Communication Equipment (Equipament de comunicació de dades)

Estàndard de facto: és aquella norma (o patró) que es caracteritza per no haver estat consensuada ni legitimada per un organisme d'estandardització a aquest efecte. És a dir, es tracta d'una norma generalment acceptada i àmpliament utilitzada per iniciativa pròpia (o popular) d'un gran nombre d'interessats.

HDLC: High level Data Link Control (Control d’enllaç de dades d’alt nivell)

HMI: Human Machine Interface (Interfície màquina-humà)

I/O: Input/Output (Entrada/Sortida)

IP: Internet Protocol (Protocol d’internet)

LSB: bits menys significatius.

MAC: Media Access Control (Control d’accés als mitjans)

MBAP: Encapçalat de Protocol d'Aplicació Modbus

MSB: bits més significatius.

OSI: El model OSI (Open System Interconnection) o també dit la pila de capes o nivells del model OSI, és un model de referència ideal per als protocols de la xarxa, els quals estan separats d'acord amb les seves funcions.

PLC: Controlador de lògica programable

RS-232: és una interfície que designa una norma per a l'intercanvi en sèrie de dades binàries entre un equip terminal de dades i un DCE, tot i que existeixen altres situacions on pot ser emprat.

Introducció

L'objectiu del projecte és el desenvolupament d'una eina hardware/software que permeti la comunicació amb qualsevol mena d'instrument Modbus, independentment del fabricant o les variables que mesuri. Ja que molts equips de mesura i instrumentació utilitzen per a la seva configuració i comunicació el protocol Modbus, bé mitjançant bus de comunicacions RS-485 o Ethernet.

La problemàtica ve donada en la incompatibilitat de poder fer servir la mateixa aplicació o sistema dins d'un mateix entorn que tingui diferents instruments. Aquest inconvenient ve donat perquè la informació dels instruments s'organitza mitjançant un mapa de memòria Modbus, que és diferent per a cada tipus d'instrument i fabricant. Per resoldre aquest problema es durà a terme una descripció en un format estàndard del mapa de memòria Modbus de cada instrument.

Per poder dur a terme l'objectiu es realitzarà un estudi exhaustiu del protocol Modbus per poder comprendre bé el seu funcionament i els trets característics per tal d'assolir l'objectiu proposat.

Pel que fa a la programació de l'aplicació es farà amb el llenguatge de programació gràfica LabVIEW, també s'haurà de fer un estudi exhaustiu de les diferents funcionalitats que ofereix i es necessiten per poder crear una aplicació que pugui dur a terme la interconectivitat de dispositius amb Modbus.

Per a la validació de l'eina creada amb LabVIEW es proposa un set d'instrumentació basat en diferents cabalímetres de diferents fabricants que seran tots ells instal·lats en un banc d’assajos per a realitzar la comparació dels valors obtinguts amb els paràmetres descrits pels fabricants de cada dispositiu. Després de la configuració adient i la correcta execució del programa, l'objectiu és que es puguin visualitzar en un mateix entorn les variables que s'escullin donats cabalímetres de diferents fabricants amb els seus respectius mapatges de memòria completament diferents.

1 Historia Modbus

Modbus és un protocol que es troba estretament relacionat amb els primers PLC (Controlador Lògic Programable). Ja que al final de la dècada dels 60 tots els sistemes de control de la indústria es duien a terme amb el que es denominava lògica cablejada basant-se en la utilització de relés electromagnètics, contactors de potència, relés temporitzats i molts altres components utilitzats pel control de sistemes de lògica combinacional. El problema d'aquest sistema era la gran complexitat dels dissenys necessaris per a realitzar les instal·lacions i òbviament lligat a aquest motiu també el difícil manteniment d'aquestes. El problema no venia de la dificultat tècnica en realitzar les instal·lacions sinó en el temps de dedicació a part del disseny pel cablejat necessari per dur-la a terme, a més que hem de considerar que aquests sistemes són poc escalables.

Després d'aquests sistemes de lògica cablejada l'any 1968 GM Hydramatic (la divisió de transmissió automàtica de General Motors) va emetre una sol·licitud de propostes per a un reemplaçament electrònic dels sistemes cablejats de relés. La proposta guanyadora va venir de Bedford Associates. El resultat va ser el primer PLC, designat 084 perquè era el projecte de Bedford Associates núm. 84. Bedford Associates va començar una nova empresa dedicada al desenvolupament, fabricació, venda i manteniment d'aquest nou producte: Modicon (Modular Digital Controler). Una de les persones que van treballar en aquest projecte va ser Dick Morley, qui és considerat com el “pare” del PLC. La marca Modicon va ser venuda en 1977 a Gould Electronics, i posteriorment adquirida per la companyia alemanya AEG i després per la francesa Schneider Electric, l'actual propietari.

Amb els equips Modicon cada cop més presents a la indústria, sorgeix la imminent de comunicar-los entre ells. És aquí on es crea el MODicon Bus més popularment conegut com a Modbus. Com hem esmentat anteriorment, al 1977 Modicon és adquirida per Gould Electronic, que posteriorment al 1989 és comprada per AEG i finalment adquirida al 1994 per Groupe Schneider qui a més, l'any 1999 canvia el nom a Schneider Electric.

Al món dels protocols, a grans trets es podria dir que hi ha dos tipus de classificacions

on diferenciaríem protocols oberts i protocols per propietaris privats. Avui dia Modbus és un protocol de tipus obert gràcies al fet que l'any 2004 la companyia Schneider Electric va transferir els drets a la Modbus Organization, la missió del qual és impulsar l'adopció del conjunt de protocols Modbus i l'evolució per abordar arquitectures per sistemes d'automatització distribuïts en múltiples segments del mercat. L'Organització Modbus també proporciona la infraestructura per a obtenir i compartir informació sobre els protocols, la seva aplicació i certificació per a simplificar la implementació per part dels usuaris i reduir els costos. Actualment " The Modbus Organization" es basa en una afiliació-associació de comerç, incorporat com "Modbus Organització, Inc." sota les lleis de la Commonwealth de Massachusetts, EUA i homologat pels EUA Servei d'Ingressos Interns com a organització sense ànim de lucre.

2 Què és Modbus?

Després del repàs històric de Modbus, en aquest apartat es comença a concretar una mica més sobre el protocol, els seus tipus, l'arquitectura i altres punts d'interès amb la informació extreta de Modbus Organization (2012).

És un protocol de sol·licitud-resposta ubicat a les capes 1 física, 2 enllaços de dades i 7 d'aplicació del Model OSI, basat en una arquitectura mestre/esclau o client/servidor que s'ha convertit en un protocol de comunicació estàndard de facto en la indústria a causa del seu ampli espectre de funcionalitats i la seva capacitat d'adaptació. Centenars de proveïdors ho han implementat en milers de dispositius diferents per a transferir Entrades / Sortides discretes o analògiques i registrar dades entre dispositius de control. Per posar un exemple de la seva gran magnitud d'utilitat, els analistes de la indústria han reportat més de 7 milions de nodes Modbus només a Amèrica del Nord i Europa.

L'estructura d'aquest protocol gira entorn dels missatges dels equips físics i no als equips físics en si mateixos o al mitjà de comunicació, és a dir, que el mateix tipus de missatge que s'utilitzava abans via RS-232 és el mateix que s'utilitza avui dia via TCP/IP amb l'excepció que va inserit en la capa d'aplicació de l'estàndard OSI. És aquí on radica el secret de la durabilitat de Modbus que, a diferència d'altres protocols, no està segmentat i ha desenvolupat la capacitat d'adaptació a les noves tecnologies informàtiques mantenint la senzillesa dels missatges des de fa dècades.

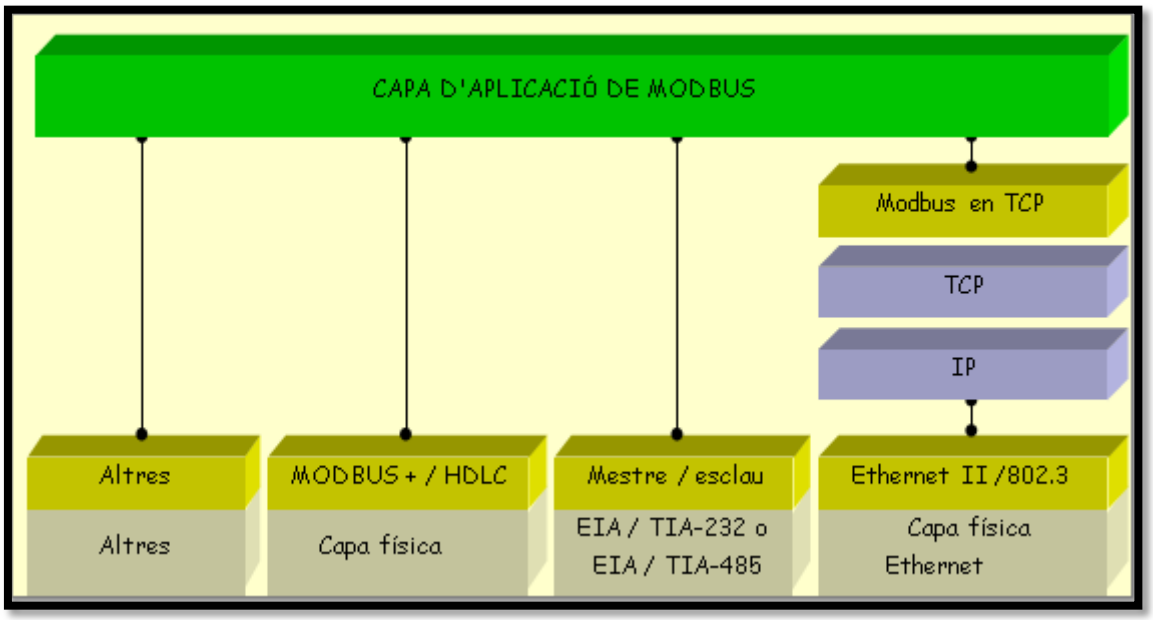


Figura 1. **Pila de comunicacions Modbus**

El protocol Modbus permet una comunicació fàcil dins de tota classe d'arquitectures de xarxa. Tota mena de dispositius (PLC, HMI, tauler de control, controlador, control de moviment, dispositiu d'E/S ...) poden utilitzar el protocol Modbus per iniciar una operació remota.

La mateixa comunicació es pot fer tant en sèrie com en xarxes TCP/IP Ethernet. També hi ha l'afegit que les passarel·les permeten una comunicació entre diversos tipus de busos o xarxa mitjançant el protocol Modbus.

2.1 Versions

A la implementació inicial era un protocol construït en base serial que no podia ser dividit en múltiples capes, però amb el temps diferents unitats de dades d'aplicació van anar introduint-se tant per canviar el paquet del format utilitzat a través de serial com per permetre la utilització de xarxes TCP/IP i UDP, amb el qual han sorgit diverses versions de Modbus. En els següents apartats s'explicarà resumidament algunes versions més conegudes i en més profunditat les versions RTU i TCP/IP que són les que s'implementaran al projecte amb més profunditat.

Pel que fa a les variants que s'expliquen a continuació és important saber que el model de dades i la crida de funcions de Modbus RTU, Modbus TCP/IP, Modbus ASCII i Modbus sobre TCP/IP són idèntiques, només l'encapsulació és diferent. No obstant això, les variables no són interoperables i el seu format de trama tampoc.

2.1.1 Enron Modbus

Enron Modbus és una modificació del protocol estàndard de comunicació Modicon Modbus. Va ser desenvolupat per Enron Corporation. Les principals diferències entre els dos protocols és la enumeració de les direccions de registre, el suport de registres de 32 bits i 16 bits, i la capacitat de transmetre registres d'esdeveniments i dades històriques.

2.1.2 Pemex Modbus

Aquesta és una extensió del Modbus estàndard amb suport per a dades històriques i de flux. Va ser dissenyada per a la companyia de petroli i gas de Pemex per al seu ús en el control de processos i mai va obtenir una adopció generalitzada.

2.1.3 Modbus ASCII

S'utilitza en la comunicació en sèrie i fa ús de caràcters ASCII per a la comunicació del protocol. El format ASCII utilitza una suma de verificació de redundància longitudinal. Els missatges Modbus ASCII estan emmarcats per dos punts inicials (":") i una nova línia final (LCR).

2.1.4 Modbus Plus

Modbus Plus és propietat de Schneider Electric i, a diferència de les altres variants, admet comunicacions peer-to-peer entre múltiples mestres. Requereix un coprocessador dedicat per a manejar la rotació ràpida de tokens similars a HDLC. Utilitza un parell trenat a 1 Mbit/s i inclou aïllament de transformador en cada node, la qual cosa fa que l'activació sigui per transició/flanc en lloc de ser activat per voltatge / nivell. Es requereix maquinària especial per a connectar Modbus Plus a una computadora, generalment una targeta feta pel bus ISA, PCI o PCMCIA.

2.1.5 Modbus sobre UDP

Alguns han experimentat amb l'ús de Modbus sobre UDP en xarxes IP, la qual cosa elimina les despeses generals necessàries per a TCP. Això vol dir que a nivell d'especificació no canvia res, si no que la diferencia rau en com es transporten els missatges.

2.1.6 Modbus sobre TCP/IP

Aquesta és una variant de Modbus que només difereix del Modbus TCP en què s'inclou una suma de comprovació en la càrrega útil com en Modbus RTU.

2.1.7 Modbus TCP/IP

Es tracta d'una variant Modbus utilitzada per a comunicacions a través de xarxes TCP/IP, connectant-se a través del port 502. No requereix un càlcul de suma de verificació ja que les capes inferiors ja proporcionen protecció. Aquest tipus de model està basat en 4 tipus de missatge:

- **Sol·licitud Modbus:** és el missatge enviat per la xarxa de l'esclau al mestre per iniciar la transmissió.
- **Indicació Modbus:** és el missatge de la sol·licitud rebuda a la part del mestre.
- **Resposta Modbus**: és el missatge de resposta enviat pel mestre a l'esclau.
- **Confirmació Modbus:** és la resposta Modbus rebuda de la part de l'esclau.

Aquest tipus es pot utilitzar tant per dos dispositius amb aplicació, aplicacions HMI/SCADA amb altres dispositius o simplement una aplicació de PC i un programa de dispositiu que brinda serveis en línia.

2.1.8 Modbus RTU

Aquest protocol mostra les dades en una vista binària compacta. RTU comprova els errors de la informació seguint els comandos amb una suma de comprovació de redundància cíclica. Els bytes s'envien consecutivament sense espais entre ells amb un espai de 3,5 caràcters entre els missatges com a delimitador. Així és com el programari sap que s'ha iniciat un nou missatge. Els missatges de RTU estan emmarcats per períodes d'inactivitat.

3 Marc general Modbus

3.1 Descripció del protocol

Com s'ha esmentat anteriorment, és un protocol de sol·licitud-resposta implementat usant una relació mestre-esclau. En una relació mestre-esclau, la comunicació sempre es produeix en parells, un dispositiu ha d'iniciar una sol·licitud i esperar una resposta. El dispositiu d'inici (el mestre) és el responsable d'iniciar cada interacció. En general, el mestre és una interfície humà-màquina (HMI) sistema SCADA i l'esclau és un sensor, controlador lògic programable (PLC) o controlador d'automatització programable (PAC). El contingut d'aquestes sol·licituds i respostes, i les capes de la xarxa a través de les quals s'envien aquests missatges, són definides per les diferents capes del protocol.

3.2 Unitat de dades de protocol (PDU)

El protocol Modbus defineix una unitat de dades de protocol simple (PDU) independent de les capes de comunicació subjacents que, en conjunt, el codi que la maneja són considerats com el nucli de les especificacions del protocol. Aquestes especificacions defineixen el format de la PDU, els diversos conceptes de dades utilitzats en el protocol, l'ús dels codis de funció per obtenir accés justament a aquestes dades i la implementació especifica, a més de les restriccions de cada codi de funcions. Fins aquí és com era la implementació inicial, ja que va ser construït en base serial, pel que no es podia dividir en més capes. Amb el pas dels anys i la seva evolució es van anar introduint diferents ADU's, ja fos per canviar el format del paquet serial utilitzat o per permetre l'ús de xarxes TCP o UDP. Aquestes noves incorporacions van portar a una separació del protocol principal, la PDU i la capa de xarxa que ve més definida per l'ADU. L'assignació del protocol Modbus en bus específics o xarxa pot introduir alguns camps addicionals a la unitat de dades d'aplicació (ADU).

Com es pot observar a la figura 2, la PDU no és més que un codi de funcions seguit d'un conjunt de dades associades. La grandària i el contingut d'aquestes dades són definides pel codi de la funció i també la PDU en si mateixa, que seria el codi de la funció més les dades.

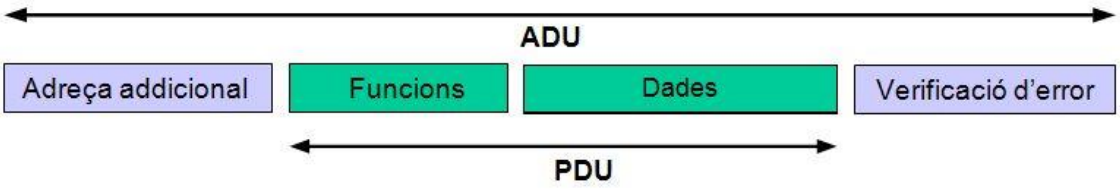


Figura 2. **Marc general Modbus**

Les dades de l'aplicació Modbus les construeix el mestre que vol iniciar la transmissió de dades. Com hem especificat abans, les funcions són les indicades d'ordenar a l'esclau quin tipus d'acció ha de realitzar. En l'apartat de funcions de la trama a enviar, la unitat de dades Modbus és d'un byte, que vol dir que es poden representar en un rang de valors d'1 a 255. Amb l'excepció del rang de valors establerts entre el 128 i el 255, que estan reservats i utilitzats per respondre les excepcions que puguin sorgir durant la comunicació. Una altra funció que tampoc és vàlida i que no es pot utilitzar és el codi "0".

3.3 Unitat de dades d'aplicació (ADU)

A més de la funcionalitat definida a la PDU principal del protocol Modbus es poden utilitzar diferents protocols de xarxa. Els més comuns son serial i TCP/IP, que seran els que tractarem per a les comunicacions del projecte. Per transmetre les dades necessàries per Modbus a través d'aquestes capes, Modbus té un conjunt de variants ADU que són dissenyades per a cada protocol específic.

3.3.1 Característiques comunes

Modbus ha de tenir algunes característiques comunes per poder garantir que la comunicació sigui fiable. L'adreça o direcció és usada en cada format d'ADU per proporcionar informació sobre l'encaminament a la capa de l'aplicació. Cada ADU conté una PDU completa a la qual inclou la informació de comprovació d'errors. La conclusió és que totes les ADU's proporcionen un mecanisme per determinar l'inici i el final d'un marc de sol·licitud, però els implementa de manera diversa.

3.3.2 Formats estàndards

Els tres formats estàndards més utilitzats són TCP, unitat de terminal remota (RTU) i ASCII. Els ADU's de la unitat de terminal remota (RTU) i ASCII normalment són fets

servir a partir de la línia serial, la qual cosa vol dir que s'envia un bit cada vegada i de forma seqüencial, com per exemple RS-485, RS-232 o fibra òptica. El format d'ADU TCP és utilitzat a través de xarxes TCP/IP o UDO/IP modernes, com per exemple Ethernet, xarxa sense fil IEEE 802.11 i també fibra òptica.

En aquests formats de capa física, l'establiment de la informació i enviament de dades pot estar configurat com a semidúplex o també conegudes com a half-dúplex, és a dir, les dades només flueixen en una direcció o l'altra (de mestre a esclau o viceversa), mai en les dues direccions al mateix temps. Existeix també la configuració dúplex, altrament nomenada full-dúplex, que consisteix en l'enviament i la recepció de les dades de manera simultània en el mateix equip. Un exemple seria un mestre que envia una sol·licitud a un esclau mentre està rebent una resposta del mateix esclau.

3.3.3 Protocol de Control de la transmissió (TCP)

Totes les sol·licituds i respostes Modbus estan dissenyades de tal manera que el destinatari pugui verificar que un missatge hagi finalitzat.

Pels codis de funció on la PDU Modbus té una longitud fixa, el codi de funció només és suficient, mentre que pels codis de funció que porten una quantitat variable de dades en la sol·licitud o resposta, el camp de dades inclou un recompte de bytes. També cal destacar que els camps a aquesta ADU estan codificats en format Big-endian[1].

Segons el document d'especificacions Modbus traduït al castellà: "el servei de missatgeria Modbus ha de proporcionar una presa d'escolta en el port 502, que permet acceptar noves connexions i intercanviar dades amb altres dispositius.

Quan el servei de missatgeria necessita intercanviar dades amb un servidor remot, ha d'obrir una nova connexió de client amb un port remot 502 per a intercanviar dades amb aquesta distància. El port local ha de ser superior a 1024 i diferent per a cada connexió de client.

---

[1] "Sistema d'adreçament de memòria en el qual el més alt valor significatiu en una seqüència de números és emmagatzemat primer a la part més baixa de la memòria" (Alicia S. Clark i Eduardo H. Clark 2004, traduït del castellà)

Si el nombre de connexions de client i servidor és major que el nombre de connexions autoritzades, es tanca la connexió no utilitzada més antiga. El mecanisme de control d'accés es pot activar per a verificar si l'adreça IP del client remot està autoritzada. Si no, es rebutja la nova connexió. S'ha d'enviar una sol·licitud Modbus en la connexió TCP correcta ja oberta. L'adreça IP del control remot s'usa per a trobar la connexió TCP. En cas que s'obrin diverses connexions TCP amb el mateix control remot, s'ha de triar una connexió per a enviar el missatge Modbus, es poden usar diferents criteris d'elecció com el més antic o el primer en connectar. La connexió ha de mantenir-se oberta durant totes les comunicacions Modbus. Com es descriu en les següents seccions, un client pot iniciar diverses transaccions Modbus amb un servidor sense esperar el final de l'anterior.

Quan les comunicacions Modbus finalitzen entre un Client i un Servidor, el client ha d'iniciar un tancament de connexió utilitzada per a aquestes comunicacions.”(Modbus Organization 2006a, traduït de l'anglès)

Les ADU’s de TCP consisteixen en l’encapçalat de protocol d’aplicació Modbus (MBAP) combinat amb la PDU de Modbus que es transporta per una xarxa TCP/IP. El MBAP és un encapçalat d’ús general que depèn d’una capa de xarxa fiable.

Els apartats de dades de l’encapçalat indiquen el seu ús. En els següents punts es citen els diferents camps pel qual es compon el MBAP:

- En primera posició inclou un **identificador de transacció**. Aquest camp és bastant important en una xarxa, la qual pot suportar múltiples sol·licituds simultàniament, és a dir, un mestre pot enviar sol·licituds de dades a diferents esclaus. Els esclaus no tenen per què respondre en el mateix orde que han estat fetes les sol·licituds, amb l’identificador de transacció el mestre pot igualar les sol·licituds amb les respostes i així analitzar les dades amb precisió. Aquesta funcionalitat és molt útil per les xarxes Ethernet.
- **L’identificador del protocol** és normalment nul però es pot utilitzar per ampliar el comportament del protocol, com per exemple fer passar diversos senyals per un mateix canal de transmissió.
- El camp de **longitud** s’utilitza al protocol per delinear la longitud de la resta del paquet de la trama. La ubicació d’aquest element també indica la dependència d’aquest format d’encapçalament en una capa de xarxa fiable. A causa del fet

que els paquets TCP tenen verificació d'errors integrada i garanteixen la coherència de les dades i també la seva entrega.

La longitud del paquet pot ser ubicada en qualsevol part de l'encapçalament. Una xarxa de tipus serial és menys fiable ja que un paquet podria perdre's, tenint l'efecte de què inclús si l'escriptura de dades llegida per l'aplicació creada incloïa informació vàlida de transacció i de protocol, la informació de la longitud alterada tornaria invàlid l'encapçalament. El TCP proporciona una protecció raonable contra aquesta situació amb el camp de longitud per poder evitar el màxim possible aquesta situació.

- El camp de **la unitat ID** generalment no és utilitzada pels dispositius TCP/IP. Com al protocol Modbus és comú que s'implementin passarel·les, converteix el protocol TCP/IP en un altre protocol. Una passarel·la és un dispositiu que permet interconnectar xarxes que utilitzen arquitectures o protocols diferents, és una porta d'enllaç en una xarxa. Per exemple, l'ús d'una passarel·la de TCP/IP a serial per poder permetre connexions de noves xarxes TCP/IP amb xarxes anteriors que es poguessin tenir i utilitzessin connexions serial. En un entorn com el que acabem de mencionar, la unitat ID és utilitzada per determinar la direcció del dispositiu esclau per la qual la PDU està destinada a arribar.

La mida de la PDU de Modbus està limitada per la primera implementació de la primera xarxa Modbus serial feta amb RS-232, on la seva ADU és de 256 bytes. Per tat, la PDU de comunicació per serial és de 256 bytes i si es resta el byte de l'adreça de l'esclau i els dos bytes del CRC es queda amb 253 bytes. A conseqüència d'això, si a aquesta PDU inicial se n'hi sumés el MBAP de 7 bytes, fa un total de 260 bytes, que és la mida de l'ADU del protocol TCP.

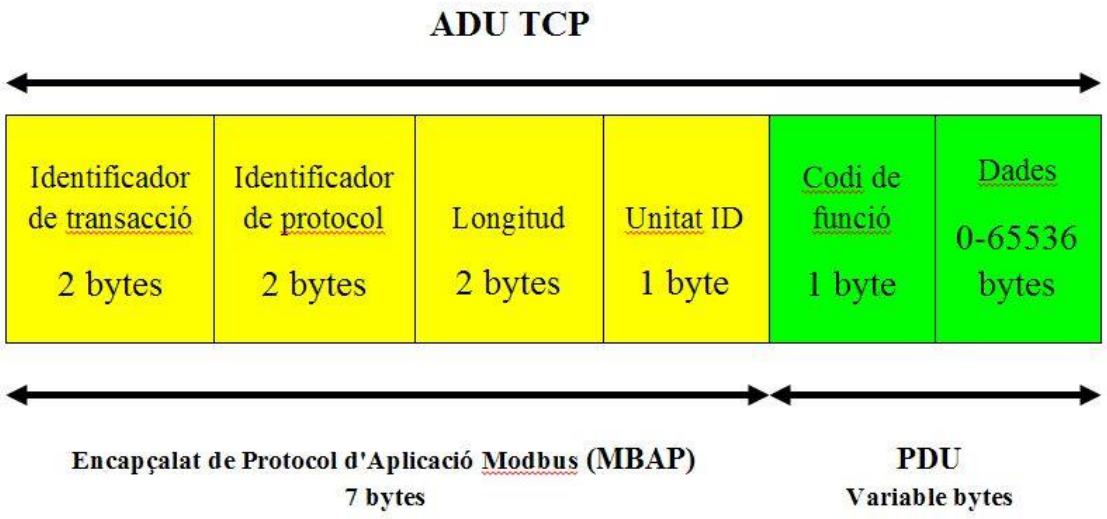


Figura 3. **Unitat de Dades d'Aplicació TCP**

3.3.4 Unitat terminal remota (RTU)

L'ADU de la unitat terminal remota és més simple que l'anterior. A diferència de l'ADU de TCP, que és més complexa, aquest tipus d'ADU inclou dos camps més d'informació a part de la PDU.

Pel que fa al camp de la direcció Modbus, en el cas de les trames enviades pel mestre, el camp de número d'esclau indica l'adreça del destinatari d'aquesta trama. Permet adreçar fins a 247 esclaus, amb les direccions d'1 a 247 (0x01 a 0xF7). El 0x00 és pels missatges de broadcast[2], així, el primer esclau comença amb la direcció 1 (d'1 a 247) ja que les adreces de 248 i 255 estan reservades. En el cas de les trames enviades pels esclaus, aquest byte serveix per a indicar al mestre a qui pertany la resposta. És a dir, cada vegada que un esclau respon, situa la seva pròpia direcció en el byte de direcció cosa que permet saber al mestre quin equip correspon cada resposta.

Normalment una gran part de les xarxes defineixen la direcció 0 com a broadcast. Això implica que un mestre pot enviar una sol·licitud a la direcció 0 i tots els esclaus

[2] "Una adreça de difusió àmplia o broadcast és aquella adreça IP que permet la transmissió de dades a una multitud de nodes receptors continguts en una mateixa subxarxa i de manera simultània."(Schneider Electric 2020, traduït del castellà)

connectats han de processar la informació però cap ha de respondre. Aquesta funcionalitat però, no l'admeten totes les implementacions de Modbus.

S'utilitza un sistema de detecció d'errors que depenen del tipus de codificació utilitzada. En el cas de RTU el mètode empleat és el CRC. Aquest mètode consisteix en el fet que el transmissor aplica un algoritme a la trama i dóna com a resultar un camp de 2 bytes, els quals s'inclouen com a últim camp a la trama. D'aquesta manera el receptor de la trama aplica el mateix algoritme prèviament definit i comprova si el resultat coincideix amb el camp CRC que el transmissor inclou a la trama enviada. En cas que no coincideixi es produeix un error.

El càlcul de CRC s'inicia carregant un registre de 16 bits (2 bytes) on tots són uns. Tot seguit comença un procés d'aplicació successiva de 8 bits del missatge als continguts actuals del registre. Només els vuit bits de dades en cada caràcter s'utilitzen per a generar el CRC els bits d'inici i parada, el bit de paritat no s'aplica al CRC. S'aplica una OR[3] amb cadascun dels caràcters de 8 bits amb el contingut de cada byte i el resultat es desplaça un bit a l'esquerra inserint un 0 en la posició de menys pes (la de la dreta). El de l'esquerra s'extreu i s'examina: si és 1 es torna a fer OR amb un valor prefixat, si és 0 no es fa cap OR. Aquest procés es repeteix fins que s'hagin realitzat vuit torns. Després de l'últim (vuit) torn, el següent byte de 8 bits es fa un OR exclusiu amb el valor actual del registre i el procés es repeteix durant vuit torns més com es va descriure anteriorment. El contingut final del registre, després que tots els bytes del missatge han estat aplicats, és el valor CRC que es compara amb l'obtingut de l'emissor per la comprovació d'errors. A l'annex A es pot veure un exemple detallat de la generació de CRC.

Avui dia, les noves implementacions del protocol a l'encapsulat dels paquets de dades hi ha un parell de temps de silenci, això vol dir que dins el bus de comunicacions hi ha temps on no hi ha comunicació. Aquests silencis s'utilitzen a la línia de transmissió per

[3] Com expliquen al seu llibre Harris Money i Harris L. (2007) una porta OR és una porta lògica digital que implementa la disjunció lògica. Quan totes les seves entrades estan en 0 (zero) o en baixa, la seva sortida està en 0 o en baixa, mentre que quan almenys una o totes dues entrades estan en 1 o en alta, la seva sortida estarà en 1 o en alta.

indicar l'inici i el final dels missatges. En l'exemple d'un RTU configurat a 9600 bauds, el temps dels silencis oscil·la al voltant dels 4ms. Per definició de l'estàndard Modbus la longitud mínima dels silencis, independentment de la velocitat de transferència dels paquets, ha de ser 3,5 vegades el temps que es triga en l'enviament d'un caràcter.

Un dels inconvenients de la implementació d'aquest protocol és el rendiment, ja que el dispositiu sigui mestre o esclau ha d'esperar que el temps mort generat pels silencis es compleixi abans de poder processar el paquet rebut. També hi ha una dificultat afegida a la introducció de noves tecnologies utilitzada per la transferència serial i velocitats molt més elevades que quan es va crear l'estàndard. Una possibilitat clara és un convertidor d'USB a serial perquè no es té cap control sobre el paquet i la transferència de les dades.

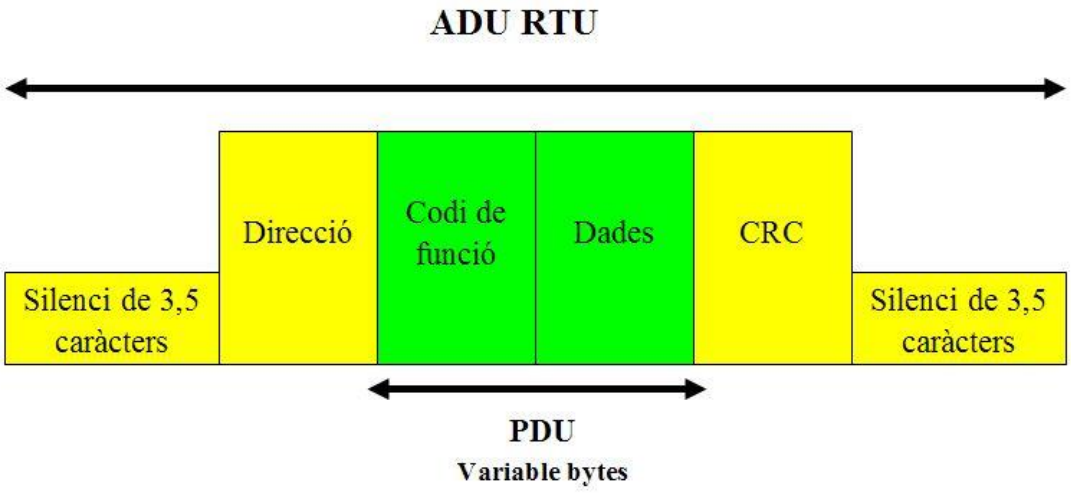


Figura 4. **Unitat de Dades d'Aplicació RTU**

3.3.5 Serial ASCII

Quan els dispositius tenen la configuració per comunicar-se amb la metodologia ASCII, cada pac de 8 bits d'un missatge s'envia com dos caràcters ASCII. Aquest mètode s'utilitza quan l'enllaç de comunicació física o les capacitats del dispositiu no permeten la conformitat respecte a l'administració dels temporitzadors. Aquests mètode és menys eficient que el RTU, ja que cada byte necessita dos caràcters. Per exemple per representar el byte 0x2F s'ha de codificar dos caràcters el 0x32 per representar el 2 i el valor 0x46 per codificar la F.

A diferència de la manera de transmissió RTU, ASCII permet intervals de fins i tot un segon entre caràcters. Sempre que l'usuari que el faci servir no hagi configurat un temporitzador major, es considerarà erroni un interval de més d'un segon entre caràcters. Es pot donar que en algunes aplicacions on la xarxa és molt extensa s'arribin a utilitzar fins i tot intervals de temps d'entre quatre o cinc segons entre caràcters.

Amb el mètode ASCII les trames enviades es delimiten amb caràcters en comptes de silencis. El caràcter d'inici són dos punts (:) que amb terminologia ASCII és 0x3A i els caràcters de final de trama són "retorn de carro" (CR) amb representació ASCII 0x0D i també salt de línia (LF) representat per 0x0A.

Els caràcters permesos en tots els camps de la trama són hexadecimals de 0-9 i de A-F, codificats amb ASCII. Els dispositius monitoritzen el bus de comunicacions contínuament fins que es troben amb el caràcter d'inici ":" i a partir d'aquest moment descodifiquen tots els caràcters següents fins que es detecten els caràcters de final de trama. A la figura 5 es mostra un marc de missatge típic.

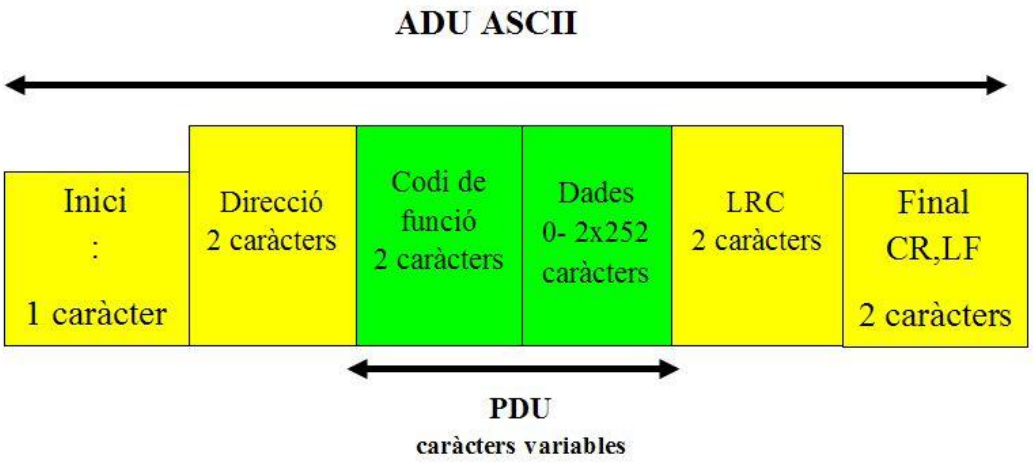


Figura 5. Unitat de Dades d'Aplicació ASCII

La direcció funciona com el cas anterior de la RTU però amb la representació amb caràcters ASCII.

Per la comprovació d'errors de transmissió es fa servir el mètode LRC. El càlcul del

LRC es fa en els continguts dels missatges, exceptuant els dos punts d'inici i el parell de caràcters de terminació CR/LF. S'aplica independentment de qualsevol mètode de comprovació per paritat utilitzat pels caràcters individuals del missatge.

El camp de LRC és un byte que conté 8 bits amb valors binaris. El valor del LRC és calculat pel dispositiu emissor del missatge, que després afegeix el LRC al missatge. A continuació el dispositiu receptor el calcularà quan rebi el missatge i comprovarà el càlcul del LRC amb el valor rebut de l'emissor, si coincideixen voldrà dir que és correcte i si no coincideixen, indicarà un error.

Per calcular-ho s'ha d'afegir successivament bytes de 8 bits al missatge, descartant qualsevol pèrdua de dades i complementar posteriorment el resultat. Es duu a terme en els bytes del missatge abans de codificar cada byte amb els seus corresponents caràcters ASCII. La computació com hem mencionat abans no implica els dos punts d'inici i els camps de finalització de CR/LF. El LRC resultant de la codificació se situa en dos bytes al final de la trama ASCII abans del CR/LF.

Per a més detalls d'un exemple de generació de LRC consultar l'annex A.

3.4 Comprovació d'errors

A Modbus hi ha diverses maneres d'implementar la comprovació d'errors, en els apartats consecutius s'explica tant per la part serial com per la part TCP.

3.4.1 Errors mètode serial

Comprovació de paritat: a la trama s'aplica a cada byte què és transmès, de manera que l'emissor calcula per a cada byte el bit de paritat i ho adjunta al missatge. El receptor calcula la paritat del missatge i comprova que coincideixi amb el bit de paritat que l'emissor ha adjuntat al missatge en el moment en que rep la trama. En cas que no sigui així i no ho pugui dur a terme, el receptor ignorarà el missatge. Això comporta que l'esclau no respongui al mestre i el temporitzador del mestre expiri.

Aquest tipus de verificació només és eficaç en els casos que el nombre de bits erronis sigui senar, ja que si fos parell no canviaria la paritat del missatge. I si el mètode de paritat no és seleccionat per la creació de la trama del protocol, no es comprova en cap

moment el bit de paritat i se substitueix per un altre bit de stop.

Comprovació de CRC/LRC: en aquest cas es com s'explica anteriorment a l'apartat 4.3.4 i 4.3.5 la verificació s'aplica a la trama completa. És un algoritme aplicat per l'emissor de la trama (sense incloure delimitadors) que dóna com a resultat un camp de 2 bytes i s'inclou a la trama prèviament a ser enviada. D'aquesta manera el receptor de la trama aplica el mateix algoritme i pot comprovar si el seu resultat coincideix amb l'enviat per l'emissor, si no coincideix la trama és ignorada pel receptor. Com en el cas anterior, l'esclau no respondria a la sol·licitud del mestre i un cop expirat el temporitzador, el mestre prendria la decisió que cregués oportuna per la situació.

3.4.2 Error mètode TCP

Quan es rep un marc de resposta en una connexió TCP, l'identificador de transacció que es troba en l'encapçalat MBAP s'usa per associar la resposta amb la sol·licitud original enviada prèviament en aquesta connexió TCP: si l'identificador de transacció no es refereix a cap transacció pendent de Modbus, la resposta ha de ser descartada.

Quan Modbus es transporta a través de TCP, es transporta informació de longitud addicional en l'encapçalat MBAP per a permetre que el destinatari reconegui els límits del missatge fins i tot si el missatge s'ha dividit en múltiples paquets per a la seva transmissió. L'existència de regles de longitud explícites i implícites, i l'ús d'un codi de verificació d'error CRC-32 (en Ethernet) dóna com a resultat una possibilitat infinitesimal de corrupció no detectada en un missatge de sol·licitud o resposta.

3.5 Model de dades

Quan un camp de dades de la trama és enviat des d'un mestre a l'esclau, els dispositius contenen informació addicional que l'esclau utilitza per obtenir l'acció que ve definida per la part de la funció de la trama enviada. Aquesta part de dades pot incloure, per exemple, adreces discretes i de registre, la quantitat d'elements que s'han de manejar o el nombre actual de bytes que hi ha al camp de dades de la trama. També pot existir la possibilitat que el camp de dades estigui buit i no contingui cap bit en determinats tipus de sol·licituds. En aquest cas a l'esclau no li serà necessari cap mena d'informació addicional com de les que s'han esmentat abans. En aquests casos la part de la trama de la funció només explica per codi l'acció a realitzar i res més.

Si no es produeix cap tipus d'error relacionat amb la funció Modbus sol·licitada en una ADU rebuda correctament, el camp de dades de la trama d'una resposta d'un esclau a un mestre contindrà les dades sol·licitades. Però, si hi ha qualsevol error que tingui a veure amb les funcions de Modbus sol·licitades, el camp de dades de la trama contindrà codi d'excepcions que com hem dit abans hi ha reservat des del rang 128 fins al 255 per gestionar-les i així l'aplicació del mestre pot utilitzar-les per decidir quina és la següent acció a seguir si això succeeix. Quan l'esclau respon al mestre que li ha fet la sol·licitud, utilitza el camp de funcions de la trama per indicar una resposta normal sense errors però també fa servir el mateix camp per respondre quan s'ha produït algun tipus d'error, aquesta resposta és anomenada resposta d'excepció. Per obtenir una resposta d'excepció, l'esclau retorna un codi que equival a la funció original enviada en la sol·licitud de la PDU amb el bit més significatiu (MSB) establert a lògica 1.

Un exemple és un client que pot llegir els estats encès/apagat d'un grup de sortides o entrades discretes o també llegir/escriure el contingut de dades d'un grup de registres.

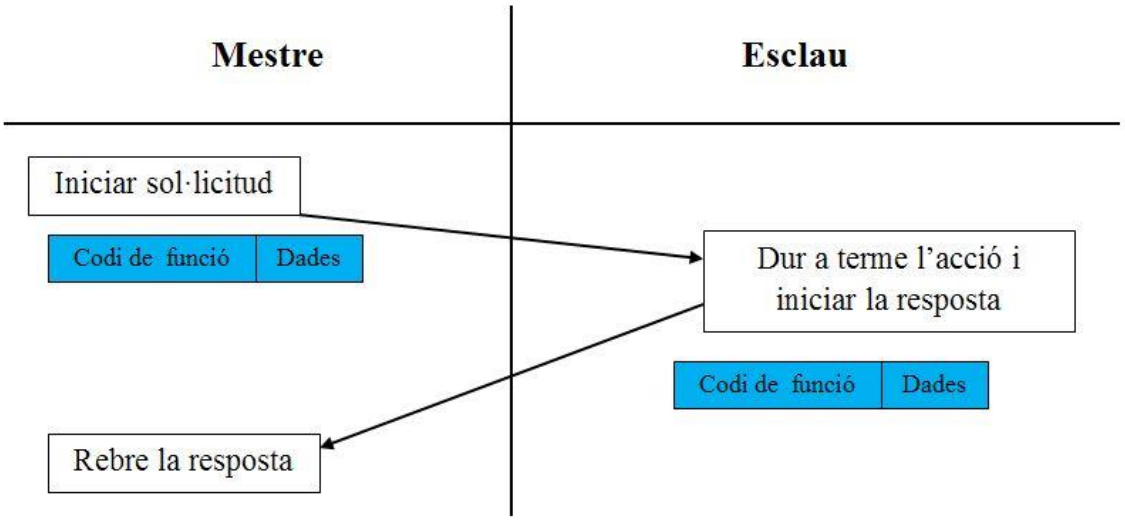


Figura 6. Transmissió de dades sense error

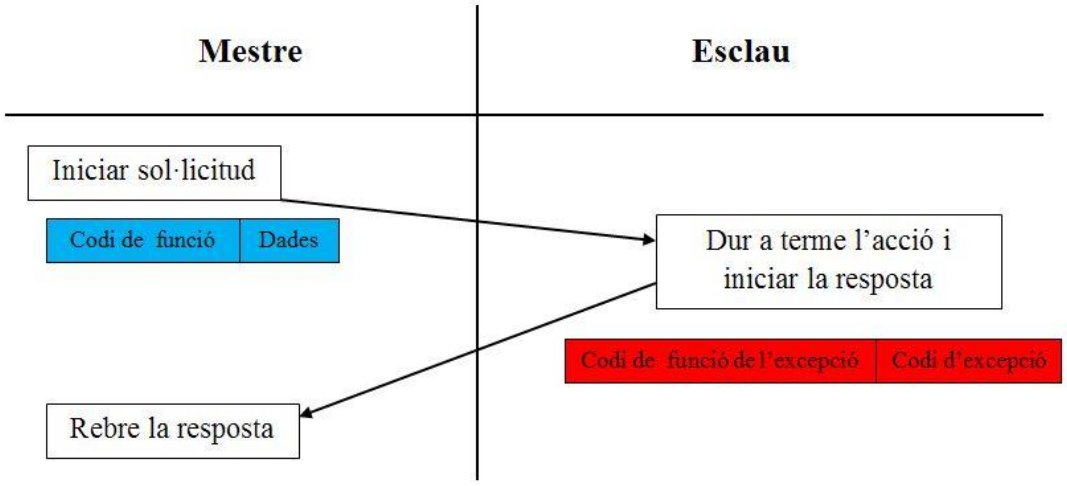


Figura 7. **Transmissió de dades amb resposta d'excepció**

3.5.1 Classificació de dades

Les dades que fa servir el protocol també poden ser classificades en dos tipus segons la seva mida: bits individual i registres de 16 bits (2 bytes). Els bits individuals per entrades i sortides digitals i els registres de 16 bits per variables que necessitin un major espai d'emmagatzematge. A la següent taula s'indica els tipus de dades disponibles:

Tipus d'objecte	Tipus de dades	Accés Mestre	Accés Esclau	Mida
Discret Input	Booleà	Només lectura	Lectura/Escriptura	1 bit
Coil	Booleà	Lectura/Escriptura	Lectura/Escriptura	1bit
Input Registers	Paraula sense signe	Només lectura	Lectura/Escriptura	16 bits
Holding Registers	Paraula sense signe	Lectura/Escriptura	Lectura/Escriptura	16 bits

Taula 1. **Model de dades Modbus**

- **Discret input:** aquests tipus de dada pot ser proporcionada per un sistema d'entrades/sortides.
- **Coil:** aquest tipus de dada pot ser proporcionada per un programa de l'aplicació.
- **Input register:** aquest tipus de dada pot ser proporcionada per un sistema d'entrades/sortides.
- **Holding register:** aquest tipus de dades pot ser proporcionada per un programa de l'aplicació.

3.6 Model d'adreçament Modbus

El protocol d'aplicació Modbus defineix amb precisió les regles d'adreçament PDU.

"En una PDU MODBUS, cada dada s'adreça de 0 a 65535. També defineix clarament un model de dades MODBUS compost per 4 blocs que comprèn diversos elements numerats de l'1 al n. En el model de dades Modbus, cada element dins d'un bloc de dades està numerat de l'1 al n. Posteriorment, el model de dades Modbus ha de vincular-se a l'aplicació del dispositiu". (Modbus Organization 2012, traduït de l'anglès)

El mapatge previ entre el model de dades Modbus i l'aplicació del dispositiu és totalment competència del proveïdor del dispositiu específic. Aquest aspecte és un dels problemes que es plantegen en aquest projecte, ja que no permet la creació d'una sola aplicació estàndard per a poder treballar des de la mateixa aplicació diferents dispositius de fabricants diferents amb el protocol Modbus.

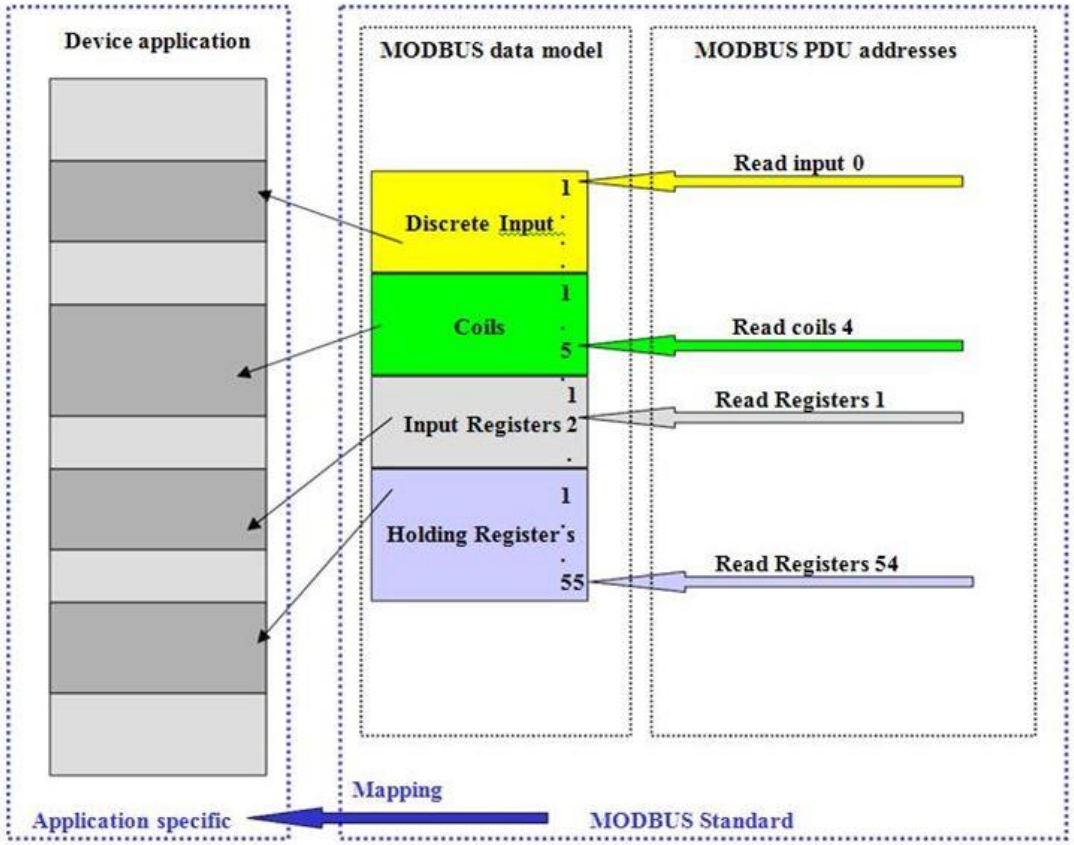


Figura 8. **Mapeig de dades Modbus**

Com s'observa a la figura 8, els rangs de memòria on se situen els diferents tipus de dades van a elecció de l'aplicació i dispositius segons el fabricant. Per tant poden existir trossos de memòria buits o que l'ordre no sigui justament l'especificat.

3.7 Funcions

Ja que el camp de codi de funció mesura un byte i el bit de major pes es fa servir perquè l'esclau indiqui si hi ha hagut error; retornant el mateix codi que se li va sol·licitar amb el bit de major pes a "1" i un byte en el camp de dades indicant el codi d'error que ha tingut lloc, queden disponibles 127 codis de funció. Cada codi de funció es correspon amb una operació diferent.

Existeixen tres tipus de categories per diferenciar els codis de funcions:

- **Públics:** els codis de funció estan molt ben definits, provats, documentats i validats per la comunitat Modbus.org.

- **Definits per l'usuari:** es fan implementacions específiques que necessiti un usuari per la creació de la seva aplicació i que no està suportada per l'estàndard de funcions.
- **Reservats:** alguns fabricants tenen funcions que per algun motiu no interessa que siguin accessibles públicament i les encabeixen en aquest apartat per mantenir la seva privacitat.

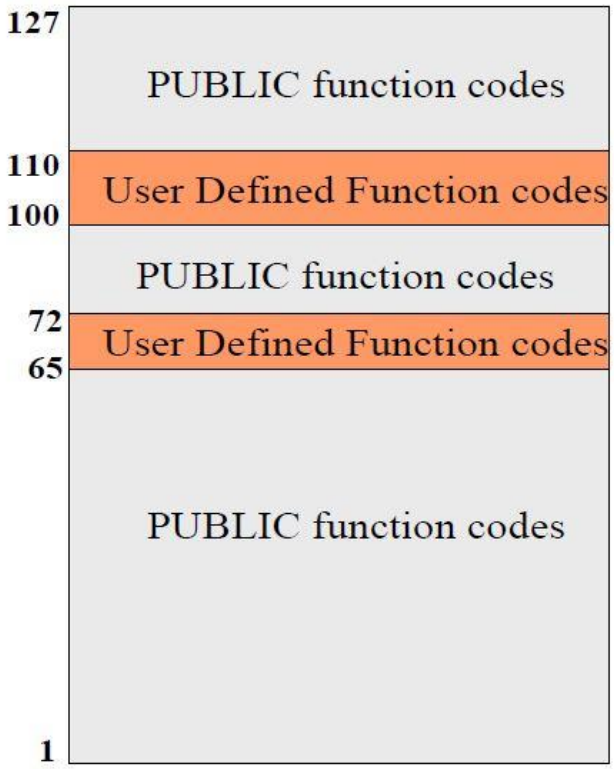


Figura 9. Categories de codis de funcions

3.7.1 Definició de funcions públiques

Es defineixen les funcions públiques més utilitzades segons el tipus d'operació que es desitgi realitzar hi ha de dos tipus:

- **Lectura / escriptura:** serveixen per consultar o escriure dades a la memòria de l'esclau.
- **Control:** permeten realitzar alguna acció sobre l'esclau com podria ser ficar-lo en modalitat de només escolta.

La següent taula indica una mostra de les funcions més utilitzades:

Comandament	Codi de funció
01	Read Coils
02	Read Discrete Inputs
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Register
07	Read Exception Status
08	Diagnostics

Taula 2. Funcions Modbus més utilitzades

A continuació es descriuran mitjançant exemples diferents funcions. Les funcions es poden agrupar de dos en dos ja que són completament equivalents, la única diferencia està en el tipus de dada que llegeixen; un tipus pot també escriure i l'altre no, però pel que fa a la mida, és exactament igual.

3.7.1.1 Read Coils(01) – Read Discrete Inputs (02)

Aquestes dues funcions serveixen perquè el mestre pugui consultar l'estat d'una determinada quantitat, que s'envia com a dada a la trama, de coils o entrades contigües a l'esclau. Es mostra a mode d'exemple la següent trama codificada en hexadecimal per poder estalviar espai i que hi càpiga, on el mestre consulta l'estat dels 11 primers coils de l'esclau amb l'adreça 20:

Adreça esclau	Codi de funció	Adreça d'inici	Quantitat de bits	CRC
14	01	00 00	00 0B	--

Taula 3. Exemple de Read Coils - Read Discrete Inputs

Com es pot observar a la taula 3, l'adreça és 14 que equival en hexadecimal al 20 decimal que és citat abans. L'adreça d'inici indica a partir de quina adreça s'iniciarà la consulta de la informació, està referenciada a zero. La quantitat de bits que es llegiran està indicada en el camp següent també en hexadecimal que correspon al valor desitjat 11 en decimal. El CRC no s'ha calculat per simplificar l'exemple, ja que l'algoritme es

pot consultar a l'annex A (necessari el càlcul de CRC).

A continuació s'indica en forma de taula la resposta de l'esclau a la consulta anterior representada a la taula 3:

Adreça esclau	Codi de funció	Byte de resposta	Estat dels bits	CRC
14	01	02	FF 07	--

Taula 4. Resposta del esclau a la sol·licitud de la Taula 3

Com s'observa a la taula 4, s'envia l'adreça de l'esclau cap al mestre perquè aquest sàpiga de qui és la resposta i el mateix codi de funció. Si hagués ocorregut algun error de codi de funció, la resposta que enviaria l'esclau seria 0x81, que es desglossa en 0x01 + 0x80, ja que 0x80 en binari és 1000 0000. Com s'ha comentat anteriorment, el MSB a "1" serveix per indicar que ha succeït un error de codi de funció.

Al camp de byte de resposta s'indica la quantitat de bytes que s'ha llegit, si el nombre de bits sol·licitats pel mestre no és múltiple de 8, sobraran bits que s'ompliran amb zeros fins al MSB. Això és més senzill d'explicar a partir de l'exemple: en el nostre cas s'han sol·licitat 11 bits, i suposarem que l'estat d'aquestes 11 entrades és "1". Els dos bytes d'estat tindrien els valors hexadecimals de 0xFF07, que en binari seria:

11111111 00000**111**

Els bits marcats en negreta són els que corresponen a l'estat de l'entrada que hem dit que suposàvem que estarien tots a "1", així quedant els zeros per acabar d'omplir el segon byte.

3.7.1.2 Read Holding Registers (03) – Read Input Registers (04)

Aquestes dues funcions són totalment anàlogues a les anteriors, però en comptes de llegir entrades d'un bit, es llegeixen registres de dos bytes. Exemple a la taula 5:

Adreça esclau	Codi de funció	Adreça d'inici	Nº de registres	CRC
14	03	00 00	00 0B	--

Taula 5. Exemple Read Holding Registers - Read Input Registers

En aquest cas l'adreça d'inici de l'esclau és la mateixa que el cas anterior de l'apartat 3.7.1.1, i pot sembla que és un error el fet que fem referència a la mateixa adreça de memòria per intentar llegir dades diferents. Això no és així, ja que el codi de funció porta implícita la informació que el dispositiu necessita conèixer per saber a quin tipus de dades ens estem referint i com es pot veure a la figura 8, per a cada tipus de dada hi ha un rang d'adreces encara que totes comencin per zero.

La resposta de l'esclau és per analogia amb la funció explicada anteriorment. Però també amb més dades, ja que es llegeixen registres de dos bytes.

3.7.1.3 Write Single Coil (5)

Aquesta funció s'utilitza per modificar l'estat d'una sortida tipus coil. S'envia l'adreça que es vol modificar, seguida de l'estat: 0x0000 per "0" i 0xFF00 per "1". Per poder aclarir l'explicació s'analitza l'exemple de la taula 6:

Adreça esclau	Codi de funció	Adreça de la bobina	Valor a escriure	CRC
14	05	00 00	FF 00	--

Taula 6. Exemple de la funció Write Single Coil

Com s'ha descrit a l'apartat anterior s'utilitza la mateixa adreça que per la bobina, la qual cosa no vol dir que ocupi la mateixa adreça de memòria que les dades llegides amb les funcions anteriorment esmentades, ja que fem referència al bloc de memòria de bobines.

La resposta de l'esclau seria una còpia de la sol·licitud sempre que s'hagi escrit sense errors per confirmar esclau, adreça i el valor escrit. En cas que es produís algun error se suma 0x80 al cdi de funció, com s'explica a l'apartat 3.7.1.1, resultant el codi de la resposta 0x85.

3.7.1.4 Write Single Register (06)

Aquesta funció succeeix com els casos anteriors que és anàloga a la funció de l'apartat 3.7.1.3 però s'escriuen 2 bytes en comptes d'escriure'n un.

Una possibilitat d'exemple seria la següent:

Adreça esclau	Codi de funció	Adreça del registre	Quantitat de bits	CRC
14	06	00 00	AA05	--

Taula 7. **Exemple de Write Single Register**

La resposta de l'esclau en cas que tot hagi sortit bé seria una còpia de la trama de la taula 7, com els casos anteriors.

4 Instrumentació i LabVIEW

Per a la creació del programa que automatitzarà la connexió dels diferents dispositius amb el protocol Modbus s'ha utilitzat el software LabVIEW de National Instruments (NI) i també per les proves un Arduino MEGA apart dels cabalímetres proporcionats per la universitat.

4.1 LabVIEW

LabVIEW (Laboratory Virtual Instrumetation Engineering Workbench) segons National Instruments (2011) és un entorn de programació gràfic de molt al nivell amb unes eines assequibles tant per usuaris experimentats com a novells. Per la seva constitució té un disseny que ajuda especialment a l'acceleració de la productivitat de qualsevol usuari. Pel fet que la seva sintaxi de programació gràfica es facilita la visualització, creació i codificació de sistemes d'enginyeria. El llenguatge que s'utilitza s'anomena llenguatge G per la seva programació gràfica i deixa enrere la feixuga programació de línies de codi.

LabVIEW és un entorn de programació la principal característica del qual és la facilitat d'ús, ja que permet fer programes relativament complexes a persones amb poca experiència en el món de la programació, com és el meu cas. També és totalment vàlid

per fer-se servir per professionals a causa del fet que compta amb una diversificació molt àmplia.

Les principals característiques que fa que el software de programació LabVIEW sigui el mes adient pel desenvolupament de la aplicació son:

- És un llenguatge que incrementa la productivitat per la part de que abstreure la complexitat del baix nivell i integrar tota la tecnologia necessària en el mateix entorn de desenvolupament.
- La possible integració a múltiples tipus de busos de comunicació
- L'avantatge de la portabilitat de la aplicació creada als diferents tipus de sistemes operatius.

4.2 Entorn LabVIEW

En els següents apartats s'exposa una introducció al entorn LabVIEW segons les fonts obtingudes d'un document explicatiu de National Instrument (2011), Lajara Vizcaíno i Pelegrí Sebastiá (2017):

4.2.1 Instruments Virtuals (VIs)

Els programes de LabVIEW són anomenats instruments virtuals o VIs ja que la seva aparença i operació imiten als instruments físics, com a oscil·loscopis i multímetres. LabVIEW conté una extensa varietat d'eines per a adquirir, analitzar, visualitzar i emmagatzemar dades, així com eines per a ajudar-lo a solucionar problemes en el codi que escrigui.

Quan crea un nou VI, veu dues finestres: la finestra del panell frontal i el diagrama de blocs.

4.2.2 Panell frontal

Quan s'obre un VI nou o existent, apareix la finestra del panell frontal del VI. La finestra del panell frontal és la interfície d'usuari per al VI.

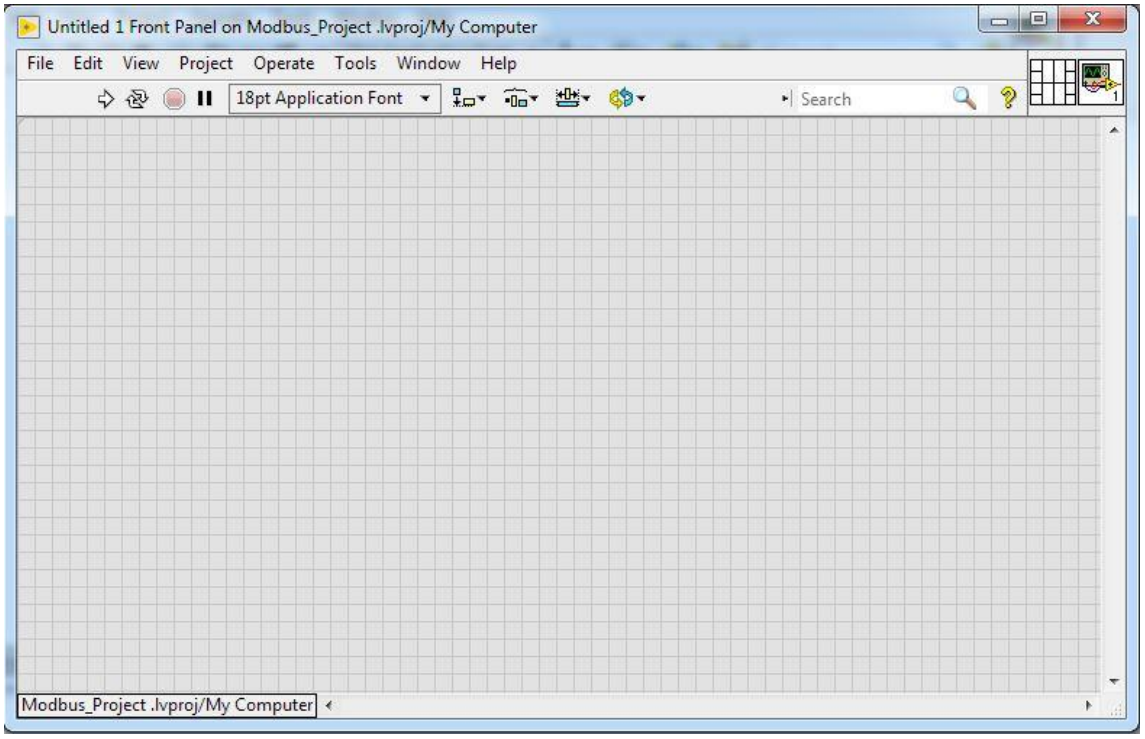


Figura 10. **Exemple de panell frontal buit**

El panell frontal es crea amb controladors i indicadors, els quals són els terminals interactius d'entrada i sortida del VI, respectivament. Els controls poden ser botons, barres lliscants i altres dispositius d'entrada. Els indicadors són gràfiques, leds i altres pantalles. Els controls simulen dispositius d'entrada d'instruments i subministren dades al diagrama de blocs del VI. Els indicadors simulen dispositius de sortida d'instruments i mostren les dades que el diagrama de blocs adquireix o genera.

Cada control o indicador té un tipus de dades associat amb ell. Els tipus de dades utilitzades amb major freqüència són numèrics, valor Booleà i cadena de caràcters.

4.2.3 Diagrama de blocs

Els objectes del diagrama de blocs inclouen terminals, subVIs, funcions, constants, estructures i cables, els quals transfereixen dades juntament amb altres objectes del diagrama de blocs.

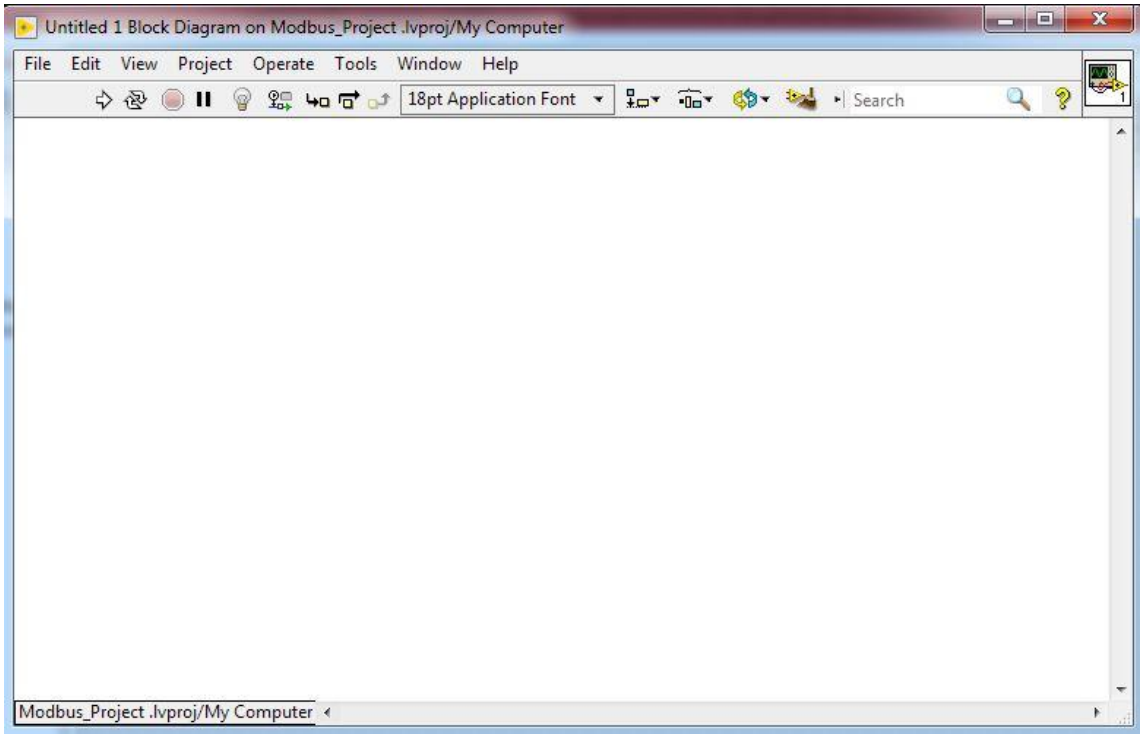


Figura 11. Exemple de diagrama de blocs buit

Després de crear la finestra del panell frontal, s'afegeix codi usant representacions gràfiques de funcions per a controlar els objectes del panell frontal. La finestra del diagrama de blocs conté aquest codi de font gràfica.

Els objectes en la finestra del panell frontal apareixen com a terminals en el diagrama de blocs. Els terminals són ports d'entrada i sortida que intercanvien informació entre el panell frontal i diagrama de blocs Els terminals són anàlegs a paràmetres i constants en llenguatges de programació basats en text.

Els tipus de terminals inclouen terminals de control o indicador i terminals de node. Les terminals de control i indicador pertanyen als controls i indicadors del panell frontal. Les dades que es proporcionen en els controls del panell frontal passen al diagrama de blocs a través de les terminals de control.

4.2.4 Flux de dades

LabVIEW segueix un model de flux de dades per a executar VIs. Un node de diagrama de blocs s'executa quan rep totes les entrades requerides. Quan el node s'executa, produeix dades de sortida i passa les dades al següent node en la trajectòria del flux de dades. El moviment de dades a través dels nodes determina l'ordre d'execució dels VIs i

les funcions en el diagrama de blocs. En el flux de control, l'ordre seqüencial dels elements del programa determina l'ordre d'execució d'un programa.

4.3 SIMEX SRL-49

"El mesurador està dissenyat per a mesurar senyals típics de corrent o voltatge. La lectura es presenta simultàniament en una pantalla LED de quatre dècades i un gràfic de barres de 20 punts i dos colors que cobreix tot el rang de mesurament. Gràcies a l'àmplia gamma de corbes característiques (lineal, arrel quadrada, quadràtica, definida per l'usuari i característica de volum per a tancs cilíndrics), els mesuradors poden usar-se en diversos sistemes de control de processos. La sortida de 24 V CC / 100 dt. està dissenyada per a subministrar transductors de mesurament, i el port RS-485 permet la transmissió de dades en sistemes de monitoratge de processos de producció. Les sortides de control REL / OC poden ajustar el nivell de senyal mesurat i es controlen d'acord amb un o dos valors de llindar. A més, el mesurador pot equipar-se amb sortides analògiques, d'acord amb la selecció del client: sortida de corrent actiu, sortida de corrent passiu aïllat o sortida de voltatge actiu. El mesurador es pot configurar o alliberar el programari a través del port de comunicació RS-485 o amb el teclat local."(SIMEX 2018)

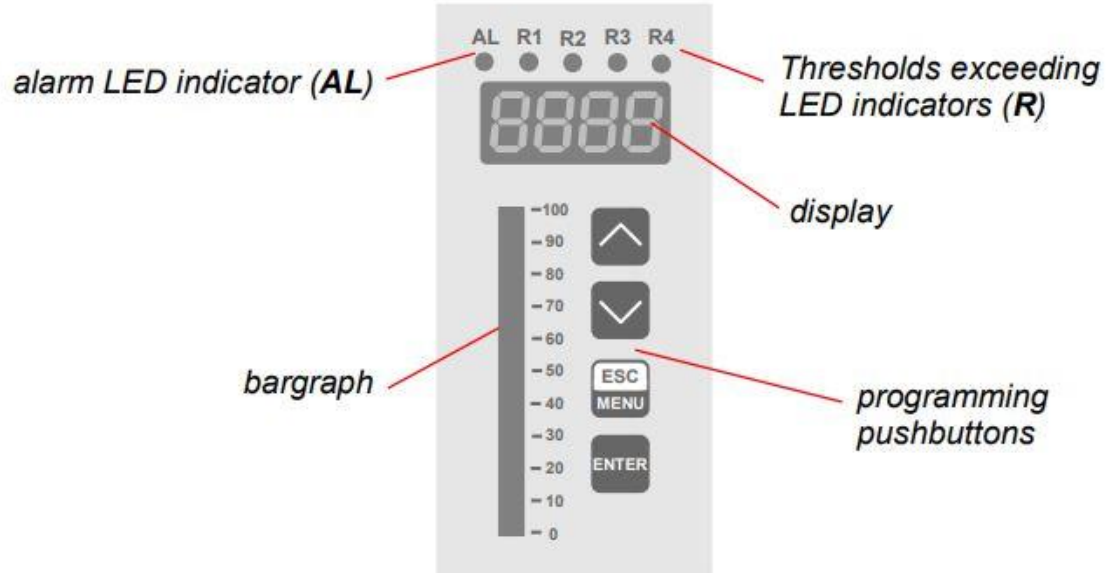


Figura 12. Pantalla frontal del SRL-49

4.4 Arduino

Amb un Arduino Mega, uns cables connectors i una protoboard de possessió personal també s'ha dut a terme la creació d'un simulador d'esclau Modbus. El codi per la creació del simulador d'un esclau ha estat extret del fòrum oficial d'Arduino (ARDUINO 2020). El codi utilitzat es pot consultar amb profunditat a l'annex B. S'ha incorporat al muntatge un convertidor TTL a RS-485 per poder dur a terme aquest simulador d'esclau i aconseguir així, la simulació de una comunicació amb el protocol Modbus sèrie.

El codi d'Arduino permet modificar els paràmetres dels bauds, nombre de bytes que genera i l'adreça de l'esclau que se li vol assignar. A més, per generar les dades d'aquests bytes s'ha connectat un potenciòmetre entre el GND i els 5V que genera Arduino. Les dades generades, per com està fet el codi seran de la posició 0 en endavant segons els bytes que hi indiquis a les variables adients del codi relacionada amb la generació de registres. En aquest cas s'ha generat 16 registres, que podran ser llegibles dels registres 0x0 al 0xF. Tots els registres són omplerts amb les dades extretes del potenciòmetre i per tant tindran el mateix valor que variarà de 0x0 a 0x400.

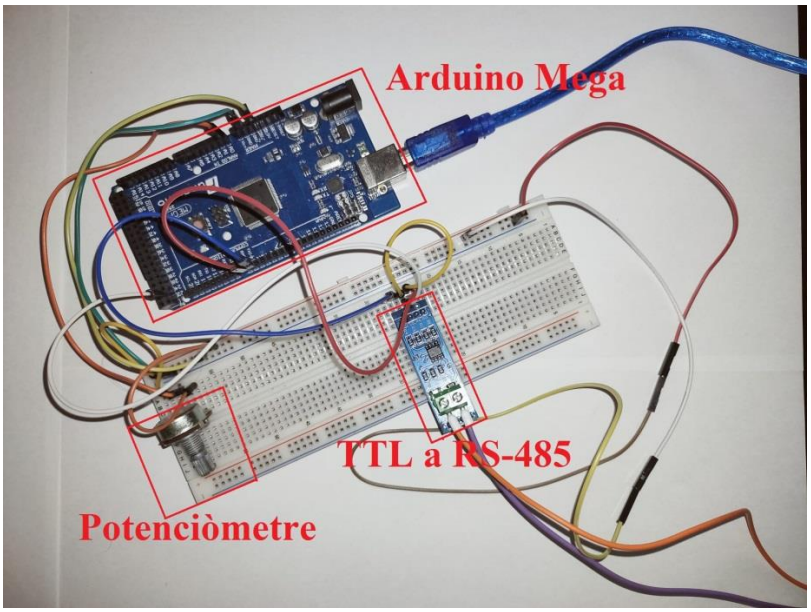


Figura 13. **Simulador esclau Modbus amb Arduino**

4.5 TUF-2000M

El mesurador de flux ultrasònic TUF-2000M està dissenyat per a mesurar la velocitat del líquid dins d'un conducte tancat. Els mesuradors de flux ultrasònics de la sèrie TUF-2000 utilitzen el principi del temps de trànsit per a mesurar la velocitat de líquids relativament nets en canonades plenes.

El mesurador de flux de temps de trànsit TUF-2000M utilitza dos transductors que funcionen com a transmissors i receptors ultrasònics. Els transductors se subjecten en l'exterior d'una canonada tancada a una distància específica entre si. El mètode de muntatge depèn de les característiques de la canonada i del líquid. El mesurador de flux funciona transmetent i rebent alternativament una ràfega d'energia del so modulada en freqüència entre els dos transductors i mesurant el temps de trànsit que triga el so a viatjar entre els dos transductors. La diferència en el temps de trànsit mesurat està directa i exactament relacionada amb la velocitat del líquid en la canonada

Figura 14. **TUF-2000M**

4.6 Convertidor RS-485 a USB

El convertidor de RS-485 a USB és una solució ràpida i segura per la interfase entre l'ordinador i els busos de comunicació industrials RS-485. En connectar el convertidor al port USB de l'ordinador és detectat i instal·lat automàticament. L'ordinador actua com si fos un port COM natiu i passa a ser compatible amb qualsevol aplicació existent de comunicació serial.

Figura 15. Convertidor RS-485 a USB

L'aïllament d'entre els ports USB i RS-485 protegeix l'ordinador de pics d'energia i errors de connexió en el bus de comunicació.

En accedir al "Gestor de Dispositius" de Windows, l'usuari pot identificar o canviar el port COM associat. La ruta és la següent: Panell de Control / Sistema / Hardware / Gestor de Dispositius / Ports COM & LPT. Després d'obrir el gestor de dispositius, és possible verificar el Port Sèrie (COM) associat al convertidor. Com es pot identificar a l'exemple de la figura 16 està associat al COM5.

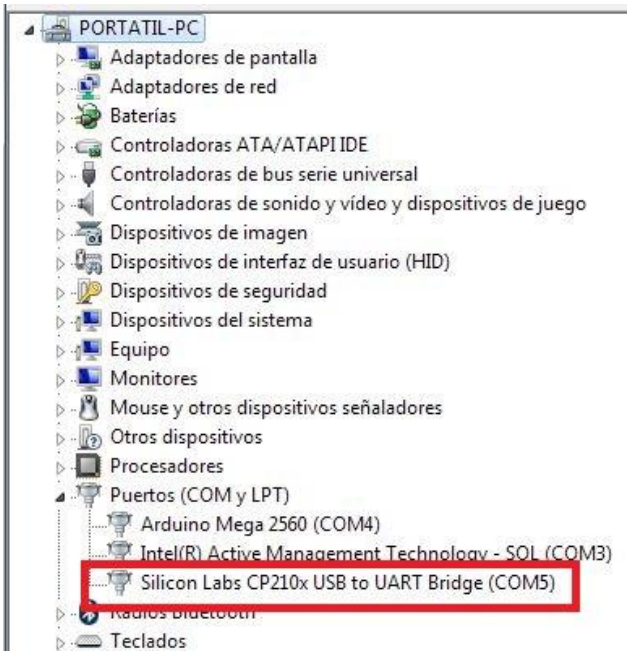


Figura 16. Gestor de dispositius

La comunicació és half-duplex, aquesta és la manera més popular d'utilització del RS-485. Només s'utilitza un parell de fils per a la transmissió i recepció de dades. Múltiples dispositius es connecten en la modalitat bus. Els diferents dispositius RS-485 utilitzen diferents notacions per a indicar la manera correcta de connectar el parell diferencial de comunicació. En el nostre cas anirà condicionada per la configuració en bus dels dispositius.

5 Aplicació Modbus

Sobre el desenvolupament fet de l'aplicació no crec que sigui de caràcter gaire interessant pel lector com s'ha produït la programació del codi. Ja que és un projecte on hi ha uns 30 arxius .vi aproximadament. És com si fos un puzle que està fet de diferents peces petites i es considera que la part més interessant a ressaltar és el funcionament de cara a l'usuari que no pas tot el codi que hi ha darrere.

5.1 Flux del programa

El primer que succeeix a l'inicialitzar l'aplicació és justament la inicialització de totes les variables que es faran servir posteriorment. Seguidament d'aquesta inicialització es passa a la selecció del port COM a utilitzar.

En el següent apartat explicarem el funcionament que succeeix després de la selecció del port com que és quan el programa entra en la màquina d'estats per fer la gestió de les variables d’entrada del programa.

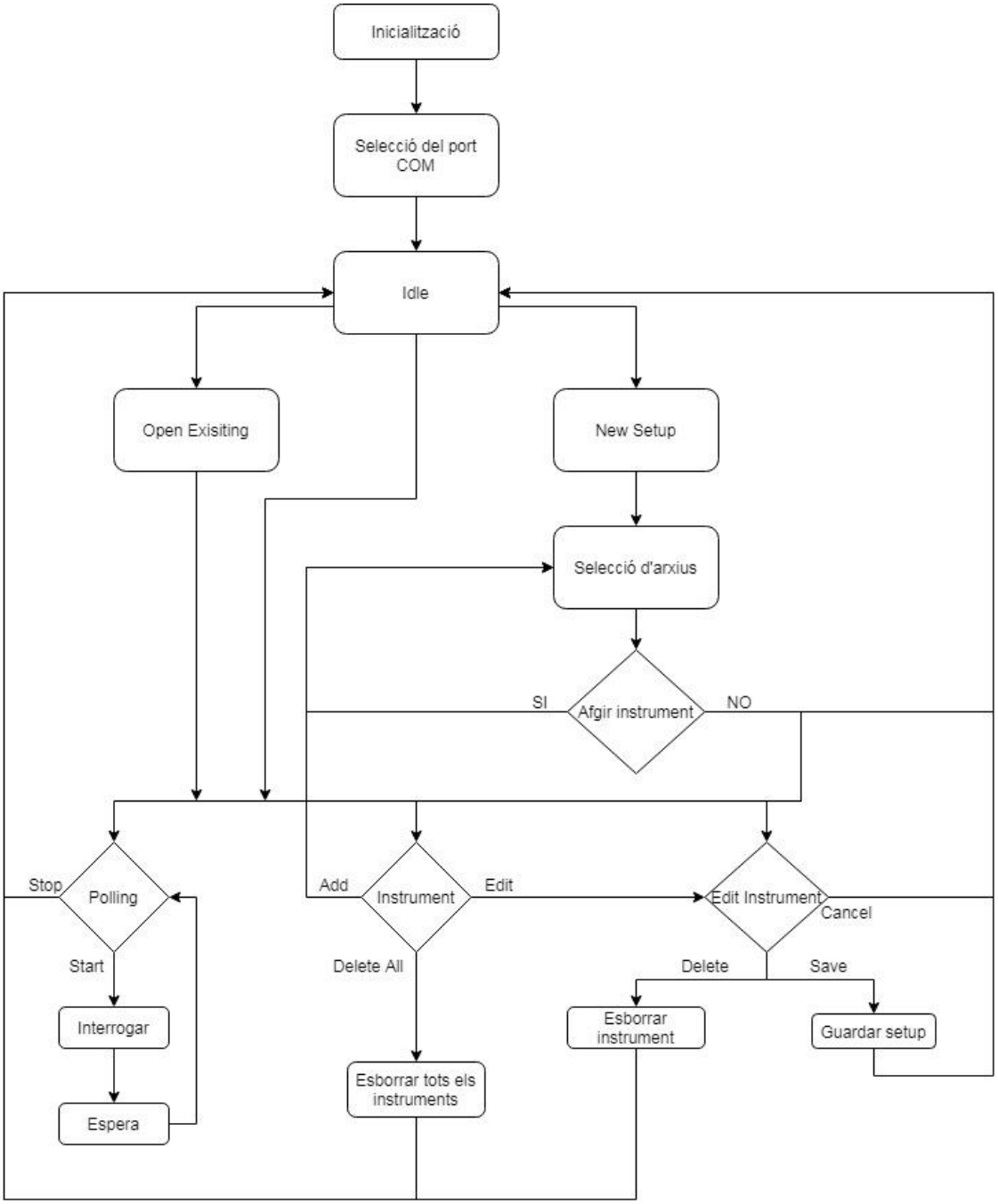


Figura 17. Diagrama de blocs de l'aplicació

Màquina d'estats:

- **Idle**: és un estat de repòs inicial on s'espera el programa a la següent acció per part de l'usuari mitjançant el menú i així poder procedir a l'execució del següent estat.
- **Open Existing**: en aquest estat de l'aplicació primerament s'obre un diàleg de fitxers per poder seleccionar el fitxer ja creat anteriorment per l'aplicació on

conté un setup ja creat prèviament.

- **New Setup**: a diferència de l'estat anterior, en aquest cas el que es realitza en obrir el diàleg de fitxers es crear un fitxer setup nou donant-li un nom escollit per l'usuari.
- **Selecció d'arxius**: al entrar el programa en aquest estat s'ha de seleccionar els arxius .ini, que explicarem més endavant, per així afegir els diferents instruments a monitoritzar per l'aplicació. A continuació s'afegiran i guardaran al setup que s'hagi seleccionat.
- **Interrogar**: en iniciar aquest estat de l'aplicació es troba l'apartat en que es fan les adquisicions de les dades seleccionades per l'usuari en l'arxiu .ini i després en la configuració de la interfície d'usuari. També en aquest estat hi ha dues opcions més, ja que a part de l'adquisició de dades es genera una gràfica amb aquestes mateixes i hi ha una part opcional de generació de un arxiu que guarda les dades adquirides amb la data i la hora de l'adquisició en un arxiu .txt.
- **Esperar**: és força evident la funció d'aquest estat, que no és més que esperar entre execució i execució de l'estat anterior d'interrogar i que ve controlat per una variable introduïda també per l'usuari anomenada time of sampling.

5.2 Funcionament de l'aplicació

Per poder comprendre el funcionament de l'aplicació hem de reprendre l'objectiu de poder connectar diferents dispositius amb un mapa de memòria diferent amb una sola aplicació i no haver de fer servir l'aplicació pròpia del fabricant per a cada dispositiu.

Per poder executar l'aplicació, el primer que s'ha de realitzar és la connexió en bus dels diferents dispositius que es desitgin monitorar i realitzar les configuracions pertinents perquè tinguin una adreça diferent cadascú, ja que si no es fes, el programa col·lidiria en preguntar al mateix dispositiu.

5.2.1 Pantalla Principal

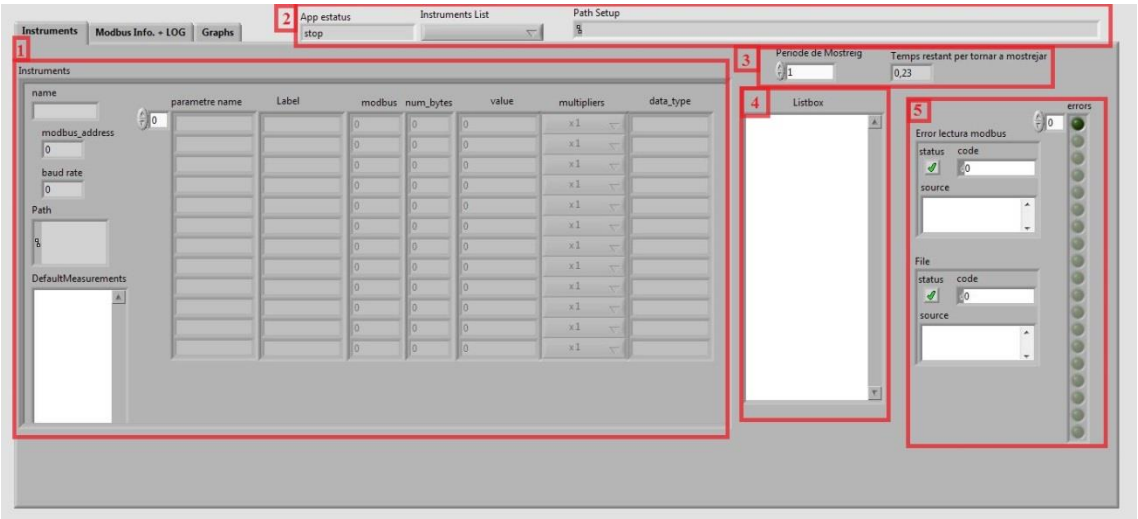


Figura 18. **Pantalla principal de l'aplicació**

1. Instruments: és l'apartat on es veuen reflectides les variables amb la seva configuració, adreça, label, etc. A banda, a l'apartat de value és on es visualitza els valors dels registres interrogats a temps reals, escollits per l'usuari. L'actualització ve donada depenent del temps que l'usuari esculli a l'apartat 3 de temps de mostreig.
2. En aquesta secció hi ha tres elements. El primer començant per l'esquerra és anomenat app status que indica l'estat de l'aplicació a temps real (màquina d'estats). Després està situat l'instrument list, on se situa la llista dels diferents instruments afegits al setup pujat o creat. Per últim se situa el path setup on es visualitza la ruta de l'arxiu setup.db que s'hagin carregat.
3. Aquí és on l'usuari pot escollir el temps de mostreig desitjat per les variables carregades a l'aplicació. També al costat hi és un comptador que indica el temps restant fins a la següent interrogació. Pot ser modificada en qualsevol moment de l'execució de l'aplicació i és un dels paràmetres que es guarda amb el setup per pròximes obertures d'aquest.
4. En aquest apartat té la funció de visualitzar els noms de les variables carregades a l'aplicació perquè puguin ser escollides en el moment del polling i ser interrogades o no. Si estan subratllades de color blau és que han estat seleccionades i seran interrogades.
5. L'últim apartat és on es poden veure els errors que poden ocórrer en l'execució

de l'aplicació. El primer és sobre la lectura dels diferents dispositius Modbus que va lligat amb els leds de la dreta, ja que els leds identifiquen en quina variable hi ha hagut l'error i per l'altra part s'observa quin tipus d'error ha ocorregut. L'altre error que es mostra és per quan hi han errades amb l'obertura de fitxers .ini

5.2.2 Arxiu de configuració

La creació d'aquest arxiu és on se solvatarà una part de la problemàtica amb els diferents mapatges de memòria. Per poder resoldre el problema abans de l'execució del programa s'hauria de realitzar la creació d'un arxiu amb extensió .ini que es configurarà de la següent manera:

El primer a fer és crear l'apartat de configuració on s'haurà d'indicar les següents variables de l'instrument desitjat:

- **Name**: nom que tindrà l'instrument a l'aplicació.
- **Baud_Rate**: bauds a que treballa el dispositiu.
- **Address_modbus**: adreça Modbus del dispositiu.
- **Adreça_decimal(0)_heax(1):** va relacionat amb els següents apartats de seccions per indicar si les direccions dels registres estan indicades amb decimal. introduint un 0 o heaxadecimal un 1.
- **Num_seccions**: nombre de seccions que s'introduiran seguidament.

A continuació de configurar aquest apartat que es dirà [config], s'ha de configurar les seccions, que serien les variables que es vulguin monitorar amb l'aplicació així relacionant cada secció a una variable del dispositiu. Per la seva correcta configuració s'ha de realitzar introduint-les de la següent manera amb la capçalera de [secció]:

- **Adreça**: registre de memòria amb que comença la variable que es vol monitorar. Tenint en compte el sistema de numeració escollit a l'apartat d'adreça_decimal(0)_hexa(1) si és el decimal o hexadecimal.
- **Description**: nom de la variable.
- **Num_bytes**: nombre de bytes que tingui la variable en qüestió.
- **Data_type**: tipus de dades emmagatzemades en els registres per al seu correcte tractament per part de l'aplicació. Els tipus que es poden definir son els

següents:

- **Integer o integerU:** es tracta d'un nombre enter de 1 byte i depenent si és amb signe o sense signe afegint una U majúscula al final si és sense signe.
- **Long o longU:** es tracta d'un nombre enter de 2 bytes i igualment que el cas anterior depenent si hi ha part negativa o no a la seva representació s'afegeix una U o no al final.
- **Real4:** registres de 2 bytes que representen un nombre de tipus flotant.

- **Writeable:** s'indica si els registres son només de lectura o també de escriptura. Per registres nomes de lectura s'indica un NO i pels que permetin també l'escriptura s'indica un SI.

```
[config]
name=Arduino
Baud_Rate=19200
address_modbus=1
adreça_decimal(0)_hexa(1)=1
Num_seccions=2

[seccio1]

adreça=0
writeable=No
description=Potenciometro
num_bytes=2
data_type=long

[seccio2]

adreça=6
writeable=No
description=Random
num_bytes=2
data_type=real4
```

Figura 19. Exemple d'arxiu .ini amb 2 variables definides

5.2.3 Creació o obertura d'un setup existent

Com es pot veure en la figura 20 en el menú de Configuration, apareixen dues opcions:

New setup, on en seleccionar-ho s'obre un diàleg d'arxiu que pots escollir el nom de l'arxiu amb extensió .db. Després de seleccionar el nom de l'arxiu desitjat, s'obrirà el diàleg d'arxius un altre cop però en aquest cas per escollir els instruments que es volen incloure en el setup creat. S'afegiran d'un en un els arxius .ini, després d'afegir cada arxiu sortirà una finestra emergent amb la pregunta de si es vol seguir afegint o no, més

arxius al setup creat.

L'altra opció que existeix en aquest menú és la d'Open existing, que la seva funcionalitat és la d'obrir un arxiu .db ja creat amb anterioritat en una execució prèvia de l'aplicació. On en obrir-ho es carregarà tots els instruments guardats amb la seva configuració a aquell setup.

Per altra banda hi ha les opcions de guardat per quan ja estigui creat el setup o obert un de creat, en qualsevol moment de l'execució del programa que es vulgui guardar el setup existeixen les opcions de save per guardar-lo en l'estat en què estigui en aquell moment amb el nom ja preestablert o save as per fer-ho amb un nom diferent del que ja té l'arxiu setup.

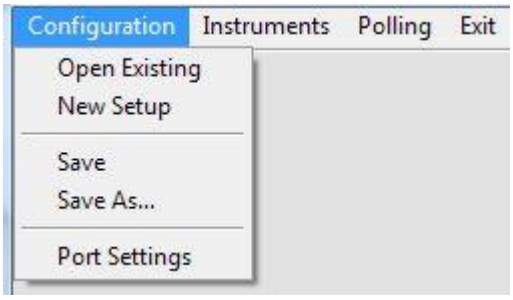


Figura 20. Apartat Configuration del menú de l'aplicació

5.2.4 Afegir, editar o esborrar dispositius

En aquest apartat del menú hi ha tres opcions que consisteixen en el següent:

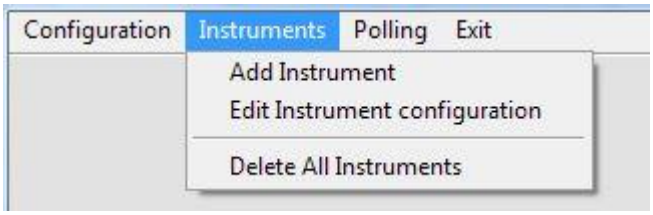


Figura 21. Pestanya Instruments del menú de l'aplicació

- **Add Instrument**: en aquesta pestanya del menú s'executa la mateixa part de després de crear un setup nou com s'ha explicat anteriorment. Hi ha la possibilitat de seguir afegint instruments a un setup ja creat, amb la seva corresponent configuració guardada en un arxiu .ini.
- **Delete all instruments:** la funció d'aquesta secció és la d'esborrar tots els

instruments carregats dintre d'un setup i deixar-lo completament buit, sense cap instrument carregat.

- **Edit instruments configuration:** en introduir-se en aquesta opció s'obrirà una altra finestra de configuració com la de la figura 22.

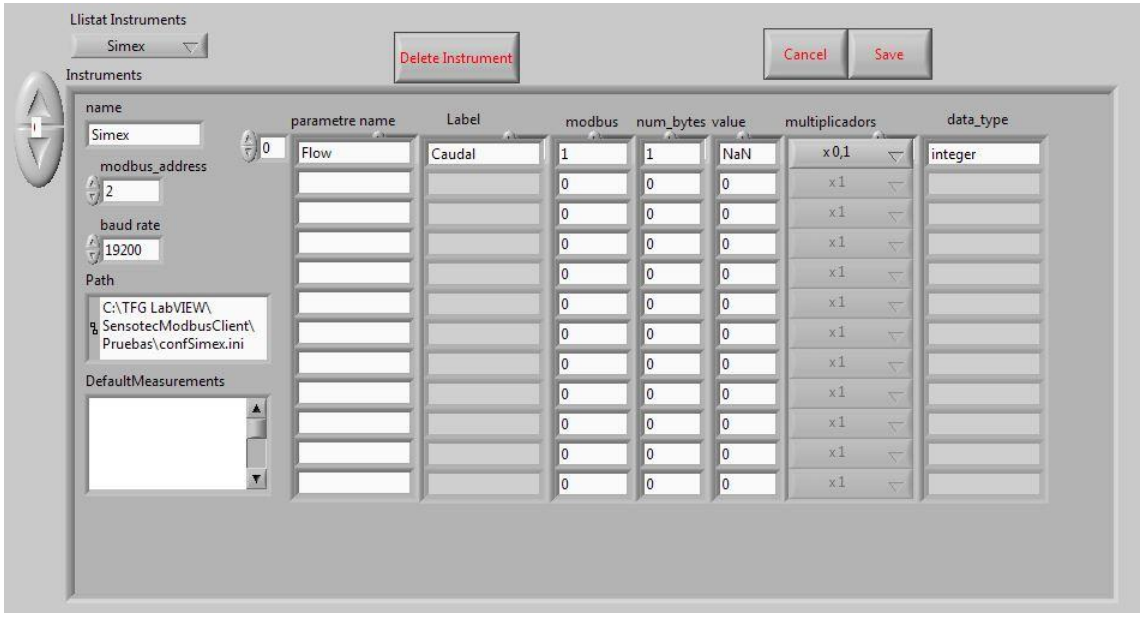


Figura 22. **Finestra d'edit instruments configuration**

En obrir-se aquesta finestra existeix la possibilitat d'editar alguns aspectes dels instruments carregats al setup, com poden ser el multiplicador en cas que calgués, alguna variable carregada a l'instrument o també introduir-hi un label per així etiquetar d'una forma més aclaridora i senzilla el nom de la variable.

Per moure's pels diferents arxius .ini carregats hi ha la pestanya de Llistat d'instruments i poder seleccionar l'instrument escollit per la modificació. En realitzar o no les modificacions adients hi ha dues opcions a escollir: cancel, per no guardar els canvis realitzats en qualsevol dels instruments modificats o save, on es guardarien els canvis realitzats. Al pressionar qualsevol de les dues opcions esmentades aquesta finestra de modificació de l'instrument es tancarà automàticament.

L'última opció que es podria utilitzar en aquesta finestra de modificació és delete instrument, que realitza la funció d'esborrar l'instrument que hagi estat seleccionat a la pestanya de llistat d'instruments. És una alternativa a l'opció que s'ha esmentat anteriorment de delete all instruments, ja que en comptes d'esborrar tots els instruments

carregats al setup, només s'esborra d'un en un per si la modificació no vol ser de tots els instruments.

5.2.5 Interrogar dispositius

Les opcions d'aquesta part del menú son ben senzilles ja que com es pot observar a la figura 23 hi ha la opció de start polling per iniciar la lectura dels registres o la opció de stop polling per finalitzar-la.

Figura 23. Menú d'interrogació

Un cop iniciada la interrogació dels dispositius carregats al setup obert, existeix la opció de seleccionar quines variables volen ser monitorades i quines no, per poder fer la selecció de les variables a monitorar de totes les que estiguin carregades per dispositiu, es podran seleccionar i des-seleccionar a l'apartat de Listbox. On apareix els noms de les variable i si està seleccionada el fons de la variable es torna blau. En aquest apartat es poden seleccionar les variables a interrogar amb un clic i, si es vol seleccionar més d'una variable a l'hora per interrogar, fer clic + Ctrl. Per des-seleccionar variables ja seleccionades també és amb un simple clic al nom del label que es visualitza a la llista Listbox i així quedarà des-seleccionada.

5.2.6 Gràfics i registre de dades

Aquest apartat està relacionat amb l'anterior, ja que s'està interrogant al dispositiu. Existeixen dues possibilitats que són la de visualitzar les variables seleccionades a través d'un gràfic en viu a l'apartat Graphs de la pantalla principal de l'aplicació amb un índex del gràfic amb checkbox per poder decidir quines són les variables a visualitzar.

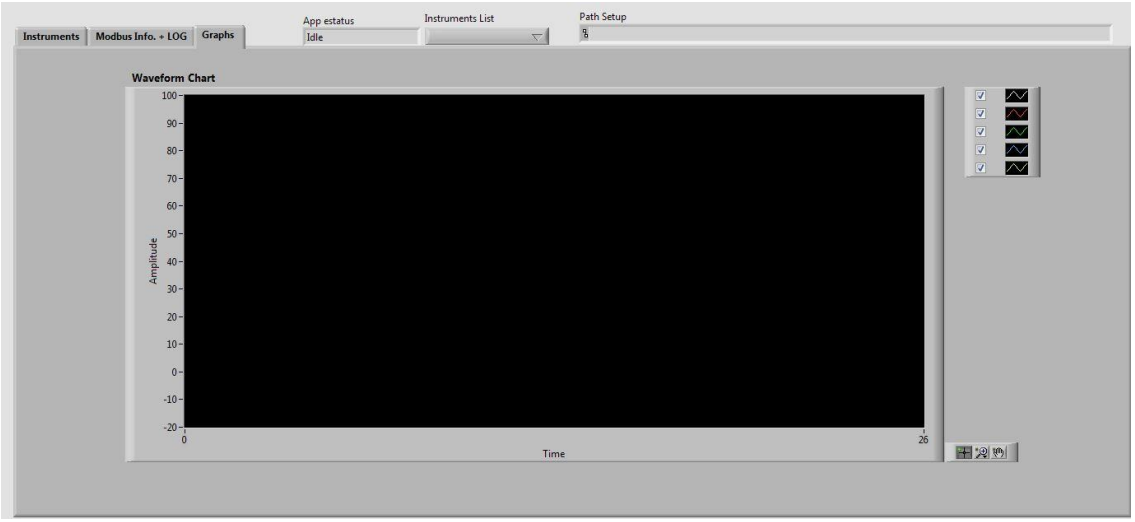


Figura 24. **Pantalla de visualització del gràfic**

L'altra opció situada a la finestra Modbus Info + LOG de la pantalla principal, permet generar un registre de les dades en un arxiu de text (.txt), que també es generaria amb únicament les dades de les variables seleccionades a la Listbox anomenada anteriorment. En aquest registre de dades, cada variable està identificada pel seu label, data i hora de la dada registrada i amb la separació d'una columna per dada registrada.

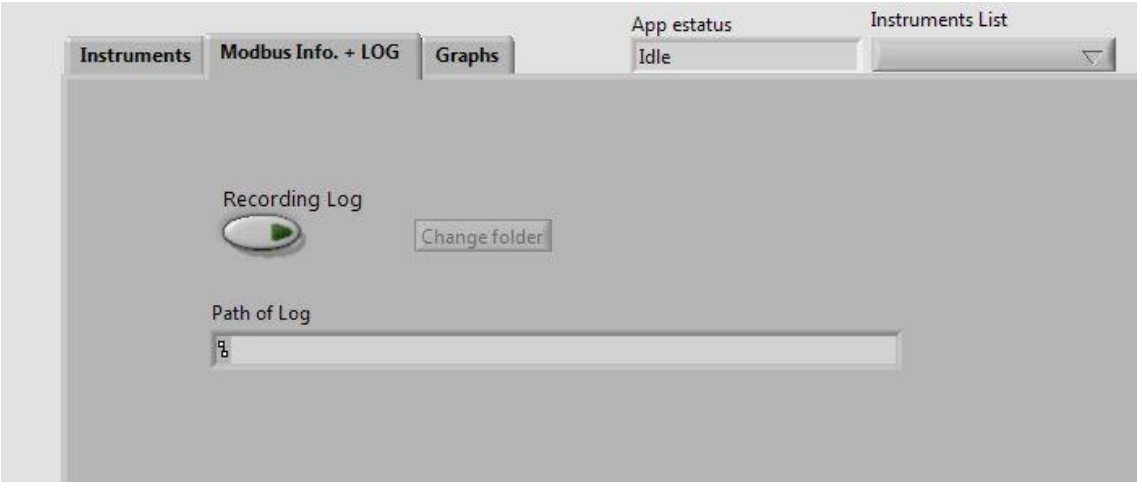


Figura 25. **Pantalla de registre de dades**

6 Comprovació de lectures

A causa de la situació sanitària amb la qual ens hem trobat en el desenvolupament del projecte, no s'han pogut realitzar totes les proves que es tenien pensades des d'un principi i s'ha hagut d'adaptar.

6.1 Fase Inicial

Primerament, per provar el correcte funcionament de l'aplicació i que les lectures fossin les esperades, es van realitzar proves amb l'Arduino amb la configuració esmentada a l'apartat 4.4 i amb el convertidor explicat a l'apartat 4.6 per poder-lo connectar al port USB de l'ordinador on s'ha desenvolupat l'aplicació i que virtualitzés el port COM per fer les proves adients. La majoria de les lectures van ser correctes i les que no van ajudar a realitzar les correccions necessàries perquè evolucionés l'aplicació.

6.2 Configuració proves en bus

En el moment en què la situació sanitària ho va permetre i es va tenir accés als altres dos dispositius comentats en els apartats 4.3 i 4.5. En obtenir aquests dispositius es va realitzar una connexió en bus dels tres dispositius per continuar amb les proves adients de l'aplicació desenvolupada on van sorgir problemes amb la gestió dels tipus de dades que anteriorment amb l'Arduino no s'havien pogut identificar, ja que només generava un tipus i, en canvi, els dos dispositius nous en generaven cinc tipus diferents de dades a gestionar. En detectar l'error de la gestió de dades va ser possible la modificació de l'aplicació pel seu correcte funcionament. Tot i fer les proves amb els dos dispositius nous, encara sorgien alguns dubtes degut a la limitació de mitjans per a mesurar cabals dels quals disposava al domicili, en especials amb el dispositiu TUF-2000M. En el cabalímetre Simex si es va aconseguir una configuració perquè les lectures les extragués de senyals de voltatge i no amb uns adaptadors per mesurar cabal. Amb el voltatge de 0-5V que genera l'Arduino i una resistència variable es va aconseguir la simulació de mesures de cabal. Les proves també es van realitzar amb altres tipus de variables com paràmetres de configuració, sortides com activació de relés, timers, etc. sobretot amb el dispositiu TUF-2000M.

Per a la realització d'aquestes proves, s'ha obtingut dels datasheets dels dos cabalímetres

la informació de les posicions de les variables a mesurar per poder dur a terme la realització del mapatge de memòria previ a l'execució de l'aplicació. Un cop executada l'aplicació amb el mateix datasheet s'ha fet la comprovació dels valors obtinguts per l'aplicació i els esperats segons les indicacions del fabricant.

S'ha provat diferents variables i configuracions. A continuació s'exposa una de les mesures fetes amb el cabalímetre Simex i el seu corresponent datasheet que es pot consultar detalladament al annex C, tot i que la configuració com s'ha esmentat anteriorment és dels tres dispositius en bus tot i que només s'analitzarà un en l'exemple, el cabalímetre SIMEX.

La visualització que extreta de l'aplicació és la següent exposada a la figura 26:

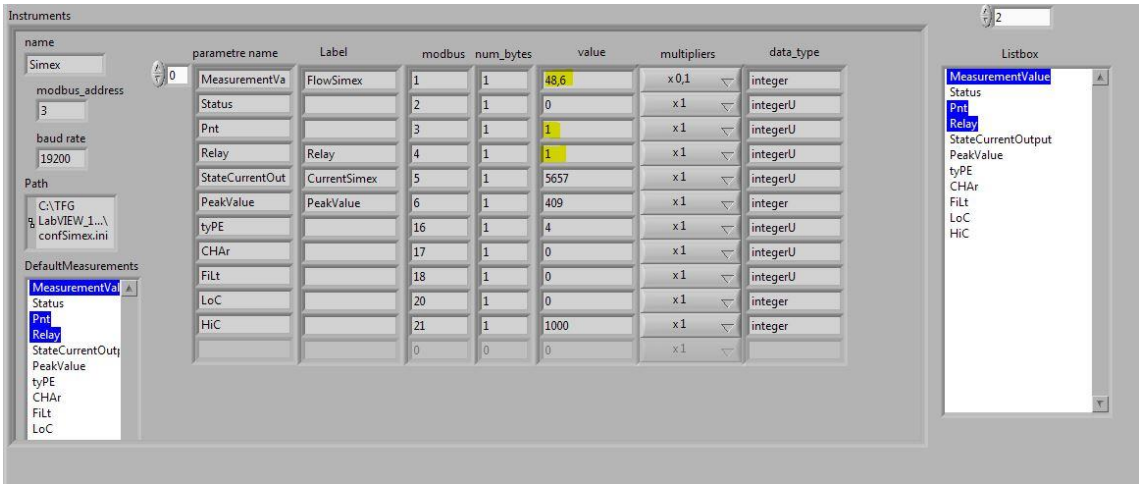


Figura 26. **Lectura del cabalímetre Simex**

Com es pot observar, hi ha tres variables del dispositiu seleccionades. L'anomenada MeasurementValue és la lectura del cabal simulat amb el voltatge de l'Arduino i la resistència variable, que mostra per pantalla al dispositiu el valor de 48.6 com comprovem a la figura 27, que coincideix amb la mesurada per l'aplicació.

Figura 27. Display del cabalímetre Simex

Aquesta variable de mesura de cabal que se situa a la posició 1 del mapatge de memòria del dispositiu ve condicionada per la variable situada al registre 3 i de nom Pnt, que és un registre de configuració. Com podem observar a la figura 26 obté un valor d'1, que vol dir que com comprovem a la figura 28 té una configuració del punt decimal al resultat de 0.0 i coincideix amb la visualització de l'aplicació.

03h	Yes	0 ÷ 3	"**Pnt** " parameter in "**InPt**" menu (decimal point position) **0** - " **0**"; **1** - " **0.0**"; **2** - " **0.00**"; **3** - "**0.000**"

Figura 28. Descripció del registre 3 del datasheet del Simex

Per altra banda com també s'observa al display del dispositiu a la figura 27 hi ha un indicador led que indica que el relé 1 està actiu. Comprovant la figura 26 de la nostra aplicació, la variable situada al registre 4 amb el nom relay té el valor d'1.

04h	Yes	see descr.	State of the relays and alarm LED (binary format) (1 - on, 0 - off): **00000000 000edcba** **a** - relay R1; **b** - relay R2; **c** - relay R3; **d** - relay R2; **e** - alarm LED; If written, only **a**, **b**, **c**, **d** bits are important (others are ignored) these bits allows user to control the relays via RS-485 interface

Figura 29. Descripció del registre 4 del datasheet del Simex

Extraient la informació del datasheet del registre 4 que podem visualitzar a la figura 29, es pot veure segons el valor en binari del registre si els relés estan actius (1) o no (0). Si passem el valor d'1 visualitzat a l'aplicació a binari és 00000001 i comparant amb la informació del registre 4 veiem que el relé "a" està actiu i és el corresponent al relé 1.

D'aquestes mesures i comprovacions podem extreure que el funcionament de l'aplicació en aquest cas és correcte tant en la visualització dels estats dels dispositius com dels registres de configuració seleccionats. Aquest procés s'ha repetit també per la resta de dispositius i la resta de variables perquè la comprovació fos més exhaustiva i poder validar la robustesa de l'aplicació.

Una altra part del projecte va ser la idea d'acudir presencialment a l'empresa Sensotec Instruments, S.A. un cop l'aplicació estigués creada i fos funcional per tal de verificar el seu correcte funcionament en les seves instal·lacions. Degut que l'actual situació sanitària no ha permès poder-hi assistir en persona, s'ha realitzat una videoconferència amb dos dels empleats d'aquesta empresa per tal de reforçar la comprovació que les mesures obtingudes dels elements esmentats eren correctes.

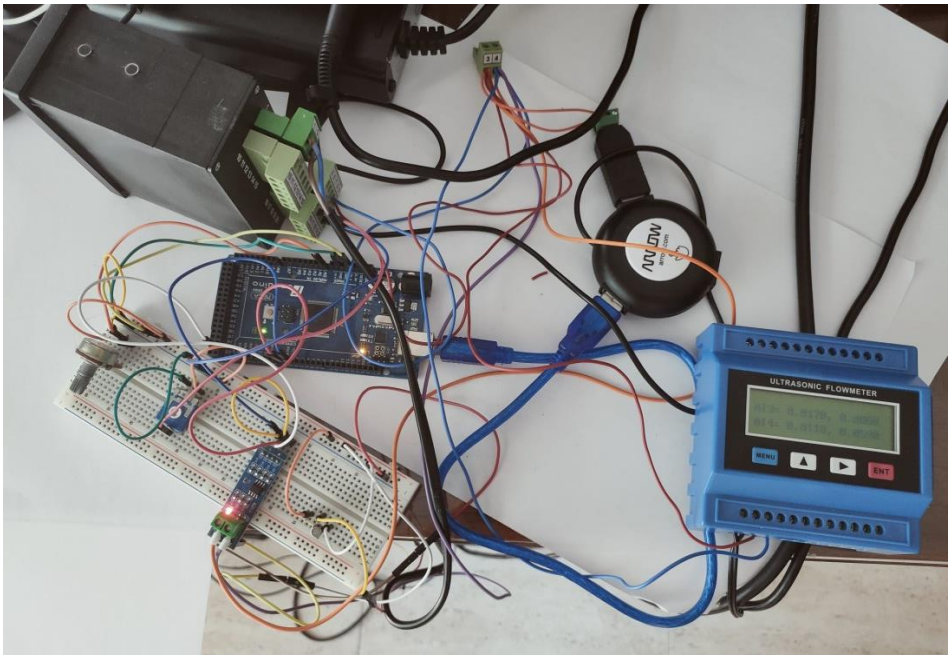


Figura 30. Setup de connexió en bus de l'Arduino i els cabalímetres

6.3 Proves a Sensotec S.A.

Un cop finalitzat l'estat d'alarma s'ha fet possible poder assistir presencialment a Sensotec Instruments, S.A. el mateix dia que es realitza l'entrega del present projecte, la qual cosa impedeix una explicació extensa de la feina realitzada.

En la visita a l'empresa de Sensotec S.A. s'han pogut realitzar proves en un banc d'assajos més similar a la realitat. Aquest banc d'assajos està compost pels següents

elements:

- SIMEX SR-49 amb els seus corresponents transmissors i receptors ultrasònics.
- TFM 2100 amb els seus corresponents transmissors i receptors ultrasònics.
- Prototip series MQU99 i MQI99.

Primerament, s'ha repassat la creació dels arxius .ini amb el personal de l'empresa per evitar alguna errada de configuració, i la creació d'un nou .ini pels prototips que encara no tenen informe tècnic oficial. Seguidament i junt amb el personal de l'empresa s'ha repassat que la connexió física fos del tot correcte i estiguessin en bus, a més de revisar que les configuracions dels bauds i les direccions fossin les correctes.

En segon lloc, s'ha procedit a realitzar una sèrie de proves de diversos valors de cabdals mesurats amb els transmissors i receptors ultrasònics. A més a l'instrument TFM 2100 se li ha connectat una sonda de temperatura PT100 per mesurar la temperatura de l'aigua d'una de les respectives canonades.

Per últim, amb l'ajuda dels coneixements dels empleats de Sensotec i els informes tècnics dels dispositius, s'ha verificat que totes les lectures fossin correctes i que tinguessin coherència amb l'informe tècnic i la visualització del display del dispositiu i l'aplicació.

Com a conclusió de les proves realitzades a Sensotec S.A. podem extreure que el funcionament de l'aplicació és el correcte i també s'ha fet un recull de propostes de millora per aquesta de parts dels professionals en la matèria que son molt valuoses per l'evolució de l'aplicació.

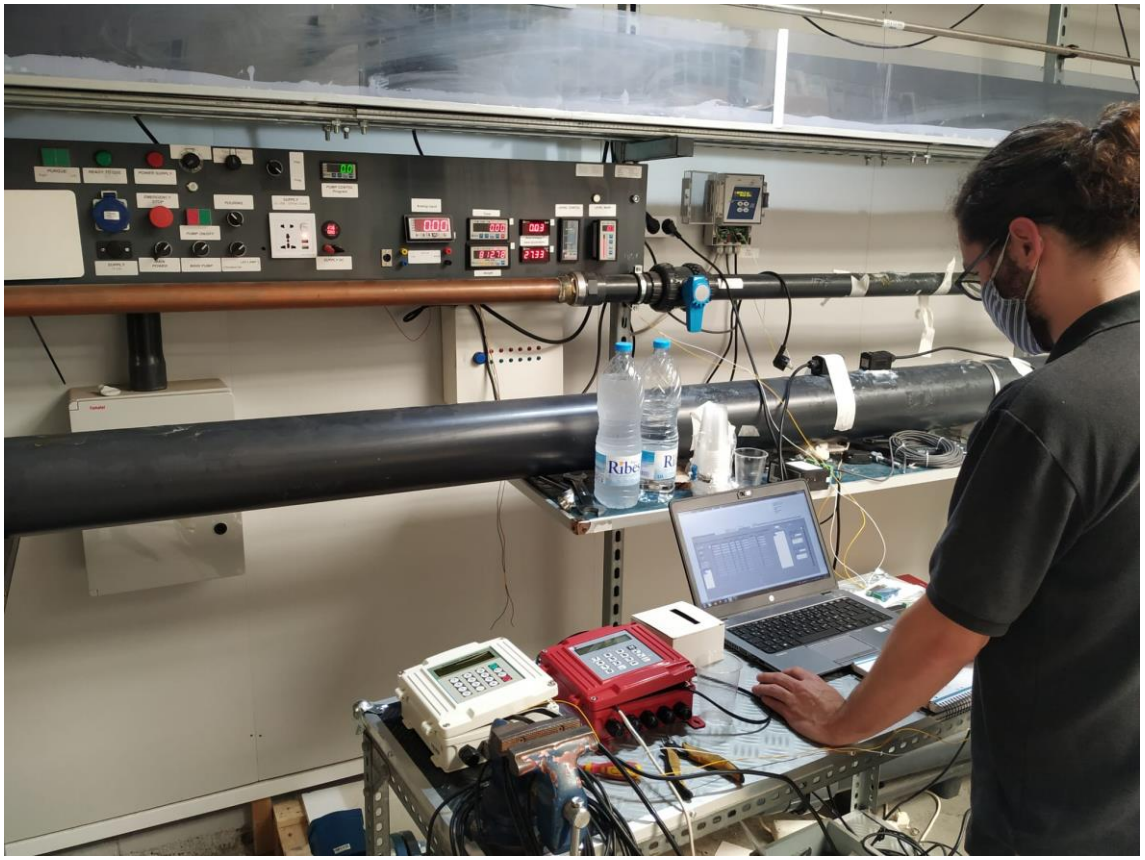


Figura 31. **Prova a Sensotec S.A**

7 Aspectes de millora

Els aspectes on podria evolucionar l'aplicació per poder oferir més prestacions i ser més complerta, com s'ha comentat per la incapacitat de l'obtenció de material la part de comunicació TCP/IP no està implementada i seria un punt interesant a afegir.

També la part de lectura és correcte i funcional totalment, però es podrien crear noves funcionalitats a l'aplicació perquè també pogués executar la part d'escriptura i controlar certs aspectes del dispositius.

Per últim, degut a la gran interacció de l'usuari amb l'aplicació hi ha punts que podrien ser estèticament més vistosos i millorar el concepte de "User-friendly", que vol dir que la seva utilització suposa senzilla per a l'usuari.

CONCLUSIONS

Durant el transcurs d'aquest projecte, es va establir com a objectiu la creació d'una aplicació per realitzar comunicacions amb el protocol Modbus de manera genèrica independentment del fabricant del dispositiu.

Amb aquest projecte s'ha aconseguit transferir les dades dels tres dispositius de fabricants diferents a la mateixa aplicació i visualitzar-los mitjançant un mapatge de memòria creat prèviament.

La condició sanitària del moment ha dificultat la creació de setups del tot adients per poder testejar els resultats obtinguts a l'aplicació de manera eficaç. Fins que va sorgir la possibilitat de testejar l'aplicació a Sensotec S.A i així comprovar el correcte funcionament en un entorn mes fiable per la validesa de l'aplicació.

Una problemàtica que no ha estat resolta per falta de material, és l'opció que les connexions no fossin només per port sèrie RS-485 sinó que també poguessin ser per Ethernet amb TCP/IP.

Com a experiència personal ha estat un projecte molt interessant, amb moltes hores de dedicació però alhora, amb molts coneixements assolits durant el seu transcurs tant teòrics com pràctics en relació al tema escollit.

AGRAÏMENTS

No m'agradaria concloure aquest projecte sense abans mostrar l'agraïment pel suport, l'ajut i la dedicació en el transcurs de tot el projecte al tutor d'aquest projecte i professor de la Universitat Politècnica de Catalunya a Vilanova i la Geltrú Joaquín del Río. També m'agradaria agrair el suport familiar i de la meva parella durant el projecte.

BIBLIOGRAFIA

ALICIA S. CLARK; EDUARDO H. CLARK. English-to-spanish Computer and Internet Dictionary. Boca Raton, Florida: Universal Publishers, 2004. ISBN 9781581124996.

ARDUINO. Modbus Arduino [en línea]. Arduino: Arduino Forum, 2020.[Consulta: 4 abril 2020]. Disponible a: < https://forum.arduino.cc/index.php?topic=240841.0 >

HARRIS MONEY, D.; HARRIS L., S. Digital design and computer architecture. 1era edició. San Francisco: Morgan Kaufmann, 2007. ISBN 9780123704979.

LAJARA VIZCAÍNO, J.R.; PELEGRÍ SEBASTIÁ, J. LabVIEW Entorno gráfico de programación. 3a Edició. Barcelona: Marcombo, 2017. ISBN 9788426724366.

MODBUS ORGANIZATION Inc. Modbus Messaging on TCP/IP Implementation Guide [en línia]. V1.0b. Hopkinton: Modbus Organization, 2006a. [Consulta: 10 març 2020]. Disponible a: < http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf>

MODBUS ORGANIZATION Inc. Modbus over serial line specification and implementation guide [en línia]. V1.02. Hopkinton: Modbus Organization, 2006b. [Consulta: 10 març 2020]. Disponible a: < http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf>

MODBUS ORGANIZATION Inc. Modbus Application Protocol Specification [en línia]. V1.1b3. Hopkinton: Modbus Organization, 2012. [Consulta: 1 març 2020]. Disponible a: < http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf>

NATIONAL INSTRUMENT. NI LabVIEW Environment: Detailed Explanation [en línea]. National Instruments, 2011. [Consulta: 18 maig 2020]. Disponible a: < http://www.ni.com/pdf/labview101/us/environment_explanation.pdf>

SCHNEIDER ELECTRIC. ¿Qué es una dirección de difusión amplia o de broadcast en una red? [en línia]. Schneider Electric, 2020. [Consulta: 17 març 2020]. Disponible a: <https://www.se.com/co/es/faqs/FA402831/ >

SIMEX. User manual METER SRL-49. Gdańsk: SIMEX, 2018. SRL-49_INSSXEN_v.2.10.001

ANNEXOS

Annex A

LRC Generation

The Longitudinal Redundancy Checking (LRC) field is one byte, containing an 8–bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The device that receives recalculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results.

The LRC is calculated by adding together successive 8–bit bytes in the message, discarding any carries, and then two's complementing the result. The LRC is an 8–bit field, therefore each new addition of a character that would result in a value higher than 255 decimal simply 'rolls over' the fields value through zero. Because there is no ninth bit, the carry is discarded automatically.

A procedure for generating an LRC is:

1. Add all bytes in the message, excluding the starting 'colon' and ending CRLF. Add them into an 8–bit field, so that carries will be discarded.

2. Subtract the final field value from FF hex (all 1's), to produce the ones–complement.

3. Add 1 to produce the twos–complement.

Placing the LRC into the Message

When the 8–bit LRC (2 ASCII characters) is transmitted in the message, the high–order character will be transmitted first, followed by the low–order character. For example, if the LRC value is 61 hexadecimal (0110 0001):

Colon	Addr	Func	Data Count	Data	Data	Data	Data	LRC Hi	LRC Lo	CR	LF
								"6" 0x36	"1" 0x31		

Example: an example of a C language function performing LRC generation is shown below.
The function takes two arguments:

unsigned char *auchMsg; A pointer to the message buffer containing binary data to be used for generating the LRC,
unsigned short usDataLen; The quantity of bytes in the message buffer.

LRC Generation Function

```
static unsigned char LRC(auchMsg, usDataLen)    /* the function returns the LRC as a type unsigned char */
unsigned char *auchMsg ;                        /*  message to calculate LRC upon */
unsigned short usDataLen ;                      /* quantity of bytes in message */
{
  unsigned char uchLRC = 0 ;                    /* LRC char initialized */
  while (usDataLen—)                            /* pass through message buffer */
  uchLRC += *auchMsg++ ;                        /* add buffer byte without carry */
  return ((unsigned char)(–((char)uchLRC))) ;   /* return twos complement */
}
```

CRC Generation

The Cyclical Redundancy Checking (CRC) field is two bytes, containing a 16–bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The device that receives recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a 16–bit register to all 1's. Then a process begins of applying successive 8–bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits and the parity bit, do not apply to the CRC.

During generation of the CRC, each 8–bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8–bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final content of the register, after all the characters of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

1. Load a 16–bit register with FFFF hex (all 1's). Call this the CRC register.

2. Exclusive OR the first 8–bit byte of the message with the low–order byte of the 16–bit CRC register, putting the result in the CRC register.

3. Shift the CRC register one bit to the right (toward the LSB), zero–filling the MSB. Extract and examine the LSB.

4. (If the LSB was 0): Repeat Step 3 (another shift).

(If the LSB was 1): Exclusive OR the CRC register with the polynomial value 0xA001 (1010 0000 0000 0001).

5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8–bit byte will have been processed.

6. Repeat Steps 2 through 5 for the next 8–bit byte of the message. Continue doing this until all bytes have been processed.

7. The final content of the CRC register is the CRC value.

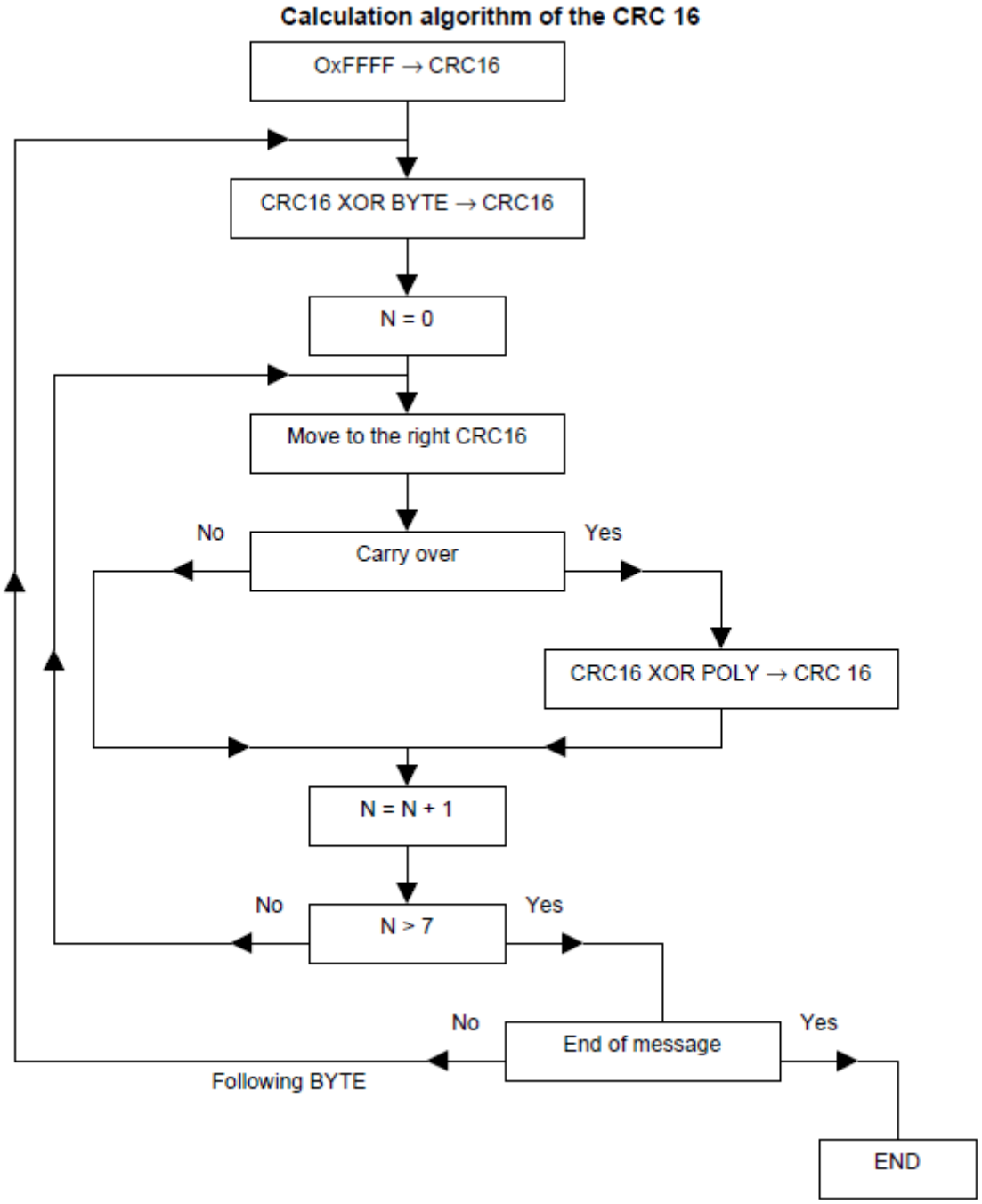
8. When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

Placing the CRC into the Message

When the 16–bit CRC (two 8–bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high- order byte.

For example, if the CRC value is 1241 hex (0001 0010 0100 0001):

Addr	Func	Data Count	Data	Data	Data	Data	CRC Lo	CRC Hi
							0x41	0x12



XOR = exclusive or

N = number of information

POLY = calculation polynomial of the CRC 16 = 1010 0000 0000 0001

(Generating polynomial = 1 + x2 + x 15 + x 16)

In the CRC 16, the 1st byte transmitted is the least significant one.

Example of CRC calculation (frame 02 07)

CRC register initialization		1111	1111	1111	1111
XOR 1st character		0000	0000	0000	0010
		1111	1111	1111	1101
	Move 1	0111	1111	1111	1110|1
Flag to 1, XOR polynomial		1010	0000	0000	0001
		1101	1111	1111	1111
	Move 2	0110	1111	1111	1111|1
Flag to 1, XOR polynomial		1010	0000	0000	0001
		1100	1111	1111	1110
	Move 3	0110	0111	1111	1111|0
	Move 4	0011	0011	1111	1111|1
		1010	0000	0000	0001
		1001	0011	1111	1110
	Move 5	0100	1001	1111	1111|0
	Move 6	0010	0100	1111	1111|1
		1010	0000	0000	0001
		1000	0100	1111	1110
	Move 7	0100	0010	0111	1111|0
	Move 8	0010	0001	0011	1111|1
		1010	0000	0000	0001
		1000	0001	0011	1110
XOR 2nd character		0000	0000	0000	0111
		1000	0001	0011	1001
	Move 1	0100	0000	1001	1100|1
		1010	0000	0000	0001
		1110	0000	1001	1101
	Move 2	0111	0000	0100	1110|1
		1010	0000	0000	0001
		1101	0000	0100	1111
	Move 3	0110	1000	0010	0111|1
		1010	0000	0000	0001

	1100	1000	0010	0110
Move 4	0110	0100	0001	0011 | 0
Move 5	0011	0010	0000	1001 | 1
	1010	0000	0000	0001
	1001	0010	0000	1000
Move 6	0100	1001	0000	0100 | 0
Move 7	0010	0100	1000	0010 | 0
Move 8	0001	0010	0100	0001 | 0
		Most significant	least significant	

The CRC 16 of the frame is then: 4112

Exemple:

An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16–bit CRC field, and the other array contains all of the values for the low byte.

Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

Note: This function performs the swapping of the high/low CRC bytes internally. The bytes are already swapped in the CRC value that is returned from the function.

Therefore the CRC value returned from the function can be directly placed into the message for transmission. The function takes two arguments:

unsigned char *puchMsg; A pointer to the message buffer containing binary data to be used for generating the CRC

unsigned short usDataLen; The quantity of bytes in the message buffer.

CRC Generation Function:

```
unsigned short CRC16 ( puchMsg, usDataLen )        /* The function returns the CRC as a unsigned short type */
unsigned char *puchMsg ;                            /* message to calculate CRC upon                         */
unsigned short usDataLen ;                          /* quantity of bytes in message                          */
{
        unsigned char uchCRCHi = 0xFF ;             /* high byte of CRC initialized                          */
        unsigned char uchCRCLo = 0xFF ;             /* low byte of CRC initialized                           */
        unsigned uIndex ;                           /* will index into CRC lookup table                      */

        while (usDataLen--)                         /* pass through message buffer                           */
        {
                uIndex = uchCRCLo ^ *puchMsg++ ;    /* calculate the CRC                                     */
                uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
                uchCRCHi = auchCRCLo[uIndex] ;
        }
        return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Annex B

```
#include <ModbusRtu.h>

// Assign the Arduino pin that must be connected to RE-DE RS485 transceiver
#define TXEN  22
/**
 *  Modbus object declaration
 *  u8id : node id = 0 for master, = 1..247 for slave
 *  u8serno : serial port (use 0 for Serial)
 *  u8txenpin : 0 for RS-232 and USB-FTDI
 *              or any pin number > 1 for RS-485
 */
Modbus slave(1,0,TXEN); // this is slave @1 and RS-485
unsigned int leer= 16;
int miarray[16];
int var=0;

const int numReadings = 16;

int readings[numReadings];
int readIndex = 0;
int total = 0;


int inputPin = A0;              //Potenciometro
void setup() {
  slave.begin( 19200 );         // baud-rate
  for (int thisReading = 0; thisReading < numReadings; thisReading++)
    readings[thisReading] = 0;
}
```

```
void loop() {
//Generació de les dades

 total = total - readings[readIndex];
  readings[readIndex] = analogRead(inputPin);
  total = total + readings[readIndex];
  readIndex = readIndex + 1;
  if (readIndex >= numReadings) {
      slave.poll( readings,16);
      readIndex=0;
  }

  }
```

Annex C

11.1. LIST OF REGISTERS

Register	Write	Range	Register description
01h	No	-999 ÷ 9999	Measurement value (no decimal point)
02h	No	0h, A0h, 60h	The status of the current measurement; **0h** - data valid; **A0h** - top border of the measurement range is exceeded; **60h** - bottom border of the measurement range is exceeded;
03h	Yes	0 ÷ 3	"**Pnt** " parameter in "**InPt**" menu (decimal point position) **0** - " **0**"; **1** - " **0.0**"; **2** - " **0.00**"; **3** - "**0.000**"
04h	Yes	see descr.	State of the relays and alarm LED (binary format) (1 - on, 0 - off): **00000000 000edcba** **a** - relay R1; **b** - relay R2; **c** - relay R3; **d** - relay R2; **e** - alarm LED; If written, only **a**, **b**, **c**, **d** bits are important (others are ignored) these bits allows user to control the relays via RS-485 interface
05h[1]	Yes	0h ÷ 1800h	State of active current output, expressed in 1/256 mA units – it means that high byte express integer part, and low byte fractional part of desired output current.
	Yes	2CCh÷1800h	State of passive current output, expressed in 1/256 mA units – it means that high byte express integer part, and low byte fractional part of desired output current.
	Yes	0h ÷ 1600h	State of active voltage output, expressed in 1/512 V units – it means that high byte express integer part, and low byte fractional part of desired output voltage.
06h	No	-999 ÷ 9999	Peak (drop) value (no decimal point)
10h	Yes	0 ÷ 5	"**tyPE**" parameter in "**InPt**" menu (nominal input range). **0** - 0-20 mA range; **1** - 4-20 mA range; **2** - 0-10 V range; **3** - 2-10 V range; **4** - 0-5 V range; **5** - 1-5 V range
11h	Yes	0 ÷ 5	"**CHAr**" parameter in "**InPt**" menu (characteristic type) **0** - linear ; **1** - square; **2** - square root; **3** - user defined; **4** - volume characteristics of a cylindrical tank in the vertical position; **5** - volume characteristics of a cylindrical tank in the horizontal position
12h	Yes	0 ÷ 5	"**FiLt**" parameter in "**InPt**" menu (measurement filtering rate)