



UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA:

INGENIERÍA ELECTRÓNICA

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO
DE:**

INGENIERO ELECTRÓNICO

**DISEÑO E IMPLEMENTACIÓN DE MÓDULOS DE RED MODBUS/TCP ENTRE
TRES AUTOMATAS PROGRAMABLES PARA ARRANQUE DE MOTOR
TRIFÁSICO DE MANERA LOCAL, REMOTO Y LECTURA DE SENSORES.**

AUTOR:

NICOLÁS EFRAÍN CRESPO DELGADO

TUTOR:

MÓNICA MARÍA MIRANDA RAMOS

GUAYAQUIL- ECUADOR

2022

**CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE
TITULACIÓN**

Yo, Nicolás Efraín Crespo Delgado con documento de identificación N° 0931149264 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 11 de mayo de 2022

Atentamente,



.....
Nicolas Efraín Crespo Delgado

CI: 0931149264

**CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO
DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA**

Yo, Nicolás Efraín Crespo Delgado con documento de identificación N° 0931149264, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del: Proyecto Técnico: “Diseño e implementación de módulos de red MODBUS/TCP entre tres autómatas programables para arranque de motor trifásico de manera local, remoto y lectura de sensores”, el cual ha sido desarrollado para optar por el título de: Ingeniero electrónico mención en Automatización Industrial, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 11 de mayo de 2022

Atentamente,



.....
Nicolas Efraín Crespo Delgado

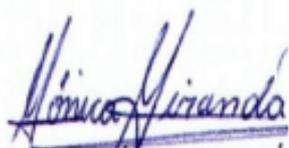
CI: 0931149264

CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN

Yo, MSc Mónica María Miranda Ramos, con documento de identificación N° 0908222987, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE MÓDULOS DE RED MODBUS/TCP ENTRE TRES AUTOMATAS PROGRAMABLES PARA ARRANQUE DE MOTOR TRIFÁSICO DE MANERA LOCAL, REMOTO Y LECTURA DE SENSORES, realizado por Nicolás Efraín Crespo Delgado con documento de identificación N° 0931149264, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 11 de mayo de 2022

Atentamente,



.....
Ing. Mónica María Miranda Ramos.

CI: 0917217785

DEDICATORIA

Este trabajo va dedicado para mi familia, padres, amigos quienes con su esfuerzo y colaboración han estado presentes en este proceso de preparación que he tomado para enfrentar con éxitos mi vida profesional, brindándome consejos y recursos necesarios para poder enfrentar este desafío de culminar mi proyecto de titulación.

Agradecer a mis padres y en especial a mi abuelita quien en vida deseo que me pueda graduar y con su ayuda, sacrificio me brindo lo necesario para encarar mis estudios hasta el final de mi carrera, aconsejándome para servir a mi país y a la sociedad.

A handwritten signature in blue ink, consisting of a large, stylized letter 'N' followed by a smaller 'E' and 'D'.

Nicolás Efraín Crespo Delgado.

AGRADECIMIENTOS

Agradezco a mi familia que con su aliento constante me ayudaron a superar este gran desafío, a todas las personas que conocí en este duro camino, pero no imposible que me ayudaron a forjar el carácter y no darme por vencido.

A mis ganas de superarme con dedicación y esfuerzo.

Agradezco a mis compañeros de mi primer trabajo, gracias a ellos nació la idea de este proyecto.

A handwritten signature in blue ink, consisting of a large, stylized letter 'N' followed by a smaller 'E' and 'C'.

Nicolás Efraín Crespo Delgado.

ÍNDICE GENERAL

TITULACIÓN.....	II
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA	III
CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN.....	IV
INTRODUCCIÓN	1
1.1 Planteamiento del problema	2
1.2 Delimitación del problema	3
1.3 Objetivos	3
1.3.1 Objetivo general	3
1.3.2 Objetivos específicos	3
1.4 Justificación del proyecto.....	4
1.5 Hipótesis.....	4
1.6 Variable e Indicadores.....	5
1.6.1 Variables	5
1.6.2 Indicadores	5
1.7 Metodología	5
1.7.1 Método Inductivo	5
1.7.2 Método Teórico y Sistemático.....	5
1.7.3 Técnica Documental.....	6
1.8 Población y Muestra	6
1.8.1 Población.....	6
1.8.2 Muestra	6
1.9 Descripción de la propuesta	7
1.10 Beneficiarios	7
1.11 Impacto	7
CAPÍTULO 2: MARCO TEÓRICO.....	8
2.1 PLC Unitronics Visión 700.....	8
2.1.1 Especificaciones técnicas modulo V700-T20BJ	9
2.1.2 Software de programación de PLC Unitronics	10
2.2 Variador de frecuencia	11
2.2.1 Concepto básico de variadores de velocidad para motores.....	12
2.3 Fuente de Alimentación.....	13

2.4	Disyuntor	13
2.5	Luces indicadoras.....	14
2.6	Pulsadores y selectores	14
2.8	Motor Trifásico	15
2.8.1	Partes de un motor trifásico.....	16
2.8.2	Funcionamiento de un motor trifásico.....	17
2.8.3	Ventajas un motor trifásico	17
2.9	Arduino	17
2.9.1	Características de Arduino	18
2.9.1.2	Ventajas y Desventajas de Arduino.....	19
2.9.2	Arduino Mega.....	19
2.9.3	Módulo Arduino Ethernet Shield	21
2.10	Potenciómetros	22
2.11	Display LCD con comunicación I2C	22
2.12	Sensores.....	23
2.13	PLC Siemens S7-1200	28
2.14	Antena Ubiquiti Nano Station Loco M5	28
2.15	MODBUS	29
2.15.1	Comunicaciones Maestro/Esclavo con Modbus	30
2.15.2	Tipos de MODBUS.....	30
2.15.3	MODBUS ASCII.....	31
2.15.4	MODBUS RTU.....	32
2.15.5	MODBUS TCP/IP	32
2.15.6	Arquitectura general del protocolo MODBUS.....	33
2.15.7	MBAP Descripción de cabecera.....	35
2.15.7	Descripción funcional de MODBUS TCP/IP	36
2.15.8	Capa de gestión TCP.....	38
2.15.9	Gestión de conexiones TCP	40
2.15.9	Parametrización IP	41
2.15.10	Modbus Utilizando NI OPC Servers	41
2.15.11	Modbus Usando APIs de Bajo Nivel.....	42
2.15.12	Ventajas y desventajas de Modbus.....	43
3.	CAPITULO 3: MARCO METODOLÓGICO	44
3.1	Descripción de módulos didácticos.....	44
3.2	Armarios Termoplásticos.....	46
3.3	Medidas y perforación en armarios industriales	46

3.4 Montaje de equipos en puertas de los armarios	47
3.5 Montaje de equipos en placa metálica.....	48
3.6 Montaje y cableado de placas metálicas en el armario.....	50
3.7 Armarios terminados para realizar prueba	51
3.8 Pruebas de conexión de los armarios	51
3.9 Configuración de Antenas Ubiquiti.....	52
3.9.1 Pasos de conexión antena:	52
3.9.2 Configuración de antena de Acceso.....	54
3.9.3 Configuración de antenas de Estaciones.....	56
3.9 Configuración y carga de librería MODBUS TCP/IP en software Arduino	59
3.10 Configuración del software VisiLogic	60
3.9 Configuración de PLC Siemens.....	62
3.10 Conexión de sensores y banda transportadora	66
4. Guía de prácticas de laboratorio	67
4.1 Práctica 1	67
4.2 Práctica 2	74
4.3 Práctica 3	85
4.4 Práctica 4	96
4.5 Práctica 5	107
4.6 Práctica 6	120
4.7 Práctica 7	134
4.8 Práctica 8	149
4.9 Práctica 9	164
4.10 Práctica 10	184
5. Análisis y resultados	200
5.1 Resultados obtenidos	200
5.2 Análisis de resultados.	205
CONCLUSIONES.....	207
RECOMENDACIONES.....	208
BIBLIOGRAFÍA.....	209

ÍNDICE DE FIGURAS

Figura 2.1: Autómata programable Unitronics modelo V700	8
Figura 2.2: Modulo I/O modelo V200-18-E2B	10
Figura 2.3: Software de programación VisiLogic.	10
Figura 2.4: Esquema E/S de un variador de frecuencia	11
Figura 2.5: Esquema interno de un variador de frecuencia	12
Figura 2.6: Fuente de alimentación MeanWell 24VDC	13
Figura 2.7: Disyuntor magneto térmico monofásico bipolar	14
Figura 2.8: Luces piloto industriales	14
Figura 2.9: Selectores y pulsadores industriales.....	15
Figura 2.10: Motor trifásico.....	16
Figura 2.11: Entradas y salidas de Arduino Mega.....	20
Figura 2.12: Arduino Mega 2560	21
Figura 2.13: Shield Ethernet	22
Figura 2.14: Potenciómetro	22
Figura 2.15: Diagrama de conexión LCD 16x2 por I2C	23
Figura 2.16: Kit de módulo de sensores y actuadores.	23
Figura 2.17: Dibujo técnico sensor ultrasónico.	24
Figura 2.18: Diagrama de sensor de Luz o Fotorresistencia.....	25
Figura 2.19: Fotorresistencia.	25
Figura 2.20: Sensor Infrarrojo.	26
Figura 2.21: Sensor de flujo.	27
Figura 2.22: PLC Siemens S7-1200	28
Figura 2.23: Antena Ubiquiti Loco Nano Station M5	29
Figura 2.24: Una relación de red Maestro-Esclavo	30
Figura 2.25: MODBUS TCP/IP commutation architecture.....	33
Figura 2.26: Marco general MODBUS	34
Figura 2.27: MODBUS Request/response over TCP/IP.....	34
Figura 2.28: MODBUS messaging Service Conceptual Architecture.	37
Figura 2. 29: MODBUS con NI OPC SERVERS.....	42
Figura 2.30: Modbus Usando APIs de Bajo Nivel.....	43
Figura 31: Esquema de proyecto	44
Figura 32: Armario Industrial Famatel	46
Figura 33: Perforación de puerta para elementos	47
Figura 34: Cableado de elementos en puerta de armarios.....	47
Figura 35: Cableado de tablero metálico	48
Figura 36: Resistencia de Pull Up/Down.....	49
Figura 37: Cableado de tablero metálico Unitronics	49
Figura 38: Cableado y unión de puertas con tableros metálicos.....	50
Figura 39: Instalación de antena y puertos de red para salidas en módulo Arduino.....	51
Figura 40: Armarios terminados por dentro	51
Figura 41: Funcionamiento de módulos	52
Figura 42: Antena Ubiquiti Loco M5	52
Figura 43: Alimentación de voltaje Antenas	53
Figura 44: Configuración IP en PC	53
Figura 45: Ingreso por Web a antenas.	53
Figura 46: Login a web de Antenas	54
Figura 47: Configuración Wireless de Antenas	54

Figura 48: Desactivación de airMax.....	55
Figura 49: Configuración de pestaña Network.....	55
Figura 50: Conexión entre antena con router con acceso a internet.....	56
Figura 51: Configuración de IP estática.....	56
Figura 52: Configuración de modo de antena.....	57
Figura 53: Verificación de estado de conexión.....	57
Figura 54: Panel de LEDS.....	58
Figura 55: Panel de leds con red activa.....	58
Figura 56: Escaneo de antenas conectadas a la red de acceso.....	59
Figura 57: Configuración y carga de librería MODBUS TCP/IP en software Arduino.....	60
Figura 58: Configuración del software VisiLogic.....	60
Figura 59: Selección del módulo de entradas y salidas.....	61
Figura 60: Descripción de entradas y salidas.....	61
Figura 61: Configuración de PLC Siemens.....	62
Figura 62: instrucción "MB_SERVER".....	63
Figura 63 Conexión de sensores y banda transportadora.....	66
Figura 64: Tablero de control Arduino.....	200
Figura 65: Planta didáctica, llenado de tanques y banda transportadora.....	201
Figura 66: Tablero de control PLC Unitronics.....	202
Figura 67: Motor trifásico en funcionamiento.....	202
Figura 68 Prueba de equipos conectados a la red.....	204
Figura 69 Prueba de conexión de Antenas Ubiquiti.....	204

ÍNDICE DE TABLAS

Tabla 1: Especificaciones PLC.....	9
Tabla 2: Especificaciones Arduino Mega 2560.....	19
Tabla 3: Especificaciones Ubiquiti Nanostation.....	29
Tabla 4: Formato de byte en modo ASCII.....	31
Tabla 5: Formato de byte en modo RTU.....	32
Tabla 6: Cabecera MBAP.....	35
Tabla 7: Mensajería MODBUS en la guía de implementación de TCP/IP V1.0b de MODBUS.....	37
Tabla 8. Elementos en las estaciones y direcciones IP.....	45
Tabla 9: Parámetros de instrucción "MB_SERVER".....	63

RESUMEN

AÑO	ALUMNO	DIRECTOR DE PROYECTO	TEMA DE TITULACIÓN
2022	NICOLÁS EFRAÍN CRESPO DELGADO	ING. MÓNICA MIRANDA RAMOS	DISEÑO E IMPLEMENTACIÓN DE MÓDULOS DE RED MODBUS/TCP ENTRE TRES AUTOMATAS PROGRAMABLES PARA ARRANQUE DE MOTOR TRIFÁSICO DE MANERA LOCAL, REMOTO Y LECTURA DE SENSORES.

El proyecto técnico planteado tiene como objetivo principal enseñar la técnica de comunicación entre tres autómatas programables diferentes, en este caso con el protocolo MODBUS/TCP-IP, además de variación de frecuencia de un motor trifásico. Se incluye también la lectura de sensores desde el módulo de Arduino.

Daré los detalles del uso de protocolo, el diseño de circuito para variación de velocidad de motores y las ventajas y desventajas de este proyecto en general.

Esto se logrará a través de tres módulos didácticos equipados con controladores lógicos Programables, PLC Unitronics Visión 700, una placa ARDUINO MEGA acoplado con un ARDUINO ETHERNET, PLC SIEMENS S7-1200, un MOTOR TRIFÁSICO, antenas UBIQUITI NANOSTATION LOCO M5 como medio de comunicación entre las estaciones, Computador personal.

Los módulos servirán para el personal docente y estudiantes ya que el proyecto estará dividido en varios niveles de dificultad para el desarrollo de las prácticas propuestas, lo cual servirá como nuevo material académico para los estudiantes.

Se utilizará la plataforma TIA Portal para PLC Siemens, VisiLogic para el PLC marca Unitronics y Arduino IDE, para la programación de las prácticas propuestas. Estas herramientas de software procesarán y transmitirán los datos de las salidas y entradas como motores, indicadores, selectores, sensores, etc.

Palabras claves: MODBUS/TCP, PLC, SCADA.

ABSTRACT

YEAR	STUDENT	ADVISOR	TITLE
2022	NICOLAS EFRAIN CRESPO DELGADO	ING. MONICA MIRANDA RAMOS	DESIGN AND IMPLEMENTATION OF MODBUS TCP/IP NETWORK MODULES BETWEEN THREE PROGRAMMABLE AUTOMATES FOR LOCAL AND REMOTE THREE- PHASE MOTOR START-STOP AND SENSORS READING.

The proposed technical project has the main objective of teaching the communication technique between three different programmable robots, in this case with the MODBUS/TCP-IP protocol, in addition to the frequency variation of a three-phase motor. It also includes reading sensors from the Arduino module.

I will give the details of the protocol usage, the circuit design for motor speed variation and the advantages and disadvantages of this project in general.

This will be achieved through three didactic modules equipped with, PLC Unitronics Vision 700, PLC SIEMENS S7-1200, an ARDUINO MEGA board coupled with an ARDUINO ETHERNET, a THREE-PHASE MOTOR, antennas UBIQUITI NANOSTATION LOCO M5 as a means of communication between stations, personal computer.

The modules will serve for teaching staff and students, since the project will be divided into various levels of difficulty for the development of the proposed practices, which will serve as new academic material for students.

Use the platform Tia Portal for Siemens PLC, VisiLogic for the Unitronics brand PLC and Arduino IDE, for the programming of the proposed practices. These software tools process and transmit the data of the outputs and inputs such as motors, indicators, selectors, sensors, etc.

Keywords: MODBUS / TCP, PLC, SCADA.

INTRODUCCIÓN

Las redes de comunicaciones automáticas y programables de manera industrial en lugar de la tecnología alámbrica y tradicional resultan muy conveniente a nivel de comodidad, configuración y ahorro de recursos económicos, siendo la comunicación entre los dispositivos de esta red transportada por medio de protocolos de comunicación industriales, donde viaja la información de manera más eficiente que un sistema tradicional o ya antiguo. (Sicma21, 2021)

El objetivo de este proyecto es implementar en el laboratorio de automatización industrial de la UPS-G, tres módulos que ayudarán a los estudiantes a conocer sobre la importancia de las redes y comunicaciones de autómatas programables en la industria, como también el conocimiento básico de variadores de frecuencia y lectura de sensores.

El protocolo MODBUS/TCP-IP es una variante de MODBUS para protocolos de comunicaciones industriales sencillas para equipos de automatización y control. MODBUS utiliza mensajes bajo un modelo de “intranet” o de “internet” en el cual varios dispositivos estarán comunicados entre sí.

Los equipos implementados permitirán hacer prácticas de diferentes niveles de aprendizaje, entre estos estarán las comunicaciones entre autómatas como parte de la materia de redes, el uso de variadores de frecuencia para motores como parte de la materia de circuitos industriales y el manejo de sensores básicos a través de una placa de microcontrolador, en este caso Arduino Mega. También se podrá incluir sistemas SCADA y configuraciones maestro / esclavo donde el alumno podrá decidir cuál de los dispositivos podrá comandar en varios elementos a través del equipo maestro. Se podrá comprender tres tipos de programación para los dispositivos.

El proyecto ofrecerá a los docentes, prácticas completas de automatización industrial y a los alumnos a comprender la importancia de las comunicaciones entre dispositivos diferentes y realizar sistemas de control SCADA para mejorar el desempeño práctico para la preparación profesional.

CAPÍTULO 1: EL PROBLEMA

1.1 Planteamiento del problema

La Universidad Politécnica Salesiana Sede Guayaquil en su afán de formar nuevos profesionales que puedan cumplir con las exigencias laborales requiere de herramientas para incrementar los conocimientos de la carrera de ingeniería, por ello se presentará el **Diseño e implementación de módulos de red MODBUS/TCP entre tres autómatas programables para arranque de motor trifásico de manera local, remoto y lectura de sensores, para el laboratorio de automatización industrial de la Universidad Politécnica Salesiana**, que contiene módulos didácticos para practicas tecnológicas de los estudiantes.

Las herramientas que en este momento se encuentran al alcance de los estudiantes limitan en cierta medida actualizarse con la integración de distintas marcas de PLCs y controladores que trae el mercado para la industria y esto representa una desventaja para los futuros profesionales de la Universidad Politécnica Salesiana frente a las comunicaciones industriales inalámbricas basadas en MODBUS TCP/IP.

Los laboratorios de la Universidad Politécnica Salesiana Sede Guayaquil, estarán equipados para contener diferentes tipos de marcas de PLC en este caso, gracias a la implementación de una red MODBUS TCP/IP, el módulo de PLC Unitronics y el módulo con placa de programación Arduino se podrán comunicar con los PLC Siemens y computadoras de los laboratorios de la carrera de ingeniería.

El proyecto irá dirigido a los alumnos de diferentes niveles de la carrera de ingeniería, y a los docentes de la Universidad Politécnica Salesiana de Guayaquil.

1.2 Delimitación del problema

El Proyecto de Titulación se desarrolló en la Universidad Politécnica Salesiana sede Guayaquil, Ecuador en el período 2019-2021.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar e implementar una red de Comunicación Industrial para control y monitoreo de procesos industriales inalámbricos usando el protocolo MODBUS TCP/IP en tres estaciones las cuales incluye los controladores: Siemens S7-1200, PLC Unitronics Vision700 y Arduino Mega con Shield Ethernet.

1.3.2 Objetivos específicos

1. Diseñar los módulos didácticos para los estudiantes de Ingeniería Electrónica con mención en Automatización Industrial que pueda ser utilizado para prácticas universitarias.
2. Elaborar un manual de diez prácticas guiadas, para ser utilizada por docentes y estudiantes del laboratorio de Automatización Industrial de la Universidad Politécnica Salesiana Sede Guayaquil.
3. Elaborar una red de Comunicación Industrial usando el protocolo MODBUS TCP/IP capaz de controlar y monitorear a través de SCADA-HMI, la velocidad de un Motor Trifásico desde cualquier estación.
4. Interactuar con el medio físico a través de sensores compatibles con el módulo de Arduino con Shield Ethernet; temperatura, humedad, movimiento.
5. Demostrar en aplicaciones reales la interacción inalámbrica entre los tres módulos que ofrece las antenas de comunicación Ubiquiti nano Station Loco M5 y el protocolo MODBUS TCP/IP.

1.4 Justificación del proyecto

Actualmente en el laboratorio de Automatización Industrial de la Universidad Politécnica Salesiana sede Guayaquil se encuentra mejorando sus instalaciones ofreciendo equipos en el área Industrial, por lo cual la Universidad les ofrece la oportunidad a los estudiantes de la Carrera de Ingeniería Electrónica de realizar un trabajo de Titulación con estos nuevos equipos, diseñando e implementando módulos didácticos con el fin de complementar los conocimientos teóricos y prácticos.

En vista de lo anterior, se propone el desarrollo de dos módulos didácticos con un PLC Siemens que se adicionará una antena Ubiquiti nano loco M5, PLC Unitronics Visión700, Arduino con Shild Ethernet, para prácticas universitarias con el uso del Node Red, VisiLogic, Arduino IDE, Labview y Matlab, el mismo que se realizará en el Laboratorio de Automatización Industrial de Guayaquil. Esto ayudará a los estudiantes, ya que les permitirá entrenar el desarrollo de aplicaciones industriales, mejorar sus habilidades y dominar la formación de los futuros ingenieros introduciéndolos en la programación de PLCs

La gran importancia de poder contar con tecnología de punta es de gran ventaja para la comunidad universitaria debido a que la calidad académica mejora en sobremanera, por lo cual el proyecto de titulación posibilita las herramientas necesarias para explotar las funcionalidades de los equipos adquiridos por la universidad, las redes autónomas y la programación en PLCS son bases principales en la formación profesional de la carrera, siendo de igual manera de gran provecho en el ámbito laboral de los futuros egresados.

1.5 Hipótesis

Los dispositivos de programación de procesos serán capaces de mantener una comunicación continua entre ellos mediante una red industrial inalámbrica a implementarse

Estos dispositivos de programación de procesos se comunicarán creando una red MODBUS TCP/IP y transmitida inalámbricamente con antenas de comunicación, el protocolo MODBUS permite realizar configuraciones para comunicación de maestros y esclavos con dos o más equipos hasta 247 esclavos.

1.6 Variable e Indicadores

1.6.1 Variables

- Señales analógicas, luces indicadoras, selectores, indicadores.
- Comunicación inalámbrica de datos.
- Señales Digitales.

1.6.2 Indicadores

- Salidas digitales y analógicas.
- Visualización de registros MODBUS por medio de software.
- Visualización de estados digitales por medio de luces indicadoras

1.7 Metodología

1.7.1 Método Inductivo

Con este método se pondrá a prueba los conocimientos de materias como; Instalaciones Industriales, Redes de Computadoras II y III, Automatización Industrial I y II, Sistemas Microprocesados, Sensores y Transductores, Teoría de Control, Informática Industrial, programación utilizando TIA Portal de Siemens para el desarrollo del SCADA, VisiLogic para programación de PLC Unitronics, Arduino IDE para programación de estación Arduino y sensores, redes y comunicaciones para diseño de red multipunto para estaciones que se comunicaran a través de MODBUS TCP/IP, Matlab y Labview para modelado matemático necesario para control PID y Fuzzy Logic Control System y que servirán para el desarrollo de los módulos didácticos y su programación.

1.7.2 Método Teórico y Sistemático

- Funcionamiento del PLC Unitronics.
- Funcionamiento de PLC Siemens S7-1200 y Tia Portal.
- Funcionamiento de Microcontroladores y Arduino.
- Funcionamiento de un variador de frecuencia.
- Software de programación.

- Investigación de información referente al tema en internet.
- Búsqueda de proveedores de equipos electrónicos e industriales.

Estos puntos mencionados han sido considerados para la realización de este proyecto, incluyendo soluciones rápidas debido a la crisis sanitaria que ha venido afectando en el proceso de construcción de los módulos y sus prácticas demostrativas.

1.7.3 Técnica Documental

Se recopiló información para tener en cuenta fundamentos teóricos de todos los elementos que conforman el proyecto a implementar y que se basó en libros y páginas encontradas en internet para esta técnica.

1.8 Población y Muestra

1.8.1 Población

La población son todos los estudiantes de 5to ciclo en adelante de la carrera Ingeniería Electrónica y afines de la Universidad Politécnica Salesiana.

1.8.2 Muestra

Como muestra están los estudiantes de la carrera de Ingeniería Electrónica que están en materias como: Automatización I y II, ya que dentro del contenido esta la programación básica de PLC, Microprocesados I y II, ya que al tener una plataforma de código de micro los alumnos podrán analizar los diferentes usos, Redes de Computadora II y III, debido a que en estas materias en contenido académico está el estudio protocolos de red y protocolos de comunicación industrial, Sensores y Transductores para el estudio de diferentes Sensores básicos y de la industria, Teoría de control e Informática Industrial, ya que los alumnos podrán aplicar los conocimientos dados en control de procesos, también Instalaciones Industriales ya que se estudian diferentes conexiones y dimensionamientos eléctricos Industriales.

1.9 Descripción de la propuesta

Se implementaron dos maletas didácticas y una estación que contiene un PLC Siemens S7-1200, que servirán para el aprendizaje de comunicaciones industriales, fundamentos de electrónica, internet de las cosas y automatización industrial. Estas maletas constan de controladores como Arduino y PLC Unitronics, módulo de relés en módulo Arduino, variador de frecuencia, pulsadores, selectores, antenas de comunicación, sensores, motores, banda transportadora.

Las dos maletas constan de protecciones como disyuntores y fuentes de poder.

1.10 Beneficiarios

Los beneficiarios de las maletas didácticas serán los estudiantes de la carrera Ingeniería Electrónica de la Universidad Politécnica Salesiana de ciclos superiores y básicos.

1.11 Impacto

Los alumnos podrán manipular los equipos y aprender nuevas técnicas académicas para el conocimiento de redes Industriales, desarrollos Web, y protocolos de red.

CAPÍTULO 2: MARCO TEÓRICO

2.1 PLC Unitronics Visión 700

Unitronics fue fundado en 1989 para la industria de control de automatización y hace dos décadas lanzó el primer controlador todo en uno, un PLC HMI + E/S integradas del cual se escogió la serie Visión para este proyecto.

El Visión700 tiene 7 pulgadas, viene integrada una pantalla táctil de 800 x 480 píxeles este PLC ha sido desarrollado para abarcar la gran exigencia de utilizar pantallas HMI integradas en los equipos de control, este modelo tiene varias configuraciones de E/S como: digitales, digitales de alta velocidad, analógicas, termopar PT100, entre otras. (Colsein, 2022)

Las E/S se puede expandir usando módulos de expansión, estas pueden ser locales y remotas que pueden alcanzar hasta 1000 metros, este PLC contiene algunas formas básicas de comunicación como: Ethernet, y protocolos de comunicación industrial como BACnet, CANopen, esclavo DF1, Modbus TCP y RTU, comunicaciones con puerto RS485 y RS232, Profibus, tarjeta SD para registro de datos y realizar copias de seguridad, y su puerto de programación que puede servir también de comunicación via USB mediante puerto de comunicaciones. (Colsein, 2022)



Figura 2.1: Autómata programable Unitronics modelo V700

Fuente: (Unitronics, s.f.)

2.1.1 Especificaciones técnicas modulo V700-T20BJ

El PLC Unitronics es de tipo modular, lo que significa que solo cuenta con el CPU, y se puede conectar a varios módulos por medio de un bus dependiendo del uso que se requiera. Los módulos más comunes son de entradas y salidas digitales y entradas analógicas y salidas analógicas.

Tabla 1: Especificaciones PLC

Fuente: (Unitronics, s.f.)

Fabricante	Unitronics
Modelo	V700-T20BJ
Tipo de base	Modular
Tamaño de memoria	Application Logic – 2MB, Images – 60MB, Fonts – 1MB
Tiempo de ciclo	9 micro segundos
Número de contadores	384
Número de temporizadores	32
Comunicaciones	Modbus TCP/IP Profibus/device CANbus BACnet
Modulo I/O: modelo	V200-18-E2B
Entradas	16 entradas digitales, 2 high-speed, tipo pnp/npn
Salidas	10 salidas tipo relé, 4 salidas de transistor tipo pnp/npn, 2 high-speed output
I/O Analógicas	2 entradas analógicas / 2 salidas analógicas
Tipo de panel del operador	TFT LCD
Resolución de la pantalla	800 x 480 pixeles, 7”
Touchscreen	Resistivo, análogo
Fuente de poder	12/24VDC



Figura 2.2: Modulo I/O modelo V200-18-E2B

Fuente: (Unitronics, s.f.)

2.1.2 Software de programación de PLC Unitronics

Unitronics utiliza VisiLogic para programar PLC's de la serie Vision. El software utiliza lenguaje de programación en escalera que están compuestos de contactos, bobinas y elementos de función para redes. La licencia de Visilogic es gratuita y se puede obtener desde la página web de Unitronics.

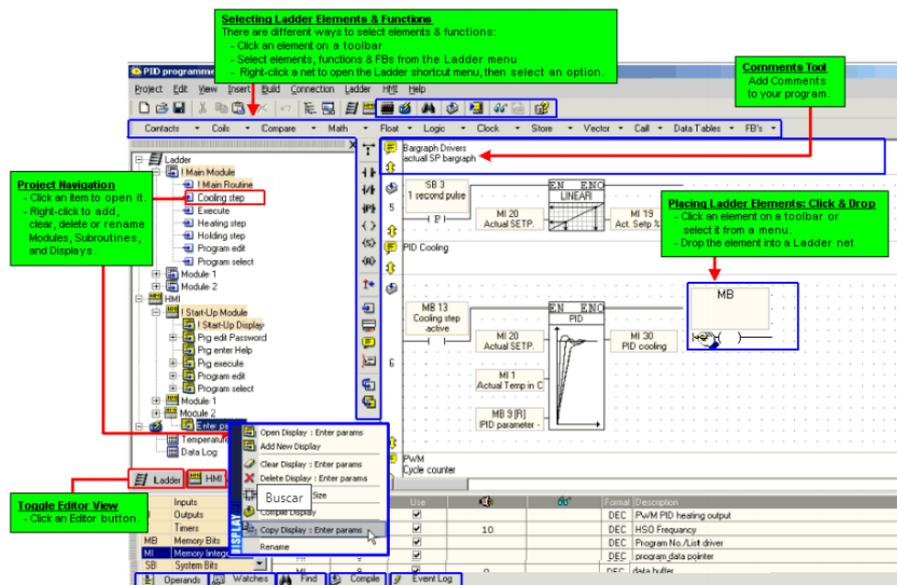


Figura 2.3: Software de programación VisiLogic.

Fuente: (VisiLogic)

2.2 Variador de frecuencia

(Fernando Sevillano, 2011). Los variadores de frecuencia son uno de los tantos dispositivos electrónicos que logran controlar motores eléctricos que funcionan a inducción, existen de tipo de corriente continua que controlan el voltaje y de corriente alterna que controlan la frecuencia; los motores que se utilizan en la industria y aplicaciones en general son de tipo trifásicos de inducción y rotor sin bobinar (Jaula de ardilla). Los variadores de frecuencia también se los nombra como inversores o variadores de velocidad.

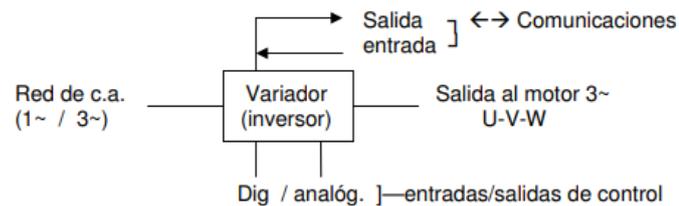


Figura 2.4: Esquema E/S de un variador de frecuencia

Fuente: (Sevillano, 2011)

Los variadores de frecuencia contienen en su interior que permiten su funcionamiento:

1. Rectificador: a partir de una entrada de corriente alterna, monofásica o trifásica se transforma a corriente continua usando como dispositivo electrónico diodos rectificadores.
2. Bus de continua: aquí se utilizan capacitores o también llamados condensadores de capacidad alta para estabilizar la tensión y hacerla continua, también almacena energía necesaria para proveer intensidad requerida por el motor.
3. Etapa de salida: luego de que el bus de continua cumple su etapa, un ondulator transforma el voltaje proveniente en una salida de voltaje trifásica, con tensión, amperaje y frecuencia variables. Se utilizan transistores bipolares (BJT), CMOS, IGBT, tiristores (SCR), entre otros, como conmutadores, el procedimiento de troceado se encarga de la señal de salida, también pueden ser por convertidores de ciclo, o por modulación de ancho de pulsos (PWM).
4. Control y E/S: dentro de los variadores de frecuencia se encuentran mandos que el usuario puede manipular para controlar diferentes bloques del variador, estos están constituidos por circuitos de control de protección, regulación, entradas y salidas

digitales y analógicas, pantalla de visualización, buses de comunicación y otros dispositivos de control que el usuario puede añadir. (Fernando Sevillano, 2011).

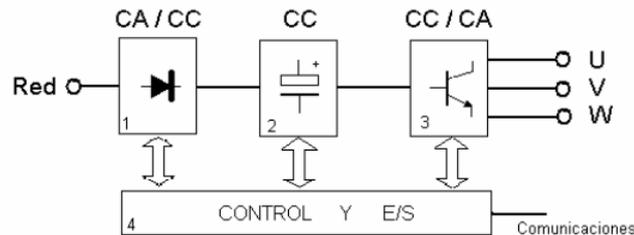


Figura 2.5: Esquema interno de un variador de frecuencia

Fuente: (Sevillano, 2011).

2.2.1 Concepto básico de variadores de velocidad para motores

Dependiendo de los polos magnéticos del motor, la frecuencia (Hz) que entrega la red de voltaje, se puede obtener la velocidad en rpm del eje de un motor asíncrono.

$$n = 60 \frac{f}{2p} \quad (1)$$

Donde;

n = velocidad (rpm)

f = frecuencia (Hz)

2p = número de pares de polos del motor

En la industria los polos que más se encuentran en Motores síncronos son 2,4,6 y 8 polos que, incluidos a la ecuación anterior, tenemos como respuesta 1500 rpm, 3000 rpm y 750 rpm, estas velocidades en motores síncronos con frecuencia de 50 Hz. Esto también depende de que país es la red de voltaje donde la frecuencia puede ser 50 Hz o 60 Hz.

2.3 Fuente de Alimentación

Para alimentar el PLC es necesario 12 o 24VDC. Se utilizará una fuente de la marca Mean Well serie DR-60 la cual tiene un gran desempeño a nivel industrial y genera poco calor, así como protecciones de sobrecargas y cortocircuitos.



Figura 2.6: Fuente de alimentación MeanWell 24VDC

Fuente: (MeanWell)

2.4 Disyuntor

También llamados interruptores automáticos que se accionan por sobrecargas dentro de un circuito, o también son interruptores automáticos activados por pérdidas de corriente o voltaje fuera de un circuito.

Dependiendo del país pueden ser llamados Disyuntores, interruptores automáticos, automáticos, taco o breaker, este es un dispositivo que interrumpe o abre un circuito eléctrico cuando suceden fallas de aislación en una maquina o instalación eléctrica.

El disyuntor puede ser rearmado una vez que se detecte la falla y sea arreglada en el circuito y eso lo diferencia de un fusible que debe ser cambiado cuando detecta una falla.

Se fabrican en diferentes tamaños y los hay para viviendas, industrias y comercios.

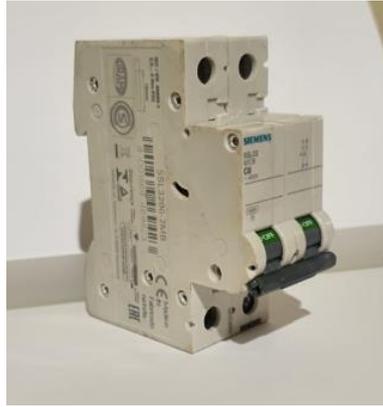


Figura 2.7: Disyuntor magneto térmico monofásico bipolar

Fuente: (El Autor)

2.5 Luces indicadoras

Se instala con el fin de indicar si un panel de control o fuente está activo o simplemente para indicar un proceso.



Figura 2.8: Luces piloto industriales

Fuente: (El Autor)

2.6 Pulsadores y selectores

Son dispositivos de mando que permiten el paso del flujo eléctrico y que el operador puede ordenar ejecución de operaciones tales como arranque, parada de motores, etc.

El pulsador es un elemento de conmutación es decir se conecta y se desconecta manualmente al aplicarle presión, su contacto solo tiene una única posición. Al pulsarlo esta posición cambia, y cuando se deja de pulsar, retoma a su posición usando un retorno de muelle o resorte que está dentro del pulsador.

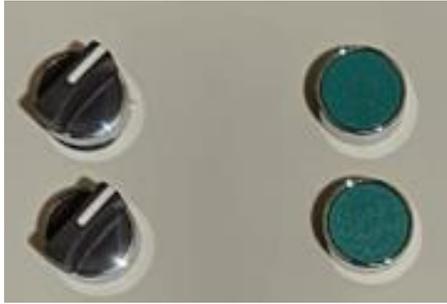


Figura 2.9: Selectores y pulsadores industriales

Fuente: (El Autor)

El selector es un elemento de conmutación monoestable con dos o más posiciones que el operador puede elegir, y que para su accionamiento suelen llevar una palanca o llave giratoria que puede ser extraíble.

2.7 Tomacorriente y toma USB

Es un dispositivo de corriente eléctrica donde se conectan distintos equipos eléctricos, los hay de diferentes tipos como empotrados a la pared y sobrepuesto.

Las ranuras de conexión dependen de la región donde este ubicado y se los clasifica por su tipo.

Se han adaptado tomas USB (Universal Serial Bus), para cargar dispositivos o toma de datos desde la pared.

2.8 Motor Trifásico

Los motores trifásicos son maquinas eléctricas que convierten energía eléctrica en energía mecánica mediante interacción electromagnética, y están diseñados para operar con corriente alterna (AC) trifásica que se utiliza en muchas aplicaciones industriales. La electricidad de corriente alterna viaja de negativo a positivo y viceversa varias veces por segundo. (Motorex, 2020)



Figura 2.10: Motor trifásico.

Fuente: (Motorex, 2020)

2.8.1 Partes de un motor trifásico

El motor trifásico se puede dividir en 3 componentes: el rotor, el estator y los escudos/carcasa.

- **El estator:** sirve como parte fija y base de motor, está constituido por la carcasa donde se encuentran las coronas de chapas de material de acero, silicio o acero, aquí se encuentran unas ranuras donde se presentan, cuando es un motor trifásico se encuentran en su interior tres bobinas y tres circuitos todos diferentes y en cada uno hay bobinas como polos armados del motor.
- **El rotor:** está constituido por un núcleo magnético ranurado de material de acero, en estas ranuras van unas barras de cobre o aluminio y sirven de conductores en una disposición que se llama jaula de ardilla. Esto se genera a que las barras están entrelazadas en cortocircuito por dos anillos rodeando la estructura.
- **Escudos o carcasa:** es la parte de afuera del motor, el material de la carcasa suele ser de aluminio o hierro colado. Esta diseñado de tal forma que su estructura permite acoger todos los componentes internos del motor, sobre los cojinetes se encuentra el eje del rotor. Además, esta estructura está diseñada para prevenir distorsiones cuando el rotor gira, vibraciones y ruido producido. (S&P, 2019)

2.8.2 Funcionamiento de un motor trifásico

Cuando el bobinado de tres fases recibe corriente eléctrica, genera un campo magnético que también induce corriente a las barras del rotor. El fenómeno que se atribuye a este funcionamiento se llama principio de inducción mutua de Faraday.

El campo magnético se genera por la aplicación de corriente alterna trifásica, esta electricidad de corriente alterna que es una onda sinusoidal, cambia de negativo a positivo muchas veces por segundo.

La corriente alterna de tres fases están desfasadas a 120° . Para que el rotor gire, las tres ondas simultáneamente generan un flujo magnético que inducen corriente a las barras del rotor creando par motor. (S&P, 2019)

2.8.3 Ventajas un motor trifásico

- Son ligeros y pequeños pudiendo igualar a motores de combustión.
- El rendimiento de estos motores es más alto ya que su par de giro es constante y elevado.
- Necesitan muy poco mantenimiento.
- Depende de la necesidad se pueden realizar de diferentes tamaños ya que son escalables. (S&P, 2019)

2.9 Arduino

Es una plataforma electrónica de código abierto basada en hardware y software de construcción sencilla. Las placas de Arduino contienen entradas y salidas digitales que son capaces de leer botones, switches y sensores digitales, estas pueden ser configuradas de cualquier forma desde la programación, también cuenta con salidas y entradas analógicas capaces de leer sensores analógicos y variar velocidad de motores DC mediante PWM de bajo voltaje y amperaje. Su lenguaje de programación es sencillo e intuitivo. (Arduino, 2018)

La tarjeta electrónica digital de hardware, está construido por un microcontrolador de tipo AVR y esta es una de las más utilizadas en la construcción de estos prototipos. (Estrada, 2017)

Por otra parte, Estrada (2017) define a Arduino de la siguiente manera:

Una herramienta de procesamiento digital similar a un ordenador. Por lo tanto, cuenta con elementos de entrada o salida digital a los cuales se les puede conectar componentes electrónicos como sensores, teclados, pantallas LCD, entre otros. Adicional el controlador incorpora entradas analógicas útiles en la medición de señales en sensores análogos. Para poder visualizar la información, se cuenta con un puerto de comunicación Serial-USB que mediante un puerto USB, de una computadora, nos permite el envío y recepción de mensajes mediante una USART o también llamada UART. (Estrada, 2017)

2.9.1 Características de Arduino

Entre los grandes beneficios que trae trabajar con Arduino se destacan los siguientes:

- **Bajo Costo:** Los prototipos de Arduino son económicos al alcance de cualquier tipo de usuario sea este inexperto o avanzado.
- **Multiplataforma:** Funcionan en todos los sistemas operativos comerciales existentes.
- **Entorno de programación sencillo y funcional:** Su diseño está inspirado para todo público que desea incursionar en el mundo de la electrónica, por lo tanto, es una herramienta de código abierto para aumentar la experiencia entre programadores.
- **Hardware de código abierto y extensible:** Datos de código y construcción del prototipo están publicados bajo licencia de Creative Commons para que los diseñadores de circuitos puedan crear sus propias versiones de los módulos y adaptarlos a estos. (Arduino, 2018)

2.9.1.2 Ventajas y Desventajas de Arduino

Ventajas

- No se necesita se programador experto para manejar un Arduino.
- El uso del hardware y software son libres.
- Su programación está basada en C, se puede encontrar también una gran cantidad de librerías en la web.
- Existen dispositivos denominados Shield que son compatibles con aplicaciones de Arduino.

Desventajas

- El lenguaje no es tipo ensamblador por lo que se usan librerías que al momento de la ejecución de un programa pueden generar retrasos en la ejecución de un programa principal.
- El controlador ya está prediseñado por lo que resulta más complicado adaptar o personalizar alguno a conveniencia. (Martinez, 2015)

2.9.2 Arduino Mega

Arduino Mega está basado en microcontrolador ATmega2560, cuenta con 54 pines de E/S digitales de los cuales 15 pueden configurarse como salidas PWM, también contiene 16 entradas analógicas. 4 puertos serie UART, un oscilador de cristal de 16 MHz, conexión USB, conector de entrada de voltaje, ICSP y un botón de reinicio de placa. (Arduino, 2020)

Tabla 2: Especificaciones Arduino Mega 2560

Fuente: (Arduino, 2020)

Microcontrolador	ATmega2560
Tensión de alimentación	5 V
Tensión de entrada (recomendado)	7V - 12V
Tensión de entrada (limite)	6V - 20V
E/S digitales	54 - 15 de PWM
Entradas analógicas	16
Corriente DC de E/S	20mA

Corriente DC para salida 3.3V	50mA
Memoria flash	256 KB los cuales 8 KB son usados por el gestor de arranque
SRAM	8KB
EEPROM	4KB
Velocidad de reloj	16 MHz
LED_BUILTIN	13
Longitud	101,52 milímetros
Anchura	53,3 milímetros
Peso	37 gramos

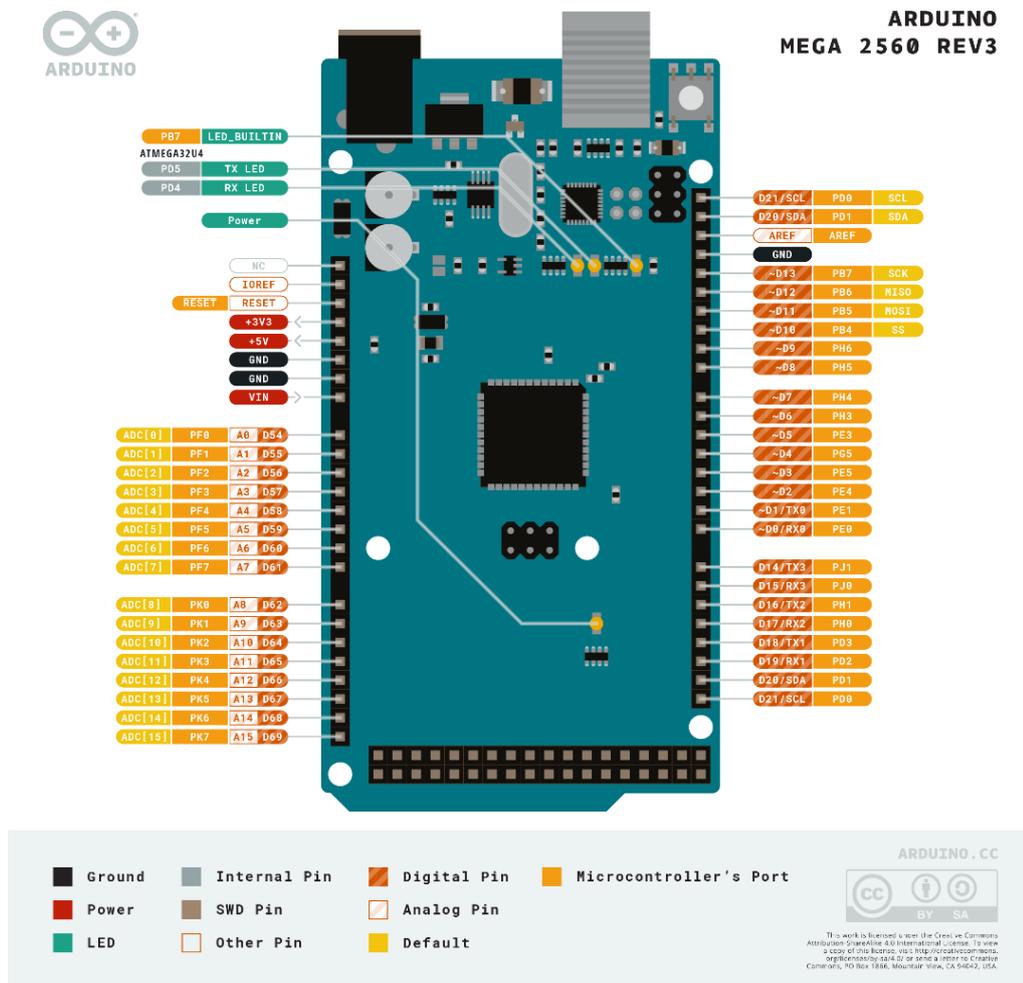


Figura 2.11: Entradas y salidas de Arduino Mega

Fuente: (Arduino, 2020)



Figura 2.12: Arduino Mega 2560

Fuente: (Arduino, 2020)

2.9.3 Módulo Arduino Ethernet Shield

Esta placa permite a un Arduino conectarse a internet. Se debe colocar este módulo a un Arduino, conectar el cable RJ45 a la red y seguir las instrucciones de programación para su aplicación usando la red, está basada en el microchip Ethernet Wiznet, este equipo ofrece una conexión IP que puede usar TCP y UDP. Tiene capacidad de usar hasta cuatro comunicaciones de socket simultaneas. Para usar las aplicaciones de red en Arduino es necesario usar la biblioteca de Ethernet. (Arduino, 2020)

Este Shield es compatible con Arduino Uno y Arduino Mega, la comunicación del Shield W5100 como su tarjeta SD se realiza mediante comunicación de tipo bus SPI a través de la cabecera ICSP, para ello los pines digitales 10,11,12,13 en Arduino Uno y los pines 50,51,52 en Arduino Mega. Estos pines no se pueden utilizar para otra entrada o salida (Arduino, 2020)

El módulo Arduino Shield Ethernet contiene una serie de LEDs informativos que serán necesarios para revisión de errores y conexiones en los proyectos:

- PWR: indicador de placa y el módulo encendido.
- LINK: Indicador de presencia de enlace de red y es intermitente cuando transmite o recibe datos.
- FULLD: indicador de que conexión de red es full duplex.
- 100M: indicador de conexión de red es a Mb/s 100.
- RX: intermitencia cuando el Shield recibe datos.
- TX: intermitencia cuando el Shield envía datos.
- COLL: parpadea cuando existen colisiones de red. (Arduino, 2020)

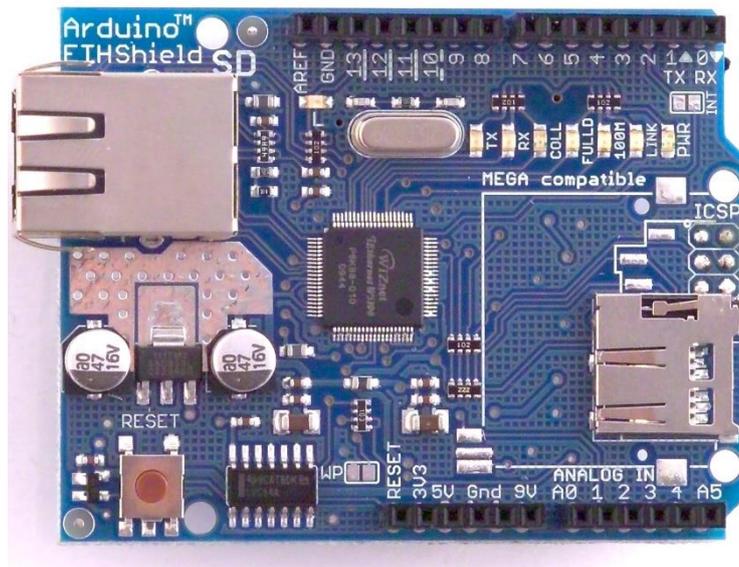


Figura 2.13: Shield Ethernet

Fuente: (Web Robotica, s.f.)

2.10 Potenciómetros

Un potenciómetro es un tipo de resistencia variable mecánico donde el usuario al armar el circuito de un potenciómetro y al manipular la perilla central, este varía su voltaje ya que el dispositivo actúa como un divisor de tensión. (helloauto, 2022)



Figura 2.14: Potenciómetro

Fuente: (El autor)

2.11 Display LCD con comunicación I2C

El chip I2C está basado en el controlador HD44780 de Hitachi, que es un periférico de uso común, que se utiliza para proyectos en su mayoría con Arduino y chips microcontroladores. Se sabe que para la conexión de la LCD a Arduino es necesario el uso de muchos pines de E/S de cualquier microcontrolador, para esto se utiliza el adaptador PCF8574 que permite realizar la conexión al microcontrolador usando solo dos conexiones digitales a través del bus I2C. Arduino mejora la experiencia de

comunicación en el software usando la biblioteca LiquidCrystal_I2c para su programación. (Geekfactory, 2017)

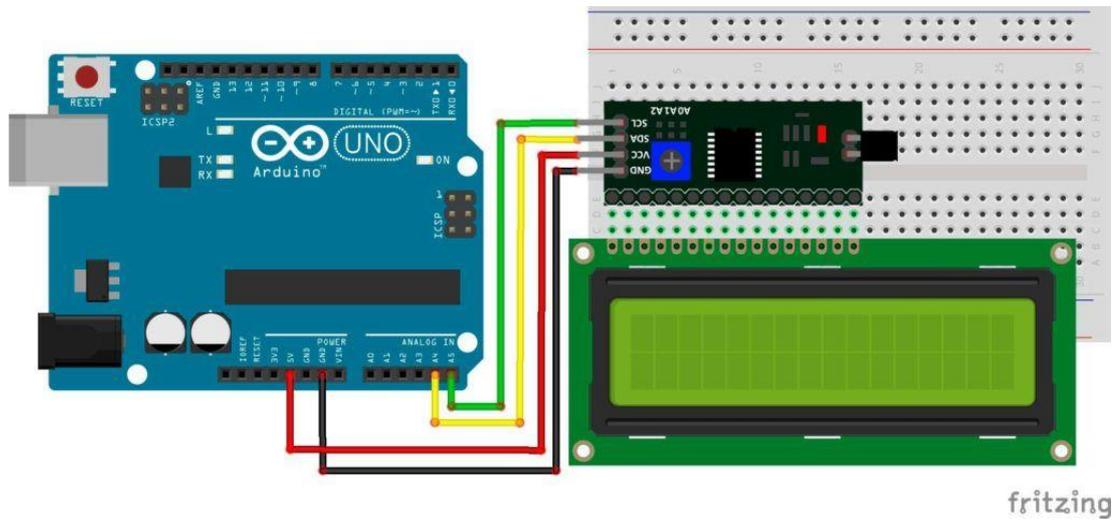


Figura 2.15: Diagrama de conexión LCD 16x2 por I2C

Fuente: (Geekfactory, 2017)

2.12 Sensores

Los sensores son dispositivos capaces de detectar la magnitudes físicas o químicas, y convertirlas en variables eléctricas, por ejemplo, las variables pueden ser: temperatura, intensidad de luz, distancia, posición, humedad, movimiento, etc. La energía eléctrica puede ser una resistencia (RTD), capacidad eléctrica como un sensor de humedad o de tipo capacitivo, y de voltaje como los termopares o foto transmisores. (Gines, 2019)



Figura 2.16: Kit de módulo de sensores y actuadores.

Fuente: (Gines, 2019)

2.12.1 Sensor Ultrasónico

Los sensores ultrasónicos miden la distancia usando ondas ultrasónicas, un emisor envía una onda y recibe con el receptor la acción que hace la onda al chocar con una superficie, esta acción se realiza al medir la distancia entre el objeto contando el tiempo entre la emisión de la onda y su recepción. (Keyence, 2022)

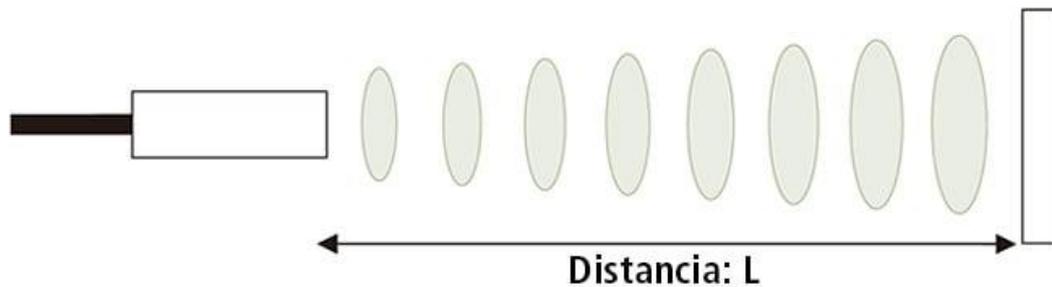


Figura 2.17: Dibujo técnico sensor ultrasónico.

Fuente: (Keyence, 2022)

La distancia es calculada con la formula: $L = (0.5 * T * C)$ (2), donde C es la velocidad del sonido, T el tiempo de emisión y recepción del sonido. (Keyence, 2022)

Ventajas:

- No necesita contacto con el objeto a censar.
- Sensible a objetos frágiles.
- Detecta cualquier material de cualquier color.
- Tienen un rango de error pequeño, dependiendo de la marca.

Desventajas:

- Son sensibles a los ruidos
- No detectan superficies de tela con facilidad
- Tienen puntos ciegos y pueden generar falsas alarmas.

2.12.2 Sensor Temperatura DS18B20

El sensor de temperatura DS18B20 lleva dentro de su encapsulado un circuito de conversión analógico a digital, que luego se puede utilizar usando un microcontrolador con protocolo 1-wire para poder tener la información de la temperatura desde el sensor, como el nombre del protocolo lo indica, solo se necesita un cable de línea de datos

además de la alimentación para realizar la comunicación sensor – microcontrolador, se puede además adicionar más dispositivos dentro de este bus de conexión. (Factory, 2019)

Características de sensor DS18B20:

- Rango de temperatura de -55 a 125° C
- Resolución digital de 9 a 12 bits configurables
- Protocolo 1-Wire
- Identificador único de 64 bits
- Múltiple conexión de sensores al mismo puerto
- Precisión de $\pm 0.5^{\circ}\text{C}$
- Tiempo de captura de 750ms
- Alimentación de 3V a 5.5V. (Bricogeek, 2022)

2.12.3 Sensor de Luz (Fotorresistencia)

La fotorresistencia componente electrónico en el cual su resistencia varia, por lo general su valor disminuye ante la intensidad de luz, son llamadas también célula fotoeléctrica o resistor dependiente de luz. (Portilla, 2015)

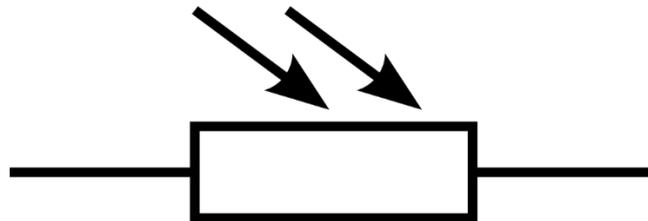


Figura 2.18: Diagrama de sensor de Luz o Fotorresistencia.

Fuente: (Portilla, 2015)



Figura 2.19: Fotorresistencia.

Fuente: (Portilla, 2015)

2.12.4 Sensor Infrarrojo

Los sensores infrarrojos son componentes electrónicos que tienen un LED infrarrojo y un fototransistor colocados juntos, de forma que uno de los LED actúa como emisor y el otro como receptor. El LED infrarrojo emite un tipo de luz que no se puede observar por un humano ya que es de una longitud de onda mayor o es de menor frecuencia. Si esta luz choca con una estructura de color blanco se reflejará y llegará al fototransistor. Si por el contrario golpea una estructura negra, el material absorbe la mayoría de la luz y no reflejará la luz al fotorreceptor. (prometec, 2022)



Figura 2.20: Sensor Infrarrojo.

Fuente: (prometec, 2022)

2.12.5 Sensor de flujo de líquidos YF-S201

El sensor de flujo o también llamado caudalímetro es una herramienta instrumental para medir el flujo, caudal o gasto volumétrico que pasa por un canal. El caudal es la cantidad de líquido que circula a través de una tubería por unidad de tiempo, y que su unidad se expresa en litros por minuto (l/m), litros por hora (l/h) o metros cúbicos por hora (m³/h). (Mechatronics, 2021)

Un caudalímetro se suele colocar en una tubería de líquido, el sensor de flujo ½" YF-S201 fue diseñado para medir flujo de agua en tuberías de ½" de diámetro, pueden ser implementados para medir otros tipos de fluidos como: bebidas alcohólicas, bebidas con gas, combustible, etc. Este sensor electrónico contiene una turbina y es compatible con sistema de control digitales de bajo costo tales como: Raspberry, PLC, Arduino entre otros. Este sensor tiene tres cables, rojo para alimentación a 5V, el cable negro corresponde a GND y el cable color amarillo es la señal digital de pulsos cuyo nombre se conoce como efecto Hall. (Mechatronics, 2021)

Su funcionamiento es simple, el caudal pasa por el sensor y esta acción hace que gire la turbina que está dentro del sensor a un sensor de efecto Hall que emite pulsos cuando la turbina completa una vuelta, este se conecta a un pin de entrada digital de Arduino quien se encargara de el control de este teniendo en cuenta los pulsos generados o pulsos promedios para determinar valores tales como volumen, flujo constante. La Fórmula matemática de este sensor es: Flujo de líquido en L/m = Pulsaciones del sensor en frecuencia (Hz) / 7.5 (Mechatronics, 2021)



Figura 2.21: Sensor de flujo.

Fuente: (Mechatronics, 2021)

Características:

- Modelo YF-S201
- Voltaje: 5 a 18 VDC
- Corriente máxima: 15 mA a 5VDC
- Salida: 5V TTL
- Flujo de trabajo: 1 a 30 Litros / min
- Rosca: ½" NPS
- Material de plástico
- Pulsos por litro: 450
- Máxima presión de entrada de flujo: 2 MPa
- Factor de conversión: 7.5 (Mechatronics, 2021)

2.13 PLC Siemens S7-1200

El controlador lógico programable S7-1200 es uno de los más cotizados del mercado de la automatización industrial por su potencia y flexibilidad para abarcar distintas necesidades dentro de la industria. Esta CPU incluye un microprocesador, fuente de poder incorporada en algunos modelos, entradas y salidas digitales, puertos de comunicación como PROFINET, entradas y salidas de alta velocidad y entradas analógicas incorporadas, todo dentro de una carcasa resistente que conforman al PLC. La programación de este controlador es simple para el usuario, las instrucciones como lógicas booleanas, contaje, temporización funciones matemáticas, entre otras, pueden controlar procesos que junto a la interacción de entradas y salidas físicas de sensores o actuadores pueden complementarse para el desarrollo. También existen puertos para comunicación con otros equipos, la CPU incorpora PROFINET para la comunicación en red, hay disponibles módulos adicionales para comunicación en redes PROFIBUS que pueden ser GPRS, RS232, RS485. (Siemens, 2014)

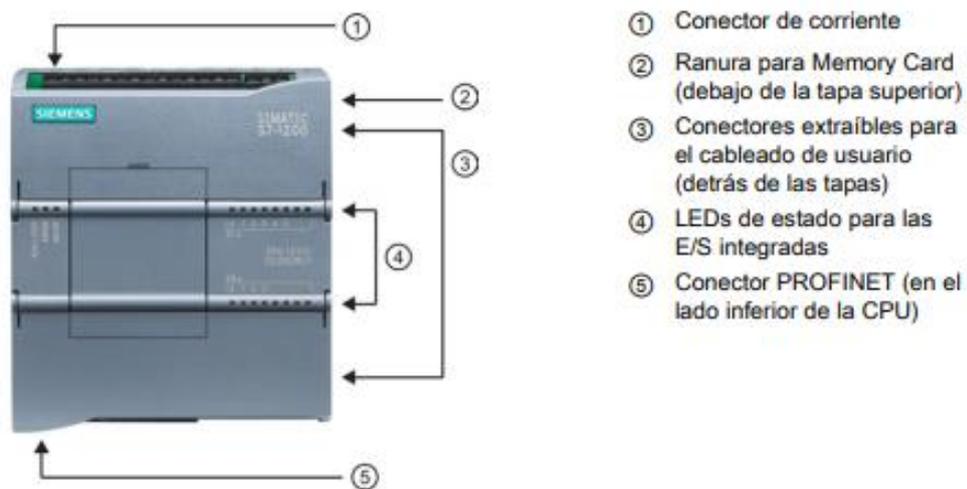


Figura 2.22: PLC Siemens S7-1200

Fuente: (Siemens, 2014)

2.14 Antena Ubiquiti Nano Station Loco M5

Las antenas Ubiquiti son dispositivos potentes para redes inalámbricas ya que cuentan con un estándar de 150 Mbps de velocidad al aire libre y hasta 5 Km de alcance dependiendo del modelo.

Estos dispositivos incorporan una antena con ganancia de 13 dBi, cruz de polaridad de factores compactos, y doble polaridad de 5 GHz con aislamiento óptimo para la comunicación. Los dispositivos Ubiquiti Nanostation tienen tecnología POE (Power Over Ethernet) que permiten la transmisión de datos y electricidad por medio del cable UTP hasta 100 metros de distancia lo que permite instalar estos equipos en lugares de difícil acceso sin la necesidad de tener una toma eléctrica cercana.

Tabla 3: Especificaciones Ubiquiti Nanostation

Fuente: (El Autor)

Procesador	Atheros MIPS 24 KC, 400 MHz
Memoria	32 MB SDRAM, 8 MB Flash
Interfase de Red	1x10 / 100 BASE-TX (Cat. 5, RJ-45) Ethernet Interfase
Peso	0.18 kg
Poder de consumo Máximo	5.5watts
Trabajo a intemperie	-30 C a 80 C
Trabajo sobre humedad	5 a 95 % de humedad
Alimentación	110-240VAC 15VDC 0.8 US
Tamaño	163x31x80



Figura 2.23: Antena Ubiquiti Loco Nano Station M5

Fuente: (El Autor)

2.15 MODBUS

MODBUS es un protocolo de comunicación industrial que fue desarrollado en el año 1979 para que equipos de automatización puedan interactuar entre sí. Originalmente solo fue implementado como un protocolo de nivel aplicación con el objetivo de

transferir datos de manera serial, actualmente este protocolo ha crecido de tal forma que en sus implementaciones se puede incluir aparte de protocolo serial, protocolos de red como TCP/IP y UDP. (National Instruments, 2019).

Según (National Instruments, 2019), MODBUS es un protocolo de solicitud-respuesta implementado usando la relación maestro y esclavo, la comunicación siempre ocurre en pares, un dispositivo inicia la solicitud y luego espera por una respuesta, y el dispositivo de inicio (Maestro) es responsable de iniciar cada diálogo. El equipo maestro por lo común suele ser una interfaz hombre máquina (HMI) o SCADA, y el esclavo puede ser un controlador PLC o sensor.

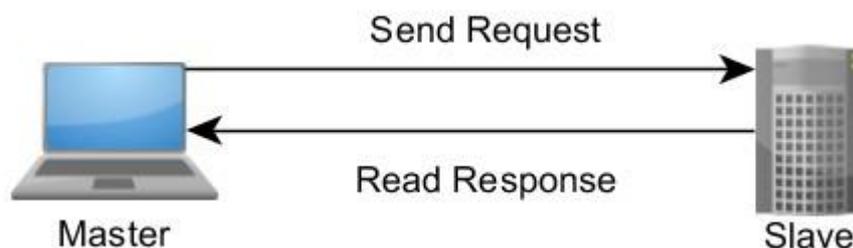


Figura 2.24: Una relación de red Maestro-Esclavo

Fuente: (National Instruments, 2019).

2.15.1 Comunicaciones Maestro/Esclavo con Modbus

Por lo general el protocolo MODBUS funciona con un maestro y uno o muchos equipos esclavos. El maestro es quien se encarga de controlar la comunicación con los esclavos en todo momento, según la especificación de este protocolo los equipos esclavos pueden ser hasta 247 conectados a la misma red. (Schneider Electric, 2008)

Cada esclavo en su configuración posee una dirección única, es por esto que el maestro sabe con quién debe hacer enlace.

2.15.2 Tipos de MODBUS

MODBUS para poder comunicarse intercambia peticiones y respuestas, los dispositivos conectados a la red MODBUS organizan los datos en tramas. MODBUS es un protocolo que se encuentra a nivel de capa aplicación entonces necesita ser utilizado sobre una pila de protocolos que resuelva que tipo de red va a ser empleada.

Dependiendo de la arquitectura de protocolo que se va a usar MODBUS se distingue en tres tipos: RTU, ASCII, y TCP/IP.

Los dispositivos controladores pueden configurarse para comunicarse en la red estándar MODBUS utilizando estos modos: RTU, ASCII y TCP/IP, para ASCII o RTU que usan puertos y parámetros de comunicación Serial (Baud rate, parity mode, etc), mientras se configura cada controlador. Los parámetros en los modos Serial deben ser los mismos para todos los dispositivos que estén en la red MODBUS, así mismo cuando se maneje MODBUS TCP/IP las direcciones IP, puerto de comunicación TCP e ID de esclavo deben estar emparejadas. (Modbus, 2006)

2.15.3 MODBUS ASCII

Cuando se configura a los dispositivos controladores para realizar una comunicación MODBUS ASCII (Código estándar americano para el intercambio de información), cada byte de 8 bits se envía como un mensaje de dos caracteres ASCII. Una de las principales ventajas del modo de comunicación ASCII es que permite que se produzcan rangos de tiempo de hasta un segundo entre envío de caracteres sin causar errores. En la siguiente tabla se ven los formatos de cada byte en modo ASCII:

Tabla 4: Formato de byte en modo ASCII

Fuente: (Modbus, 2006)

Sistema de codificación	Hexadecimal caracteres ASCII 0-9, A-F Un carácter hexadecimal contenido en cada carácter ASCII del mensaje.
Bits por Byte	1 bit de inicio. 7 bits de datos, el Bit menos significativo se envía primero. 1 bit para paridad par/impar, no hay bit para no paridad. 1 bit de parada si se usa paridad. 2 bits si no hay paridad.
Campo de verificación de errores	Verificación de redundancia cíclica (CRC)

2.15.4 MODBUS RTU

Cuando se configura a los dispositivos controladores para realizar una comunicación MODBUS RTU (Unidad Terminal Remota), cada byte de 8 bits contiene dos caracteres hexadecimales por mensaje, una de las principales ventajas de este modo es que se puede manejar una mayor densidad de caracteres que permiten un mejor desempeño de datos que el modo ASCII por la misma velocidad de baudios.

Los mensajes deben transmitirse en un flujo continuo. El formato de cada byte en modo RTU es:

Tabla 5: Formato de byte en modo RTU

Fuente: (Modbus, 2006)

Sistema de codificación	Binario de 8bits, hexadecimal 0-9, A-F Dos caracteres hexadecimales contenidos en cada campo de 8bits del mensaje.
Bits por Byte	1 bit de inicio. 8 bits de datos, el Bit menos significativo se envía primero. 1 bit para paridad par/impar, no hay bit para no paridad. 1 bit de parada si se usa paridad. 2 bits si no hay paridad.
Campo de verificación de errores	Verificación de redundancia cíclica (CRC)

2.15.5 MODBUS TCP/IP

Según (Modbus, 2006), su servicio de mensajería facilita una comunicación cliente – servidor con los dispositivos que estén conectados a la misma red Ethernet TCP/IP. Estos son los tipos de mensajes que tiene el formato cliente – servidor:

- MODBUS Request: es el mensaje enviado en la red por el Cliente para iniciar una transacción.

- MODBUS Indication: es el mensaje de solicitud recibido en el lado del servidor.
- MODBUS Response: es el mensaje de respuesta enviado por el servidor.
- MODBUS Confirmation: es el mensaje de respuesta recibido en el lado del cliente.

Estos servicios de mensajes son usados para la comunicación en tiempo real de información entre:

- Dos aplicaciones en dispositivos.
- La aplicación en dispositivo y otro dispositivo.
- Aplicaciones y dispositivos HMI/SCADA.
- Una PC y un programa de dispositivos que proporciona servicios en línea.

2.15.6 Arquitectura general del protocolo MODBUS

Los sistemas de comunicación a través de MODBUS TCP/IP pueden abarcar diferentes tipos de dispositivos:

- Un cliente MODBUS TCP/IP y dispositivos de servidor conectados a una red TCP/IP
- Los dispositivos de interconexión como puente, enrutador o puerta de enlace para interconexión entre la red TCP/IP y una subred de línea en serie que permite conexiones de dispositivos finales MODBUS serial line Client y Server. (Modbus, 2006).

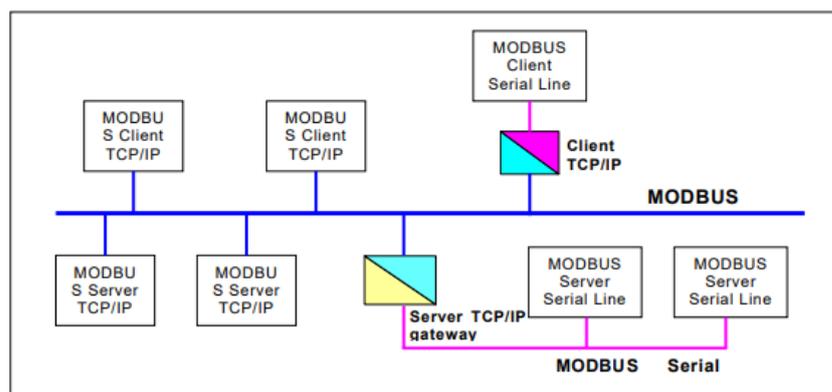


Figura 2.25: MODBUS TCP/IP commutation architecture

Fuente: (Modbus, 2006)

El protocolo MODBUS define un protocolo de unidad de datos simple (PDU) que es independiente de la capa de comunicación subyacente. El mapeo del protocolo MODBUS en específicos buses o redes puede introducir campos adicionales en la unidad de data aplicación (ADU). (Modbus, 2006)

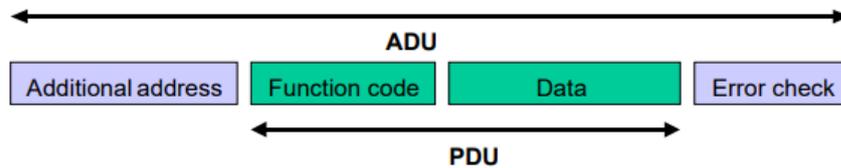


Figura 2.26: Marco general MODBUS

Fuente: (Modbus, 2006)

El cliente que inicializa una transacción de MODBUS construye un MODBUS Unidad de Data Aplicación (ADU). El código de función le dice al servidor que tipo de acción realizar.

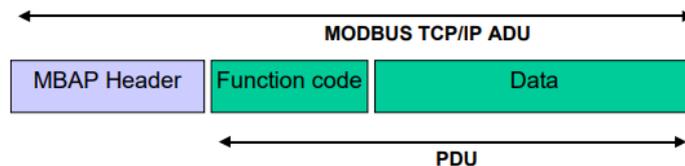


Figura 2.27: MODBUS Request/response over TCP/IP

Fuente: (Modbus, 2006)

El protocolo TCP/IP utiliza un encabezado especial para identificar el bloque de datos de la aplicación MODBUS llamado encabezado de protocolo de aplicación MODBUS (MBAP).

Este encabezado tiene algunas diferencias con los datos de la aplicación MODBUS RTU. Se debe tener en cuenta que el encabezado está estrechamente relacionado con la configuración del puerto serie.

- El campo 'Slave address' de MODBUS, es usado normalmente en la línea Serial que es cambiado por un simple byte 'Unit Identifier' de un solo byte dentro del encabezado MBAP. El Identificador ID es usado para comunicarse

a través de enrutadores, puertas de enlace, que utilizan una sola dirección IP para admitir múltiples dispositivos MODBUS Independientes.

- Todas las solicitudes y respuestas MODBUS están diseñadas para permitir que el destinatario verifique que el mensaje está completo. Para códigos de característica con una longitud fija de MODBUS PDU, un código de función basta. Para códigos de característica que contienen una cantidad variable de datos en la solicitud o respuesta, el campo de datos contiene un conteo de la cantidad de bytes.
- Cuando MODBUS se envía a través de TCP, se incluye información adicional sobre la longitud del transporte en el encabezado de MBAP para que el destinatario pueda confirmar la recepción del mensaje, incluso si el mensaje se divide en varios paquetes para su envío y transmisión. Existen reglas de longitud explícitas e implícitas, y el uso de códigos de verificación de errores CRC-32 (en Ethernet) puede provocar daños y corrupciones no detectadas en los mensajes de solicitud y respuesta.

2.15.7 MBAP Descripción de cabecera

La cabecera MBAP contiene los siguientes campos:

Tabla 6: Cabecera MBAP

Fuente: (Modbus, 2006)

Campos	Tamaño	Descripción	Cliente	Servidor
ID de transacción	2 bytes	Identificador de transacción MODBUS Solicitud/Respuesta	Inicializado por el cliente	Recopilado por el servidor desde la solicitud recibida
ID de protocolo	2 bytes	0 = Protocolo MODBUS	Inicializado por el cliente	Recopilado por el servidor desde la solicitud recibida
Longitud	2 bytes	Número de seguidores de bytes	Inicializado por el cliente (Solicitud)	Inicializado por el servidor (Respuesta)

Identificador de unidad	1 byte	Identificador de esclavo remoto conectado en línea Serial o en otros buses.	Inicializado por el cliente	Recopilado por el servidor desde la solicitud recibida.
-------------------------	--------	---	-----------------------------	---

- El identificador de transacción es utilizado para hacer coincidir transacciones, el servidor MODBUS copia la respuesta del ID de transacción de la respuesta.
- El identificador de protocolo es utilizado para multiplexación dentro de sistema. MODBUS se identifica con el valor 0.
- El identificador de unidad es un campo que se usa para enrutar dentro del sistema. Es típicamente usado para comunicación MODBUS PLUS o un esclavo MODBUS SERIAL a través de pasarela entre una red TCP MODBUS y una línea serial MODBUS. Este campo se establece por el cliente en la solicitud y devuelve el mismo valor al servidor.
- Las ADU MODBUS TCP/IP se envían por el puerto 502 registrado mediante TCP.

2.15.7 Descripción funcional de MODBUS TCP/IP

Esta arquitectura de componente de MODBUS son un modelo genérico que incluye componentes tanto de cliente como servidor y se puede utilizar en cualquier dispositivo. Es posible que algunos dispositivos solo proporcionen componentes de cliente o servidor.

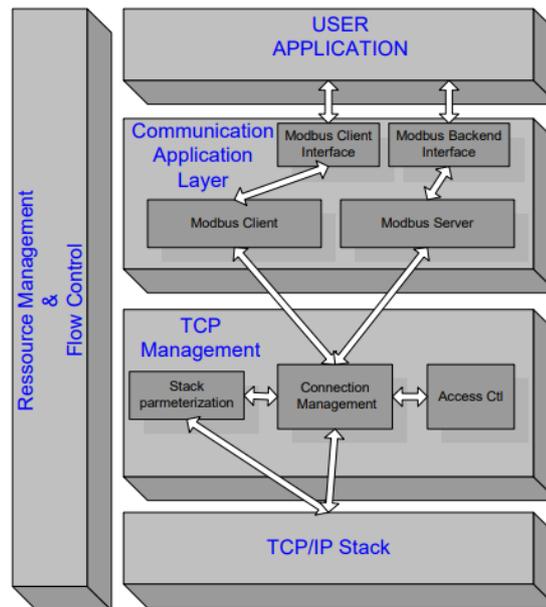


Figura 2.28 MODBUS messaging Service Conceptual Architecture.

Fuente: (Modbus, 2006)

- Comunicación Capa Aplicación

Los equipos MODBUS proporcionan una interfaz cliente – servidor. Puede proporcionar una interfaz interna que permite a los usuarios aplicar objetos de la aplicación indirectamente. Esta interfaz consta de cuatro áreas: entradas discretas, salidas de bobina discretas, registro de entradas y registros de salida.

Tabla 7: Mensajería MODBUS en la guía de implementación de TCP/IP V1.0b de MODBUS

Fuente: (Modbus, 2006)

Tablas primarias	Tipo de objeto	Tipo	Comentario
Discretos Input	Single bit	Read-Only	Este tipo de datos puede ser proporcionado por un sistema de E/S.
Coils	Single bit	Read-Write	Este tipo de datos puede ser alterado por una aplicación de programa.

Input Registers	16-bit Word	Read-Only	Este tipo de datos puede ser proporcionado por un sistema de E/S.
Holding Registers	16-bit Word	Read-Write	Este tipo de datos puede ser alterado por una aplicación de programa.

- **MODBUS Client**

El cliente MODBUS permite que la aplicación de usuario controle el intercambio de información con un dispositivo remoto. El cliente realiza una solicitud a partir de parámetros contenidos en la solicitud enviada por la interfaz de la aplicación cliente. Los clientes utilizan transacciones MODBUS y su gestión implica esperar el procesamiento de confirmación MODBUS.

- **MODBUS Server**

Cuando se recibe una solicitud MODBUS, el equipo inicia una acción propia para leer, escribir o realizar otra acción. El procesamiento de estas acciones es completamente transparente para los desarrolladores de aplicaciones. La función principal del servidor MODBUS es esperar una solicitud en el puerto TCP 502 y luego generar una respuesta de acuerdo al contexto del equipo.

- **MODBUS Backend interface**

La interfaz de Backend de MODBUS es una interfaz del Servidor MODBUS al usuario en la que se definen los objetos de la aplicación.

2.15.8 Capa de gestión TCP

Una de las principales funciones del servicio de mensajería es la gestión de la comunicación, establecimiento y terminación del control de flujo de conexiones TCP establecidas.

- **Gestión de Conexión**

El módulo de control de conexión TCP debe utilizarse para la comunicación entre el cliente MODBUS y el módulo servidor este es responsable de administrar las conexiones globales de mensajería TCP.

Hay dos posibilidades para la gestión de la conexión. Es transparente para la aplicación de usuario porque la aplicación de usuario administra la conexión TCP por sí misma o la conexión está completamente administrada por este módulo. La última solución es menos flexible.

La escucha en el puerto TCP 502 está reservada para la comunicación MODBUS. Por defecto, escuchar en este puerto es obligatorio. Sin embargo, algunos mercados o aplicaciones pueden requerir un puerto separado específicamente para MODBUS sobre TCP. Por esta razón, se recomienda encarecidamente que los clientes y servidores permitan a los usuarios establecer números de puerto MODBUS sobre TCP. Tenga en cuenta que el puerto de servidor TCP 502 debe estar disponible además del puerto específico de la aplicación, incluso si algunas aplicaciones tienen un puerto de servidor TCP diferente configurado para el servicio MODBUS.

Una comunicación entre un módulo MODBUS cliente y servidor requiere el uso de un Módulo de gestión de conexiones TCP. Se encarga de gestionar la mensajería global Conexiones TCP.

Se proponen dos posibilidades para la gestión de la conexión. O la propia aplicación de usuario gestiona las conexiones TCP o la gestión de la conexión es totalmente realizado por este módulo y, por lo tanto, es transparente para la aplicación del usuario. En el último la solución implica menos flexibilidad.

El puerto TCP de escucha 502 está reservado para comunicaciones MODBUS Es obligatorio escuchar de forma predeterminada en ese puerto. Sin embargo, algunos mercados o aplicaciones pueden requerir que otro puerto esté dedicado a MODBUS sobre TCP. Por esa razón, Es muy recomendable que los clientes y los servidores den la posibilidad al usuario para parametrizar el número de puerto MODBUS sobre TCP. Es importante tener en cuenta que incluso si otro puerto de servidor TCP está configurado para el servicio MODBUS en ciertas aplicaciones, el puerto 502 del servidor TCP debe estar disponible además de cualesquiera puertos específicos de la aplicación.

- **Módulo de control de acceso**

Si se desea que los datos internos de los dispositivos estén protegidos ante anfitriones indeseables, se puede implementar sistemas de seguridad en la comunicación si es necesario.

2.15.9 Gestión de conexiones TCP

Las comunicaciones MODBUS requieren que se establezca una conexión entre cliente y servidor bajo TCP.

Este establecimiento de conexión puede activarse por el módulo de aplicación de usuario o se puede programar para que sea automático mediante un módulo de gestión de conexiones TCP. En el primer caso un usuario debe tener conocimientos de TCP/IP para saber cómo comunicarse en el segundo escenario tiene más flexibilidad al realizarse la conexión por si sola ya que la aplicación solo envía y recibe mensajes MODBUS cuando el módulo de gestión se encarga de establecer una nueva conexión TCP cuando sea requerido.

Reglas de Implementación:

1. Se recomienda implementar la gestión automática de conexión TCP cuando un usuario lo requiera explícitamente.
2. Se recomienda mantener abierta la conexión TCP con un dispositivo remoto u no para abrirlo y cerrarlo para cada transacción de MODBUS TCP/IP, Sin embargo, el cliente MODBUS debe ser capaz de aceptar una solicitud cercana desde el servidor y cerrando la conexión. La conexión se puede volver a abrir cuando sea requerido
3. Se recomienda que un cliente MODBUS abra un mínimo de conexiones TCP con un servidor MODBUS remoto (Con la misma dirección IP). Una conexión por aplicación podría ser buena elección.
4. Se pueden activar varias transacciones MODBUS simultáneamente en la misma conexión TCP. Si se hace esto, se debe utilizar el identificador de transacción MODBUS para identificar de forma única las solicitudes y respuestas coincidentes.
5. Para la comunicación bidireccional entre dos entidades MODBUS remotos (cliente y servidor respectivamente), debe abrir conexiones separadas para el tráfico del cliente y el tráfico del servidor.
6. Una trama TCP transporta solo una ADU MODBUS. No se aconsejaría enviar demasiadas solicitudes o respuestas MODBUS a la vez en la misma PDU TCP.

2.15.9 Parametrización IP

Estos parámetros deben configurarse en la implementación IP MODBUS:

- **Dirección IP local:** Pueden ser direcciones IP de clase A, B, C.
- **Máscara de subred:** La división en subredes de una red IP se puede realizar por varias razones: diferentes medios físicos (como Ethernet, WAN, etc.), uso más eficiente de direcciones de red y la capacidad de controlar el tráfico de la red. La máscara de subred tiene que ser coherente con la clase de dirección IP de la dirección IP local.
- **Puerta de enlace predeterminada:** La dirección IP de la puerta de enlace debe estar en la misma subred de la IP local. El valor 0.0.0.0 no debe asignarse. En caso de no existir una puerta de enlace definida debe establecerse en 127.0.0.1 o en la IP local.

2.15.10 Modbus Utilizando NI OPC Servers

NI OPC Server es un programa de servidor OPC independiente rica en funciones que puede formar la columna vertebral de los sistemas SCADA. Como todos los servidores OPC antes del lanzamiento de OPC UA, el servidor NI OPC es solo para Windows, lo que lo hace más adecuado para usar como sistema de control que como sistema de control esclavo de alta velocidad. Los servidores NI OPC proporcionan un amplio conjunto de controladores que permiten la comunicación con dispositivos Modbus, así como con dispositivos que utilizan una amplia gama de protocolos específicos de proveedores o protocolos estándar como OPC UA. (ni.com, 2021)

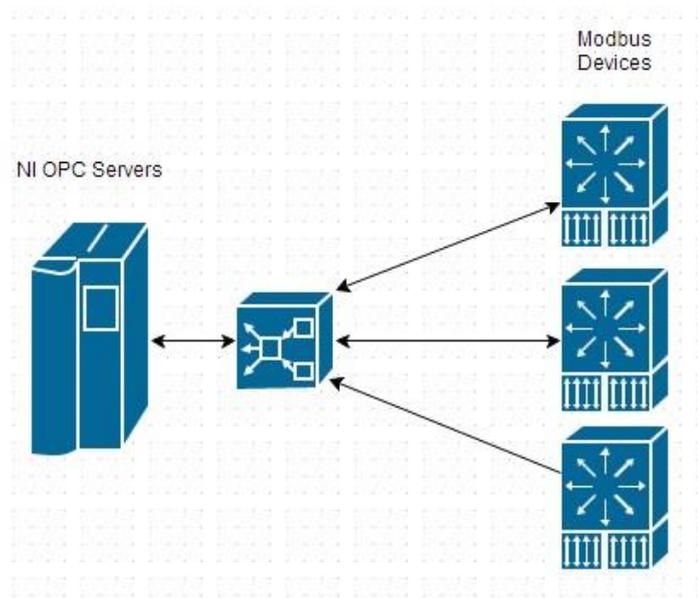


Figura 2. 29 MODBUS con NI OPC SERVERS.

Fuente: (ni.com, 2021)

2.15.11 Modbus Usando APIs de Bajo Nivel

Existen muchos controladores Modbus de bajo nivel en diferentes idiomas. El programa Labview proporciona una API Modbus de bajo nivel que complementa la funcionalidad de los servidores NI OPC y Modbus I/O. Estos controladores permiten definir explícitamente lo que sucede cuando se envía o recibe una solicitud de Modbus. Por lo general, también ofrece un mejor rendimiento que los controladores de alto nivel. A cambio, normalmente necesita escribir una gran cantidad de código de procesamiento de datos que permita que su aplicación interactúe de manera eficiente con otros dispositivos en su sistema. La funcionalidad y el comportamiento de este código depende de si el equipo es configurado como maestro o esclavo. (ni.com, 2021)

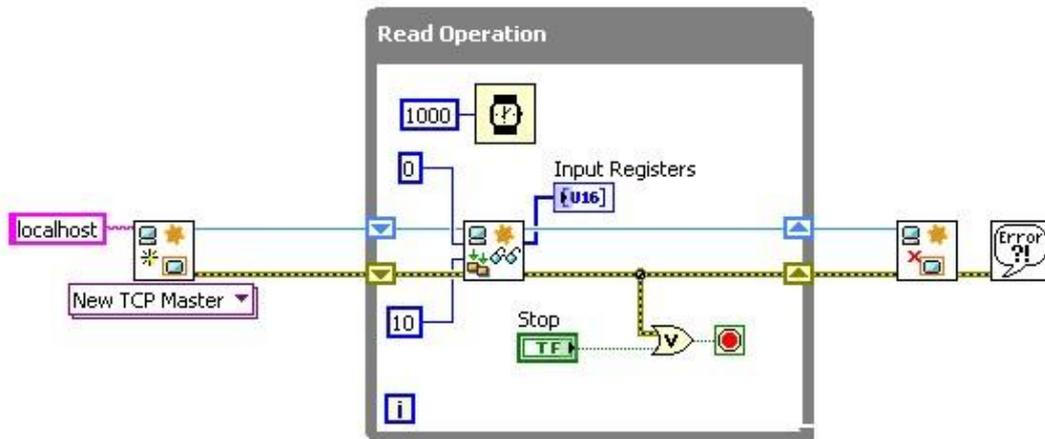


Figura 2.30 Modbus Usando APIs de Bajo Nivel

Fuente: (ni.com, 2021)

2.15.12 Ventajas y desventajas de Modbus

Ventajas

- Puede ser usado en varios proyectos.
- Su implementación es flexible.
- Es gratuito.
- Su arquitectura de modo solicitud y respuesta es a nivel de capa aplicación.

Desventajas

- Suele complicarse trabajar de modo solicitud y respuesta ya que desaprovecha la red o complica el código en la implementación.
- Suele sobrecargar la red.
- El tamaño de paquete de envío es limitado, es por ello que disminuye la velocidad de transferencia máxima en cada consulta.
- La transferencia de datos es imposible cuando se trata de ser continua o de tipo streaming.
- No es seguro al no implementar aplicaciones de seguridad.
- Su funcionamiento es de tipo consulta o polling. (ni.com, 2021)

3. CAPITULO 3: MARCO METODOLÓGICO

3.1 Descripción de módulos didácticos

Los elementos que se conectaran vía radio enlace se ilustran en la Figura 3.1 Y se detalla brevemente:

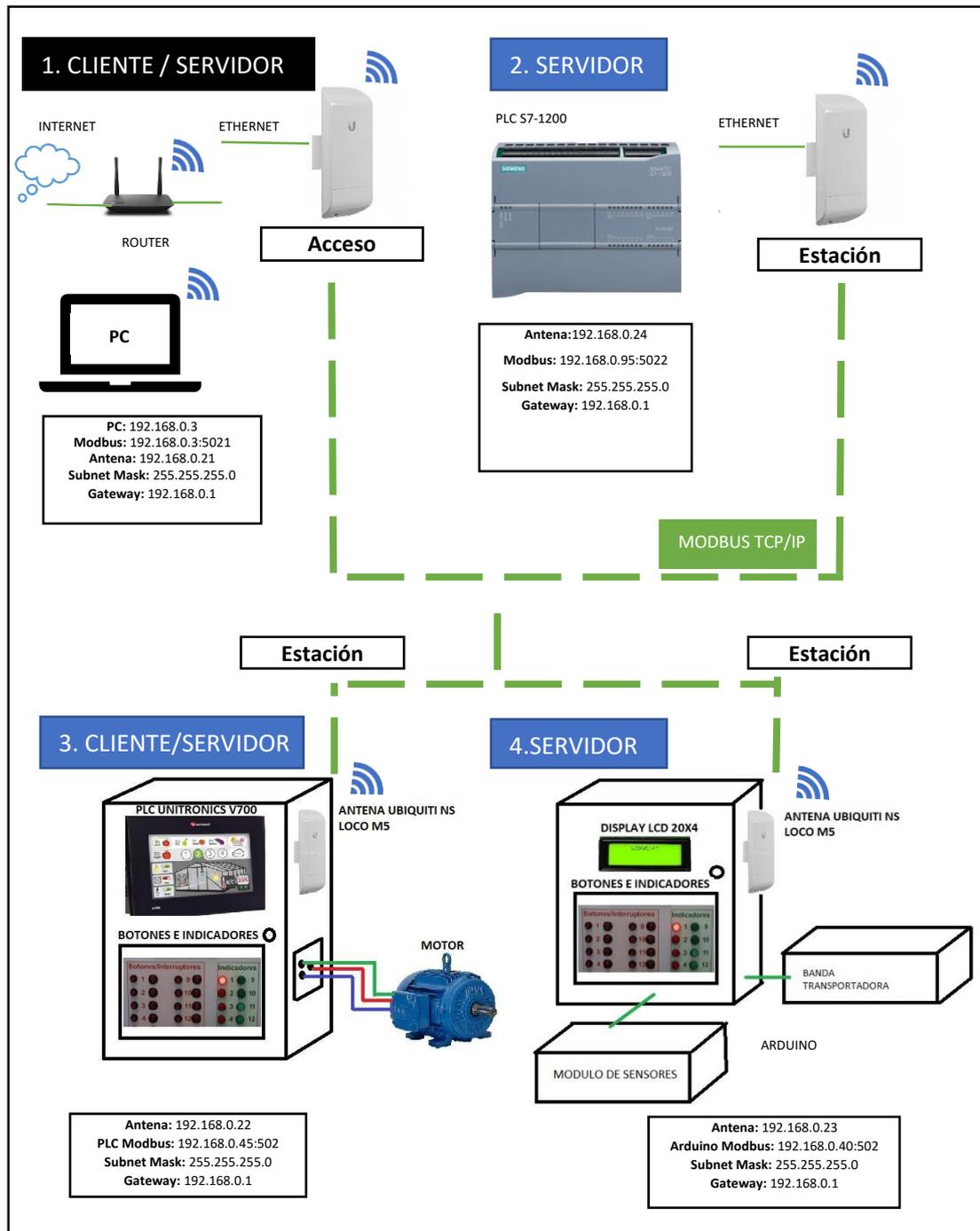


Figura 31: Esquema de proyecto

Fuente: El autor

A continuación, se indica los elementos de interés a usar en cada una de las estaciones y las direcciones IP a configurar en cada una la siguiente tabla se detalla:

Tabla 8. Elementos en las estaciones y direcciones IP

Fuente: El autor

ESTACIONES	ELEMENTOS A USAR	DIRECCIONES IP
Estación 1: PC	Antena Router Labview Wincc de Siemens VisiLogic Arduino IDE Matlab	LAN IP Router: 192.168.0.1 PC: 192.168.0.3 Antena de Acceso: 192.168.0.21 Modbus Slave/Master: 192.168.0.3:5021 Subnet Mask: 255.255.255.0 Gateway: 192.168.0.1
Estación 2: PLC Siemens	Antena, PLCS7- 1200	Antena: 192.168.0.24 Modbus Slave: 192.168.0.40:502 Subnet Mask: 255.255.255.0 Gateway: 192.168.0.1
Estación 3: PLC Unitronics	Antena Indicadores Pulsadores y Selectores Variador de frecuencia Motor trifásico	Antena: 192.168.0.22 PLC Modbus Slave/Master: 192.168.0.45:502 Subnet Mask: 255.255.255.0 Gateway: 192.168.0.1
Estación 4: Arduino	Antena Indicadores Pulsadores y Selectores Sensores Banda transportadora Módulo de relés	Antena: 192.168.0.23 Arduino Modbus Slave: 192.168.0.40:502 Subnet Mask: 255.255.255.0 Gateway: 192.168.0.1

3.2 Armarios Termoplásticos

Se eligió dos armarios termoplásticos Magna de la marca Famatel ya que su instalación en ambientes exteriores tales como piscinas camaroneras, campos de cultivos, o bodegas, da una referencia de la durabilidad y su bajo costo, que como característica tiene que; sus marcos y puertas son de material de plástico ABS muy resistente al impacto, grado IP65 de protección contra polvo y agua, protección UV y contra golpes IK08, incluye placa metálica galvanizada, apertura con llave de ¼ de giro. Las dimensiones internas son: Altura (50 cm), ancho(40cm), profundidad (17,5cm).



Figura 32: Armario Industrial Famatel

Fuente: (El autor)

3.3 Medidas y perforación en armarios industriales

Se procedió a realizar las medidas de Luces indicadoras, selectores, pantalla LCD, pantalla del PLC Vision 700 para poder realizar las perforaciones en las puertas de los armarios, como se observa en la figura 3.3:



Figura 33: Perforación de puerta para elementos

Fuente: (El autor)

3.4 Montaje de equipos en puertas de los armarios

Como segundo paso al montaje, se procedió a colocar las canaletas pegadas con cemento de contacto en las puertas de los armarios, la colocación de luces indicadoras y selectores, la colocación de pantalla LCD (para módulo Arduino) y pantalla HMI+PLC (para módulo Unitronics), para luego cablear cada elemento y dejar con marquillas para cable al final de cada uno, a la espera de colocar la placa metálica en el interior de los armarios.

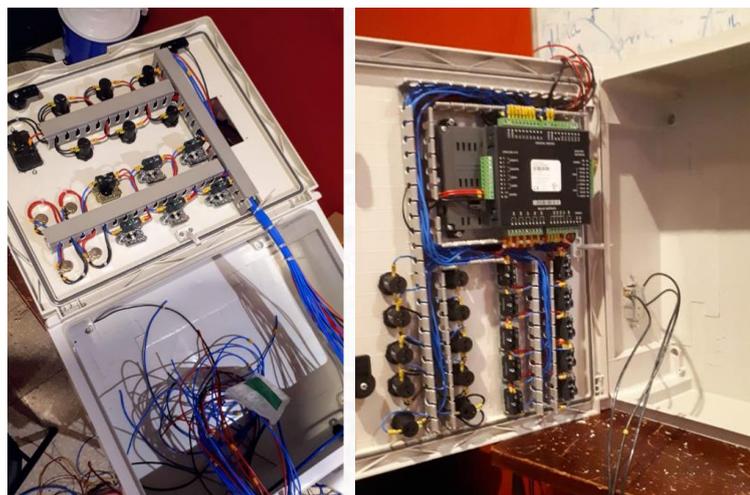


Figura 34 Cableado de elementos en puerta de armarios

Fuente: (El autor)

3.5 Montaje de equipos en placa metálica

Se estableció un orden para la colocación de equipos, como se observa en la figura 3.5 En su parte superior; los equipos de protección y alimentación de voltaje, y en la parte inferior; los actuadores y equipo de control, esto para el módulo Arduino. Para el cableado se colocó marquillas con una numeración que identifica cada tramo de conexión, así en un futuro se pueda realizar un levantamiento de planos y circuitos.

Para el módulo Arduino junto a el módulo de relés, se diseñó una pequeña placa de resistencias de pull-up, la cual recibirá la señal de pulsadores y selectores para evitar falsos estados al conectarlos a entradas digitales del Arduino.

Adicional podemos ver un adaptador de terminal de tornillos para montaje en riel din que se une a Arduino y tener una conexión más unificada en el proyecto.



Figura 35: Cableado de tablero metálico

Fuente: (El autor)

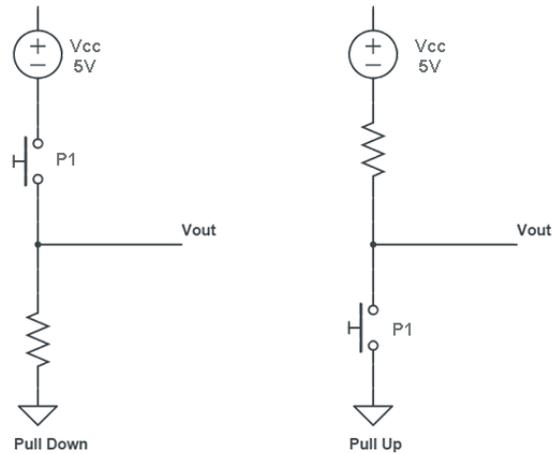


Figura 36: Resistencia de Pull Up/Down

Fuente: (El autor)

Para el módulo de PLC Unitronics se colocó en el tablero metálico, la distribución y protección de voltaje dividida, para potencia y variador de frecuencia en la parte superior; cómo se puede ver en la figura. y voltaje de control en la parte inferior, esto evitara que el operador pueda quedar expuesto a choques eléctricos. También podemos observar tanto para la figura 3.5 y figura 3.7 la colocación del dispositivo POE junto a un enchufe de 120v.



Figura 37: Cableado de tablero metálico Unitronics

Fuente: (El autor)

Se repitió el mismo orden de cableado y numeración en este módulo para poder identificar cada tramo de conexión.

3.6 Montaje y cableado de placas metálicas en el armario

Se colocó la placa metálica en el interior de los armarios en cada módulo, estos vienen con sus tornillos para ajustar en el interior, luego, se cableo los elementos de la puerta con los elementos del interior el módulo como corresponde.

Las antenas fueron colocadas en la parte derecha de cada tablero y ajustadas con amarras.

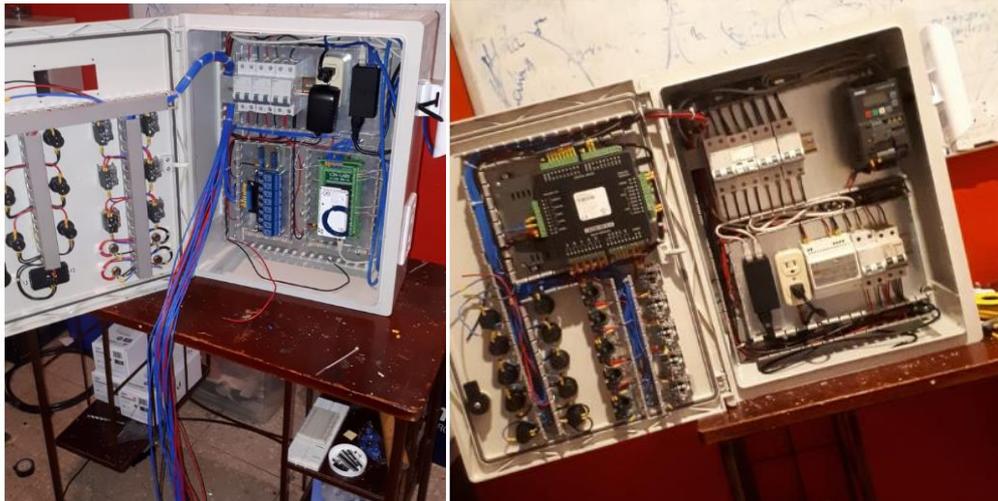


Figura 38: Cableado y unión de puertas con tableros metálicos

Fuente: (El autor)

Para el módulo de Arduino se dejó dos puertos RJ-45 como se observa en la parte inferior de la figura 3.9 que serán de uso exclusivo de conexión de entradas y salidas digitales que serán conectados hacia el circuito de sensores y banda transportadora. Para ello se usó cableado con cable UTP desde la placa Arduino y los conectores RJ-45 colocados en el interior del armario.



Figura 39: Instalación de antena y puertos de red para salidas en módulo Arduino

Fuente: (El autor)

3.7 Armarios terminados para realizar prueba

En la figura 3.10, se observan los armarios terminados y colocadas las tapas de canaletas para realizar pruebas y verificar fallas en el cableado.



Figura 40: Armarios terminados por dentro

Fuente: (El autor)

3.8 Pruebas de conexión de los armarios

Se cargó programas ejemplos para verificar el funcionamiento de selectores e indicadores, y se verifico en el módulo Arduino el funcionamiento del módulo de relés.



Figura 41: Funcionamiento de módulos

Fuente: (El autor)

3.9 Configuración de Antenas Ubiquiti

En la siguiente imagen se muestra el contenido de las antenas Ubiquiti:



Figura 42: Antena Ubiquiti Loco M5

Fuente: (El autor)

3.9.1 Pasos de conexión antena:

1. Conectar un cable Ethernet de la NanoStation hacia el puerto POE del Adaptador POE, quien suministrara 24V a nuestra antena.
2. Conectar un cable de Ethernet desde un puerto LAN de la PC al puerto LAN del Adaptador POE.

3. Conectar el cable de energía hacia un tomacorriente.



Figura 43: Alimentación de voltaje Antenas

Fuente: (El autor)

4. Configurar el adaptador Ethernet de la PC en el sistema host con una dirección IP estática en la subred 192.168.1.x, en este caso será la dirección 192.168.1.2

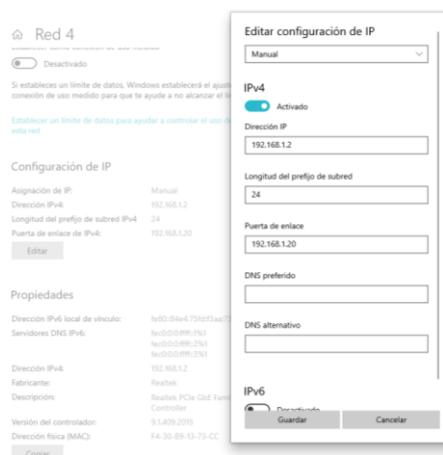


Figura 44: Configuración IP en PC

Fuente: (El autor)

3. En el navegador web, escribir <https://192.168.1.20> y luego presionar enter.

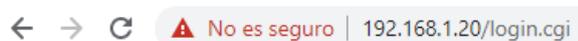


Figura 45: Ingreso por Web a antenas.

Fuente: (El autor)

4. La página de configuración aparecerá y se debe llenar los datos, el usuario y contraseña, estos serán: “ubnt”.



Figura 46: Login a web de Antenas

Fuente: (El autor)

3.9.2 Configuración de antena de Acceso

Esta configuración servirá para que otros dispositivos puedan acceder a la red de MODBUS TCP/IP que se diseñará más adelante. Para configurar el acceso se detalló los siguientes pasos:

1. En la pestaña “WIRELESS” configuramos el modo inalámbrico como punto de acceso y el nombre de la red SSID será Acceso, damos click en cambiar. Se abrirá una ventana de cambio de contraseña, que fue cambiada a 123456789, aceptamos y luego dar click en aplicar.

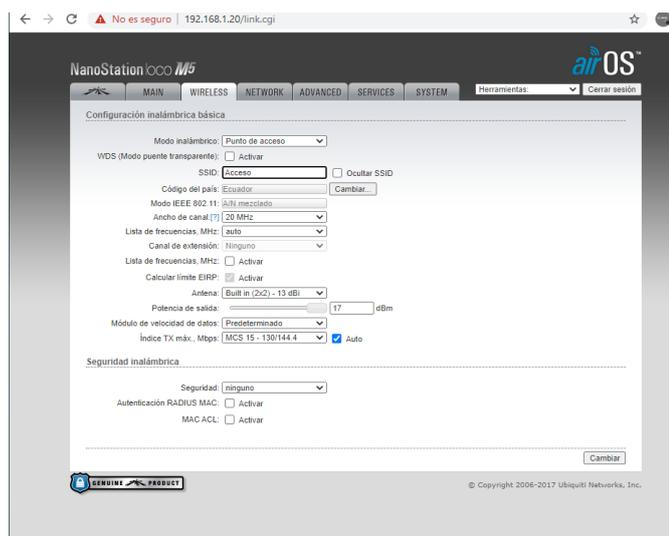


Figura 47: Configuración Wireless de Antenas

Fuente: (El autor)

2. Luego en la primera pestaña se desactivará el recuadro de airMax que no permitía que dispositivos diferentes a la marca Ubiquiti rastreen la red:



Figura 48: Desactivación de airMax.

Fuente: (El autor)

3. En la pestaña “NETWORK” dejaremos los parámetros igual recordando que el modo de máscara quedará en Puente y la dirección IP de configuración de la antena será 192.168.1.20, para acceso, luego la red “Acceso” podrá ser visible desde cualquier dispositivo.



Figura 49: Configuración de pestaña Network.

Fuente: (El autor)

4. Desconectar la antena “Acceso” de la PC y luego conectarla a un Router con acceso a internet, este paso será opcional y si se requiere estar conectado al punto de acceso y tener internet.



Figura 50: Conexión entre antena con router con acceso a internet.

Fuente: (El autor)

3.9.3 Configuración de antenas de Estaciones.

Para la configuración de las antenas donde se conectarán los dispositivos, Raspberry Pi, Unitronics y Arduino, se debe realizar los pasos del apartado 3.9.1 y luego seguir los siguientes pasos para las tres antenas de las estaciones.

1. Se configurarán en la pestaña “NETWORK” la dirección IP estática los siguientes valores:

- Para PLC Siemens: 192.168.0.24 / 255.255.255.0/ 192.168.0.1
- Para PLC Unitronics: 192.168.0.22 / 255.255.255.0/ 192.168.0.1
- Para Arduino: 192.168.0.23 / 255.255.255.0/ 192.168.0.1
- Para Acceso (PC): 192.168.0.21 / 255.255.255.0/192.168.0.1

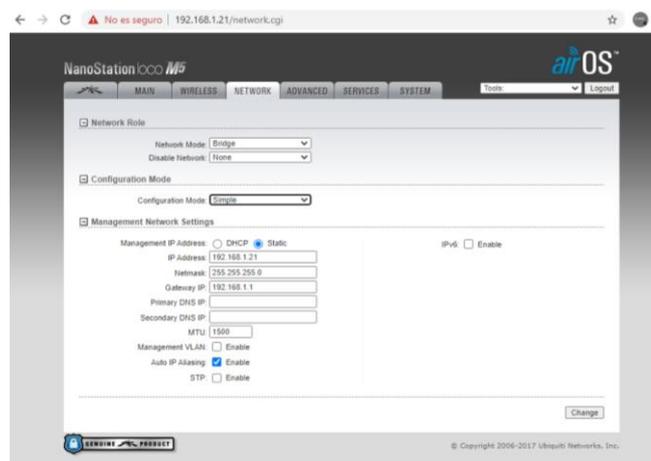


Figura 51: Configuración de IP estática.

Fuente: (El autor)

2. En la pestaña “WIRELESS” se configuró las 3 antenas el modo inalámbrico en modo “Estación”, y luego se selecciona la SSID dando click en Seleccionar, se abrirá una pantalla donde aparecerán las redes Wifi, y se escoge la red ya configurada como punto de acceso llamado “Acceso”, los demás valores quedan por defecto.

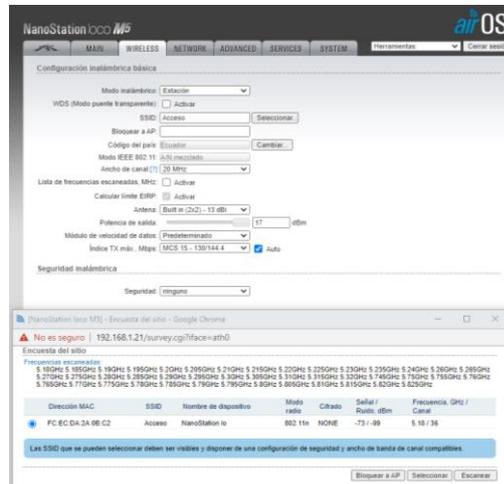


Figura 52: Configuración de modo de antena.

Fuente: (El autor)

3. Se verifica el estado de conexión y demás parámetros desde la pestaña “MAIN”



Figura 53: Verificación de estado de conexión.

Fuente: (El autor)

4. Físicamente podremos observar en las antenas el panel de LEDs, que indican lo siguiente:



Figura 54: Panel de LEDs.

Fuente: (El autor)

Desde la izquierda a derecha los LEDs de señal:

- LED de encendido
- LED de Ethernet principal LAN1, parpadea cuando hay actividad
- LED LAN2 (Para este modelo no está disponible)
- LED rojo: cuando la señal de la red inalámbrica está por encima de -94dBm
- LED naranja: cuando la señal de la red inalámbrica está por encima de -80dBm
- LED verde: cuando la señal de la red inalámbrica está por encima de -73dBm y -65 dBm

Una vez configuradas las antenas, Físicamente podremos observar el panel de leds como en la siguiente figura una vez que estas se enlacen a la red “Acceso” al momento de encenderse.



Figura 55: Panel de leds con red activa.

Fuente: (El autor)

5. (Opcional). Se puede descargar de la página oficial de Ubiquiti, el software Ubiquiti Discovery, para poder escanear las antenas conectadas a la red de acceso como se muestra en la siguiente figura:

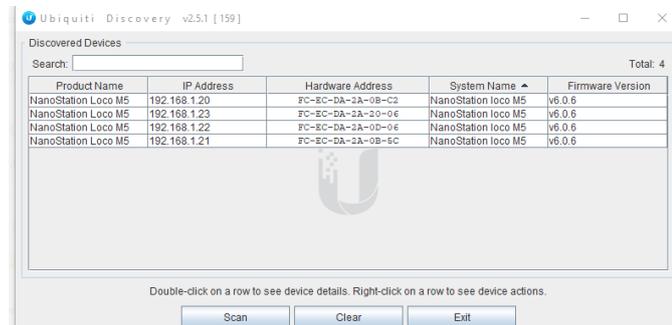


Figura 56: Escaneo de antenas conectadas a la red de acceso.

Fuente: (El autor)

3.9 Configuración y carga de librería MODBUS TCP/IP en software Arduino

Para poder usar MODBUS TCP/IP en Arduino se debe descargar la librería de Internet del siguiente enlace: <http://myarduinoprojects.com/MODBUS.html>.

Esta librería le da al programa la posibilidad de ser un maestro, un esclavo o ambos en una red TCP. El puerto 502 como se había mencionado antes es un puerto Estándar de este protocolo.

Al momento de descargar el archivo .ZIP, podremos observar 2 archivos, MgsMODBUS.cpp, y MgsMODBUS.h. Abriremos el programa Arduino IDE, programa, Incluir Librería, Añadir Biblioteca.ZIP. Se abrirá una ventana donde seleccionaremos el archivo MgsModbus-v0.1.1.zip

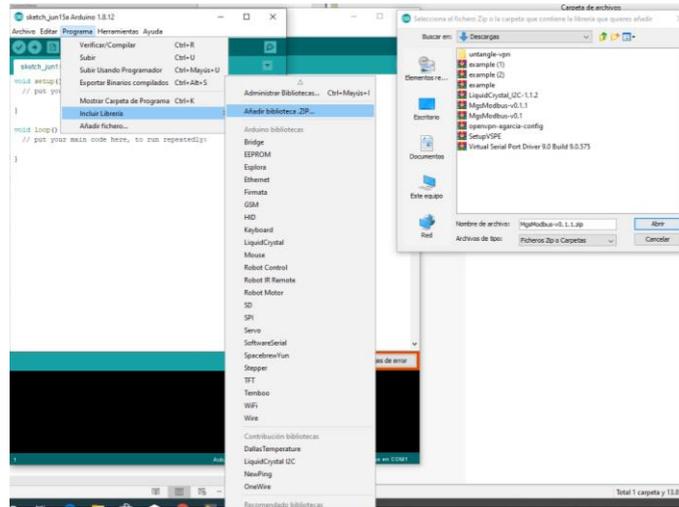


Figura 57: Configuración y carga de librería MODBUS TCP/IP en software Arduino

Fuente: (El autor)

Se debe descargar y realizar este mismo paso en caso de no tener las librerías siguientes necesarias para el proyecto: SPI.h, Ethernet.h.

3.10 Configuración del software VisiLogic

El programa VisiLogic se puede descargar de la página oficial de Unitronics. Una vez abierto el programa debemos seleccionar la versión de PLC, para este proyecto; el PLC Vision700 (V700) como se muestra en la figura:

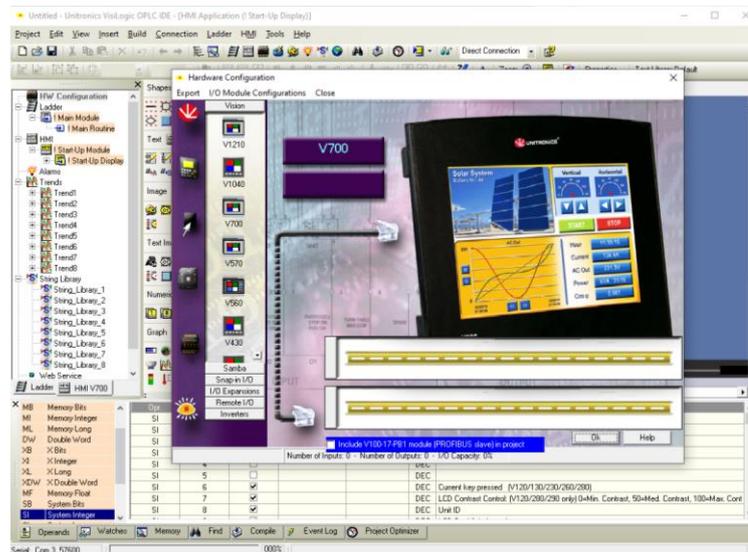


Figura 58: Configuración del software VisiLogic

Fuente: (El autor)

Luego se seleccionó el modelo del módulo de entradas y salidas; para este proyecto será: V200-18-E2B.

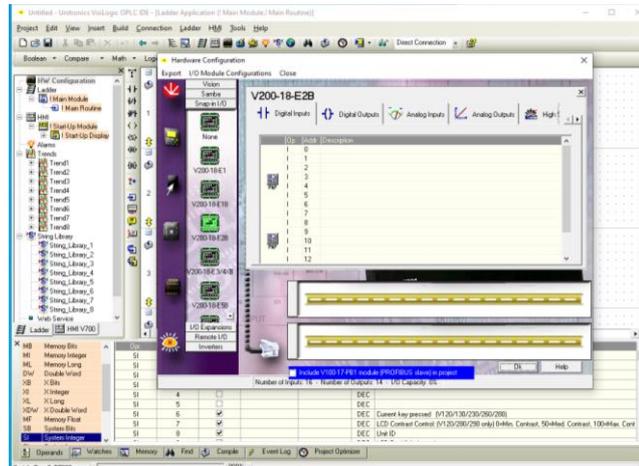


Figura 59: Selección del módulo de entradas y salidas.

Fuente: (El autor)

En esta sección de la pantalla se podrá dar descripción a las entradas y salidas que se usaran en el PLC, en la imagen se muestra cómo se debe seleccionar y configurar una entrada analógica, seleccionando el tipo de dato (MI), la dirección (100 y 101), tipo de entrada analógica (0-10V) y descripción (“A0” y “A1”), así mismo se podrá configurar salidas analógicas, se muestra en la imagen:

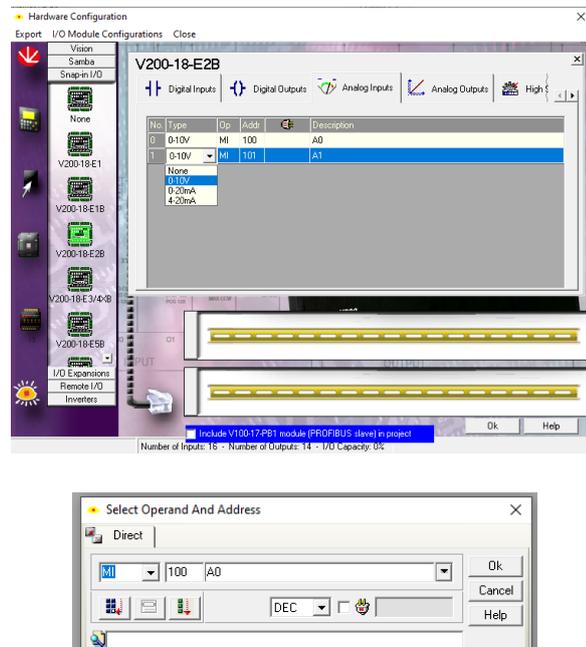


Figura 60: Descripción de entradas y salidas.

Fuente: (El autor)

3.9 Configuración de PLC Siemens.

Se debe crear un proyecto con las siguientes características del dispositivo: PLC S7-1200 en TIA Portal, también se puede usar otro tipo de CPU siempre y cuando se configure correctamente los bloques de Modbus TCP/IP para armar la red de comunicación entre Controladores.

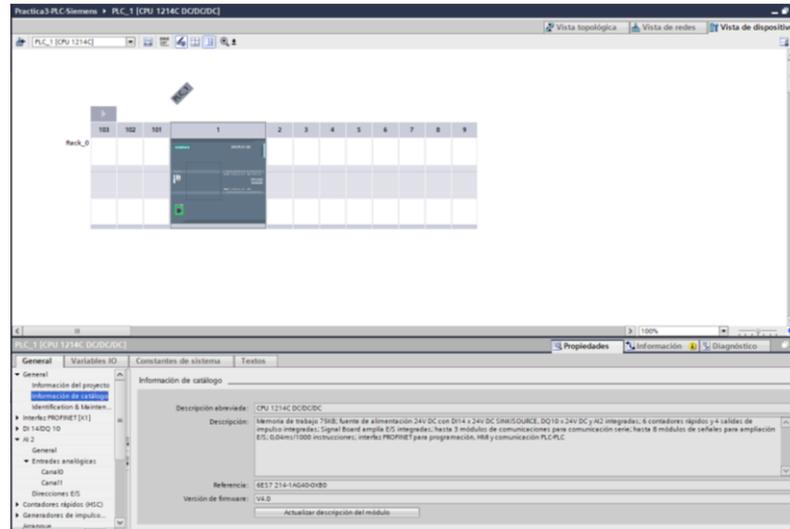


Figura 61: Configuración de PLC Siemens.

Fuente: (El autor)

Se usará el bloque de función Modbus TCP/IP en modo servidor: MB_SERVER

La instrucción MB_SERVER es de tipo servidor Modbus TCP y trabaja físicamente a través de una conexión PROFINET. Este bloque de instrucción procesa solicitudes a un cliente Modbus TCP, recibe y procesa la información y envía mensajes de respuesta. Nota: Recordar que el funcionamiento de los bloques de comunicación puede variar entre versiones de CPU del PLC de Siemens a utilizar, pero el principio sigue siendo el mismo.

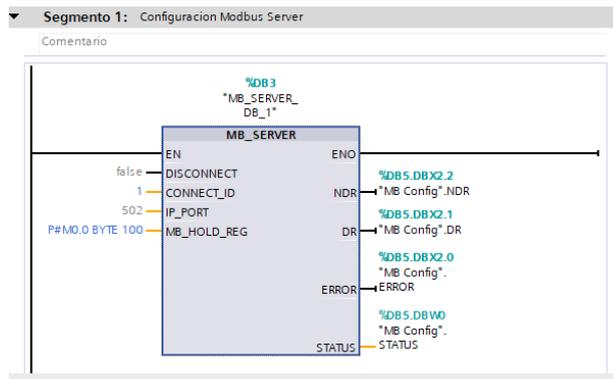


Figura 62: instrucción "MB_SERVER"

Fuente: (El autor)

La siguiente tabla muestra los parámetros de la instrucción "MB_SERVER":

Tabla 9: Parámetros de instrucción "MB_SERVER"

Fuente: (Siemens, 2014)

Parámetro	Declaración	Tipo de datos	Descripción
DISCONNECT	Input	BOOL	<p>La instrucción "MB_SERVER" establece una conexión pasiva con un módulo interlocutor. El servidor reacciona a una solicitud de conexión de la dirección IP indicada en el SDT "TCON_IP_v4" en el parámetro CONNECT.</p> <p>Este parámetro permite controlar cuándo se aceptará una solicitud de conexión:</p> <ul style="list-style-type: none"> • 0: Si no hay ninguna conexión establecida, se establece una conexión pasiva. • 1: Inicialización del establecimiento de la conexión. Si la entrada esta activada, no se ejecutan otras operaciones. Tras deshacer la conexión correctamente, el parámetro STATUS devuelve el valor 0003.
MB_HOLD_REG	InOut	VARIANT	Puntero al registro de retención Modbus de la instrucción "MB_SERVER"

Parámetro	Declaración	Tipo de datos	Descripción
			<p>MB_HOLD_REG debe remitir siempre a un área de memoria mayor de dos bytes.</p> <p>El registro de retención engloba los valores a los que está permitido acceder en un cliente Modbus 3 (lectura), 6 (escritura), 16 (escritura múltiple) y 23 (escritura y lectura en un orden).</p> <p>Utilice como registro de retención un bloque de datos global de acceso optimizado o el área de memoria de marcas.</p>
CONNECT	InOut	VARIANT	<p>Contiene la estructura de la descripción para realizar una conexión.</p> <p>Se pueden utilizar las siguientes estructuras (SDT):</p> <ul style="list-style-type: none"> • TCON_IP_v4: contiene todas las características necesarias de direccionamiento para realizar una conexión programada. La forma estándar es 0.0.0.0 (cualquier dirección IP), también puede indicar una dirección IP determinada de forma que el servidor solo interactúe a las peticiones de aquella dirección. Si se utiliza TCON_IP_v4, la conexión se realiza al llamar la instrucción "MB_SERVER". • TCON_Configured (solo con S7-1500): contiene los parámetros de dirección de una conexión configurada. En caso de usar TCON_Configured, la conexión se establece cuando la CPU haya cargado la configuración hardware.
NDR	Output	BOOL	<p>"New Data Ready":</p> <ul style="list-style-type: none"> • 0: No existen datos nuevos • 1: Se produce cuando el cliente a enviado datos a la instrucción.
DR	Output	BOOL	<p>"Data Read":</p> <ul style="list-style-type: none"> • 0: No se han leído datos • 1: El cliente Modbus ha leído datos

Parámetro	Declaración	Tipo de datos	Descripción
ERROR	Output	BOOL	Cuando ocurre un error en el bloque de instrucción MB_SERVER, esta salida de error cambia de estado a verdadero, si se quiere más detalles del error lo indica el parámetro STATUS.
STATUS	Output	WORD	Contiene en forma detallada la información que proporciona el bloque de instrucción.

3.10 Conexión de sensores y banda transportadora

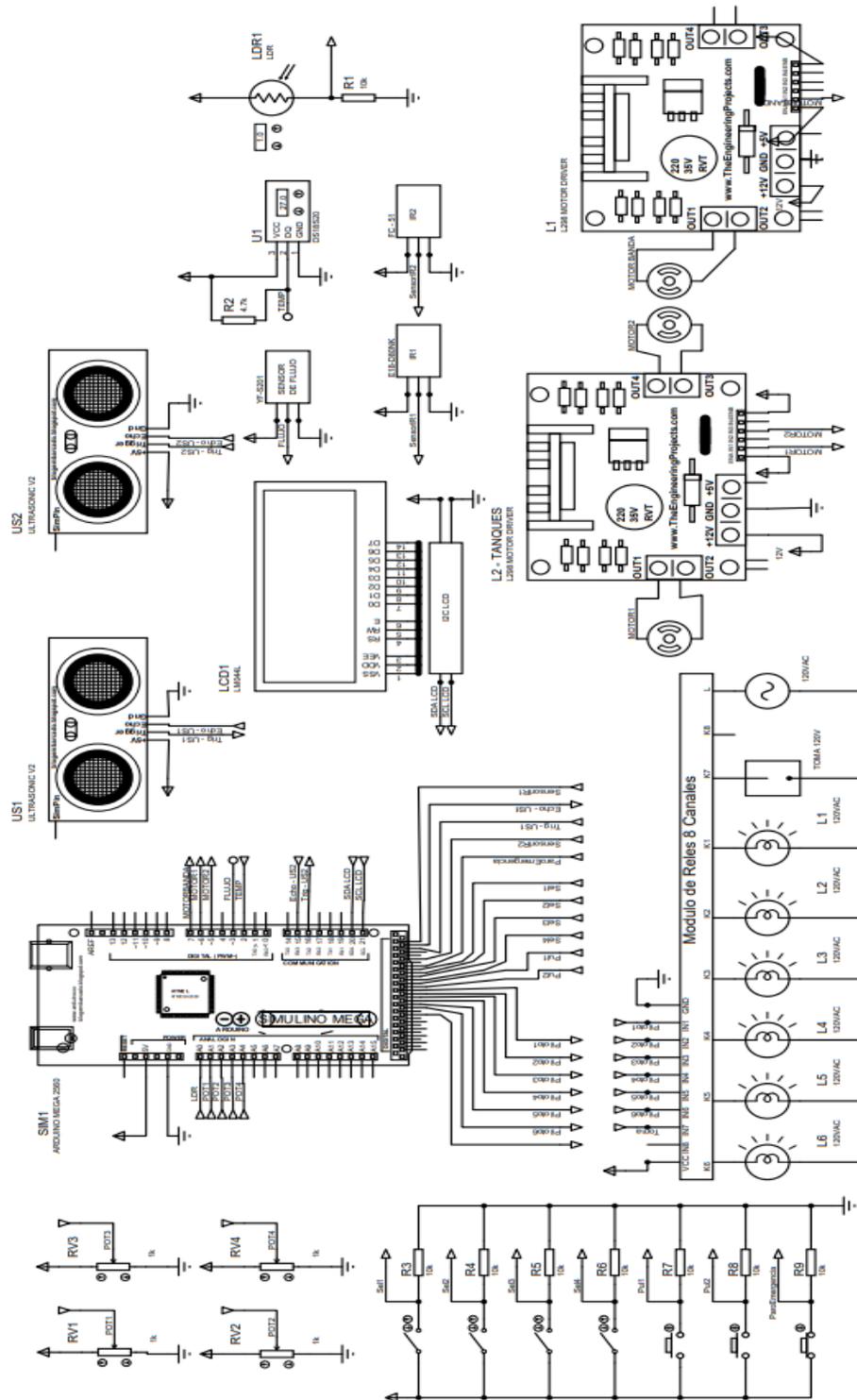


Figura 63 Conexión de sensores y banda transportadora.

Fuente: El autor

4. Guía de prácticas de laboratorio

4.1 Práctica 1

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados	
NRO. PRÁCTICA:	01	TÍTULO PRÁCTICA: Comunicación MODBUS TCP/IP entre Arduino Ethernet y SCADA Siemens para lectura de sensores	
OBJETIVOS: <ol style="list-style-type: none">1. Implementar una red MODBUS TCP/IP en Arduino.2. Diseñar una red MODBUS TCP/IP entre Arduino y Programa Wincc.3. Ayudar al estudiante a comprender el entorno Wincc.4. Realizar un aplicativo básico de MODBUS TCP/IP.			
DEV:	1. Introducción: <p>El programa desarrolla una comunicación simple entre un maestro y un esclavo (Cliente y Servidor), el módulo Arduino esclavo mandará datos de lectura de sensores y actuadores conectados a la maqueta de banda transportadora y la PC con Siemens WinCC leerá los sensores y podrá activar de forma manual alguno de los actuadores conectados.</p>		
	2. Programación y configuración de módulo esclavo Arduino <ul style="list-style-type: none">• Incluir librerías necesarias en el proyecto: <pre>Arduino_Practica1 MgsModbus.cpp MgsModbus.h 1 //UNIVERSIDAD POLITECNICA SALESIANA 2 //TESIS DE GRADO DE INGENIERIA ELECTRONICA 3 //AUTOR: NICOLAS CRESPO DELGRADO 4 //TEMA: PRACTICA 1 COMUNICACION MODBUS TCP/IP ENTRE ARDUINO 5 // ETHERNET Y SCADA SIEMENS PARA LECTURA DE SENSORES. 6 7 #include "MgsModbus.h" //Libreria Modbus 8 //Libreria para Modulo Ethernet 9 #include <Ethernet.h> 10 #include <SPI.h> 11 12 //Libreria y pin de conexion para sensor de temperatura 13 #include <OneWire.h> 14 #include <DallasTemperature.h> 15 #define ONE_WIRE_BUS 2 16 17 //Libreria y pines de conexion para sensor ultrasonico 18 #include <NewPing.h> 19 #define TRIGGER_PIN 24 20 #define ECHO_PIN 23 21 #define MAX_DISTANCE 200 22 23 // Declaracion de variables para sensores 24 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); 25 OneWire oneWire(ONE_WIRE_BUS); 26 DallasTemperature sensors(&oneWire); 27 28 //Configuracion direcciones de comunicacion TCP Ethernet 29 MgsModbus Mb; 30 byte mac[] = {0xFE, 0x62, 0xC7, 0xDC, 0xE8, 0xC4}; 31 IPAddress ip(192, 168, 0, 40); 32 IPAddress gateway(192, 168, 1, 1); 33 IPAddress subnet(255, 255, 255, 0); 34</pre> <p>Como se puede observar, se declara la variable para MODBUS Mb, además se declara la dirección mac para identificar a Arduino como dispositivo de red y configuramos la dirección IP en la que se encontrará el dispositivo MODBUS, para este caso 192.168.0.40 como se había mencionado en el capítulo 3.</p>		

- **Declaración Variables de entradas y salidas**

Para esta práctica se configurarán para lectura:

Sensores:

- Sensor de temperatura DS1820
- Sensor Infrarrojo FC-51
- Sensor Ultrasónico HC-SR04
- Sensor infrarrojo e18-d80nk
- Sensor fotorresistencia
- Selectores

Actuadores:

- Motor banda 5v/12v
- Salidas de alarma
- Luces piloto de puerta

```
Arduino_Practica1 MgsModbus.cpp MgsModbus.h

//Declaracion de variables y pines de conexion de Arduino
int sensorIR1 = 22;
int IR1;
int sensorIR2 = 27;
int IR2;
int Frecuencia;
int Selector1 = 30;
int Selector2 = 31;
int Sel1;
int Sel2;

int Piloto1 = 36;
int Piloto2 = 37;

int PhotoRes;
int Potenciometro1;
int Potenciometro2;

int motor =7;
```

- **Configuración de red e inicialización de protocolos**

A continuación, se configura los pines de uso para la conexión cliente-servidor a la red local. Además, se inician los protocolos para el funcionamiento del sensor de temperatura.

```
Arduino_Practica1 MgsModbus.cpp MgsModbus.h

void setup() {
  //Configuraciones de modos de uso de pines de conexion
  //Configuracion de puerto serie e inicializacion de protocolos
  Serial.begin(9600);
  Serial.println("Serial interface started");
  Ethernet.begin(mac, ip, gateway, subnet);
  Serial.println("Ethernet interface started");

  sensors.begin();

  pinMode(Selector1 , INPUT);
  pinMode(Selector2 , INPUT);

  pinMode(Piloto1 , OUTPUT);
  pinMode(Piloto2 , OUTPUT);
  pinMode(motor , OUTPUT);
}
```

- **Adquisición y envío de datos**

Se realiza en primer lugar la toma de datos de los diferentes sensores: temperatura, a la dirección M0 de Modbus; ultrasonido, conversión a centímetros y envío hacia

las direcciones de Modbus M4. Similarmente, la lectura de señal analógica tal que: fotorresistencia, lectura analógica y envía hacia dirección Modbus M3; potenciómetro, lectura analógica y envío de dato hacia dirección M6 Modbus. Finalmente, se realiza la lectura digital y la asignación de variables a los datos de los selectores y los infrarrojos.

```
Arduino_Practica1 MgsModbus.cpp MgsModbus.h  
  
void loop() {  
  //Adquisicion de dato de sensor de temperatura  
  //Envio de dato direccion M0 de Modbus  
  sensors.requestTemperatures();  
  Serial.print("Temperature is: ");  
  Serial.println(sensors.getTempCByIndex(0));  
  Mb.MbData[0] = sensors.getTempCByIndex(0);  
  
  //Adquisicion de dato de sensor ultrasonico  
  //Conversion de dato leido a centimetros  
  //Envio de dato direccion M4 de Modbus  
  unsigned int uS = sonar.ping();  
  Serial.print("Ping: ");  
  Serial.print(uS / US_ROUNDTRIP_CM);  
  Serial.println("cm");  
  Mb.MbData[4]= uS / US_ROUNDTRIP_CM;  
  
  //Lectura analogica y envio de dato a dir M3  
  PhotoRes = analogRead(A0);  
  Mb.MbData[3]= analogRead(0);  
  
  //Lectura analogica y envio de dato a dir M6  
  Potenciometro1 = analogRead(A1);  
  Mb.MbData[6]= analogRead(1);  
  
  //Lectura digital y asignacion de variables  
  Sel1 = digitalRead(Selector1);  
  Sel2 = digitalRead(Selector2);  
  IR1 = digitalRead(sensorIR1);  
  IR2 = digitalRead(sensorIR2);  
}
```

- **Condicionales**

Se muestran las condiciones a las cuales los actuadores e indicadores se encenderán y apagarán.

```
Arduino_Practica1 MgsModbus.cpp MgsModbus.h  
  
//Condiciones de encendido y apagado de actuadores e  
// indicadores  
if(Sel1 == HIGH){  
  Mb.MbData[12] =1;  
}else{  
  Mb.MbData[12] =0;  
}  
  
if(Sel2 == HIGH){  
  Mb.MbData[13] =1;  
}else{  
  Mb.MbData[13] =0;  
}  
  
if(IR1 == HIGH){  
  Mb.MbData[2] = 1;  
}else{  
  Mb.MbData[2] =0;  
}  
  
if(IR2 == HIGH){  
  Mb.MbData[1] = 0;  
}else{  
  Mb.MbData[1] =1;  
}  
  
int a = bitRead( Mb.MbData[16],0);  
if (a == 1){  
  analogWrite(motor, 1023);  
}else{  
  analogWrite(motor, 0);  
}
```

- **Lectura de datos desde Wincc mediante Modbus**

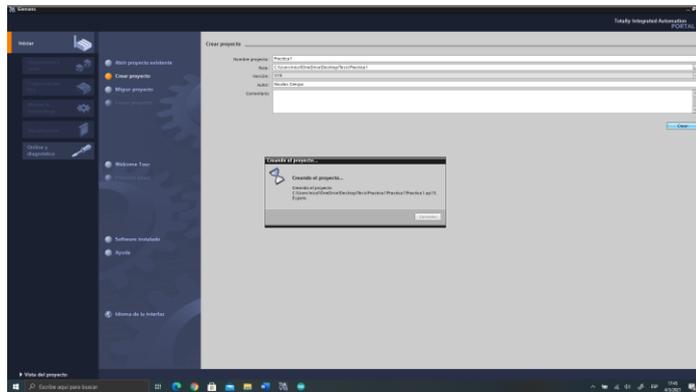
Se leen los datos por medio de Modbus en las direcciones M15 Y M16, así mediante Mb.MbsRun() nos permite inicializar Modbus como servidor.

```
// Lectura de variables Modbus M15 y M16 para encendido
//de actuadores desde pantalla Wincc
digitalWrite(Piloto1 ,bitRead( Mb.MbData[15],0));
digitalWrite(Piloto2 ,bitRead( Mb.MbData[16],0));

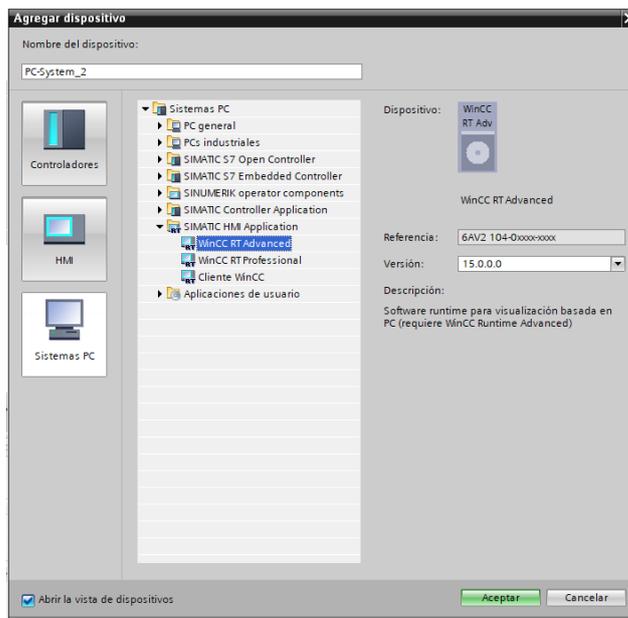
Mb.MbsRun();
}
```

3. Configuración de WinCC SCADA para leer datos MODBUS TCP

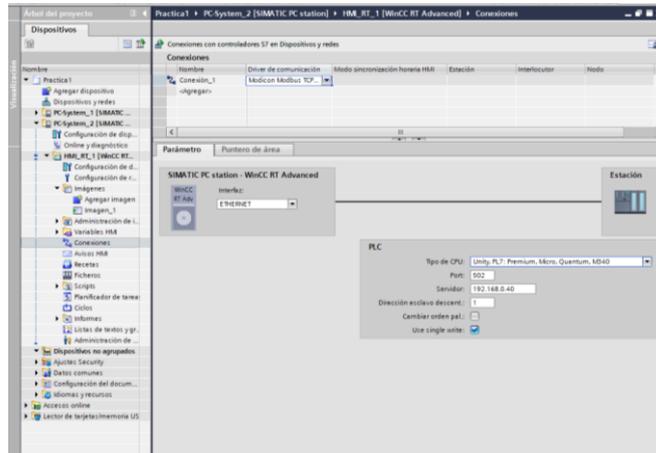
- Creación de un proyecto para esta práctica.



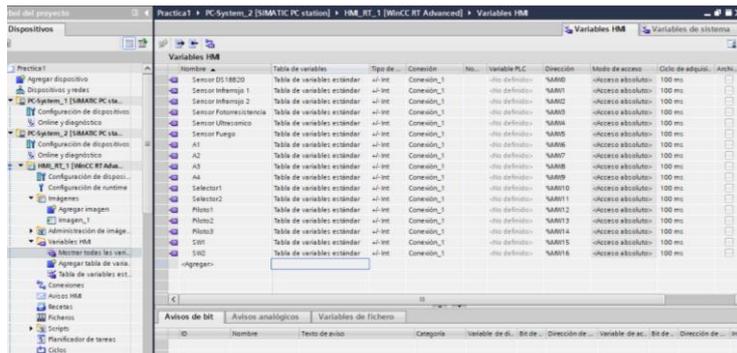
- Se agrega un dispositivo en Sistemas PC WinCC RT Advanced



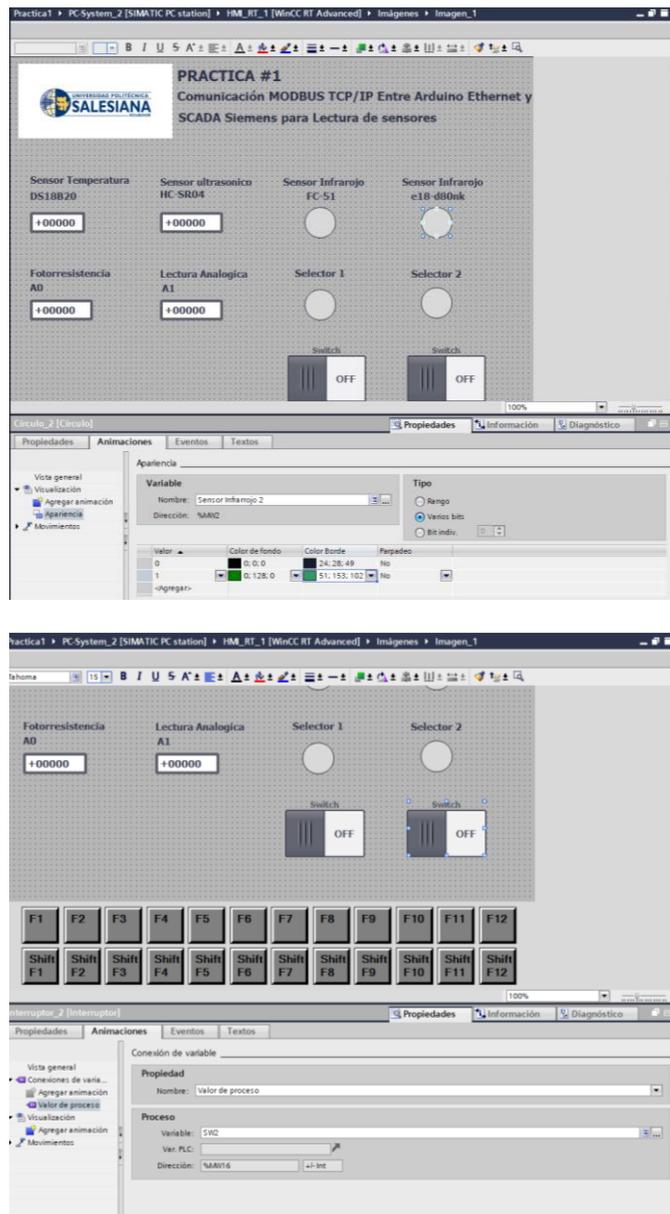
- Se crea una conexión en WinCC, con la dirección IP que configuramos en la estación de Arduino.



- Se insertan las Variables en tabla de variables WinCC, que se van a Leer y Escribir con la dirección %MW correspondiente programada en la Estación de Arduino:



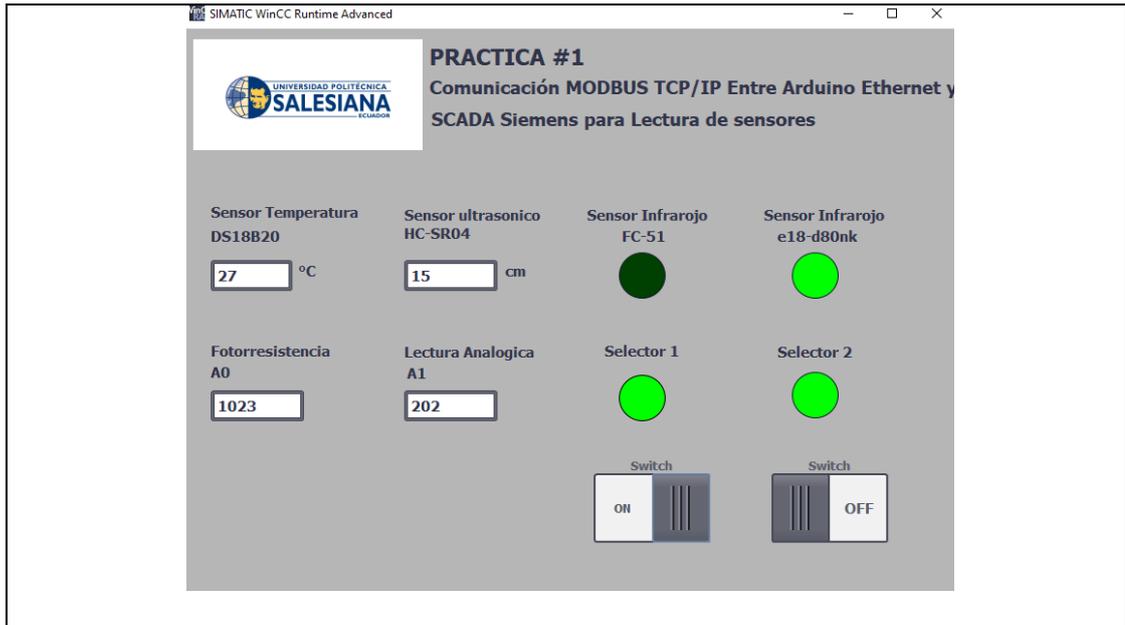
- Se crea la pantalla de usuario y se enlazan las direcciones de entrada y salida a los elementos de lectura y escritura correspondientes como se muestra en las siguientes figuras:



ACTIVIDADES POR DESARROLLAR

1. Acceder al PLC esclavo con un programa de PC o de celular de Modbus TCP, se lo puede obtener desde Windows como “Modbus Poll” o en Android como “Poll Modbus”.
2. Verificar el correcto funcionamiento de la banda transportadora y contemplar fallas en las conexiones TCP.
3. Realizar conexiones a más variables físicas para profundizar en el análisis con parámetros relacionados con potenciales aplicaciones de control en el proceso de automatización.

RESULTADOS OBTENIDOS: Se obtuvo la conexión inalámbrica entre Arduino y WinCC, observando los indicadores en las pantallas de Local y Remoto y en el tablero Esclavo, también se puede activar indicadores en tablero Arduino y motor de la banda transportadora desde WinCC.



CONCLUSIONES: Se implemento una comunicación cliente servidor, por medio de Arduino que envía los datos de lectura de los diferentes sensores y actuadores conectados a una banda transportadora. Además, que por medio del programa Siemens WinCC permite configurar una interfaz para activar de forma manual algunos de los actuadores conectados.

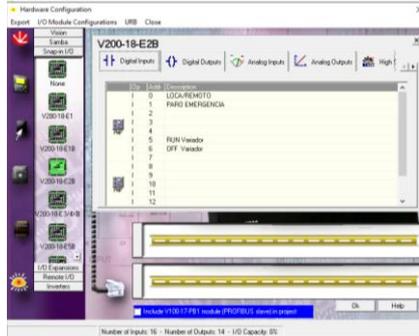
RECOMENDACIONES:

- Agregar una resistencia eléctrica que pueda monitorear la temperatura de cada objeto que pase a través de la banda transportadora.
- Para probar el tablero cliente se recomienda descargar una aplicación que pueda escanear las direcciones Modbus para verificar la conexión.
- Verificar que la dirección IP de la PC este en el mismo rango de direcciones configuradas en las antenas previamente.

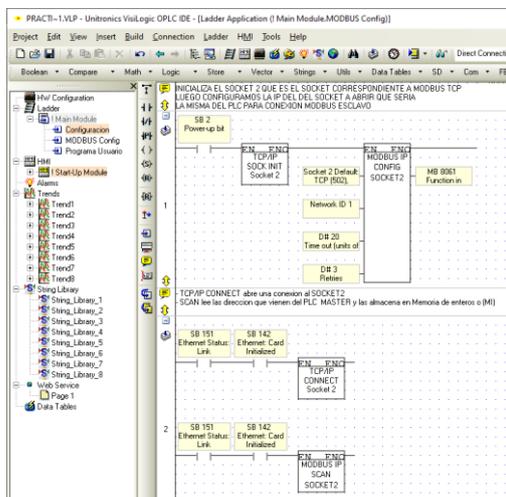
Docente/Técnico Docente: _____

Firma: _____

4.2 Práctica 2

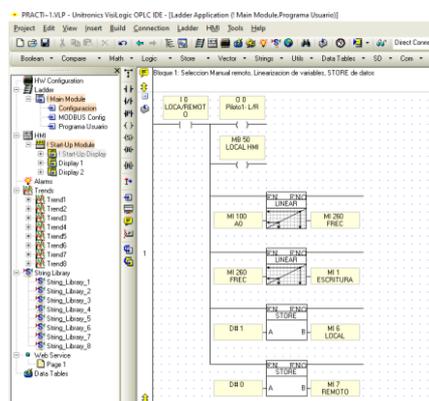
		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados
NRO. PRÁCTICA:	02	TÍTULO PRÁCTICA: Comunicación MODBUS TCP/IP entre PLC Unitronics y SCADA Siemens para variación de velocidad de motor.
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS Slave TCP/IP en PLC Unitronics. 2. Diseñar una red MODBUS TCP/IP entre PLC Unitronics y Programa Wincc. 3. Aprender el uso básico de un variador de frecuencia 4. Crear botones de mando para motor en las pantallas HMI-SCADA. 		
DEV:	1. Introducción: El programa desarrolla una comunicación simple entre un maestro y un esclavo (Cliente y Servidor), el módulo Unitronics esclavo mandará datos de lectura de la frecuencia del motor estado de los LED de arranque, paro, falla y parada de emergencia y la PC con Siemens WinCC leerá estos datos y podrá activar de forma manual alguno de los actuadores conectados.	
	2. Programación y configuración de módulo esclavo Unitronics <ul style="list-style-type: none"> • Se crea un proyecto para el PLC Unitronics con PLC V700 y Modulo I/O V200-18-E2B • Se configura tabla de Entradas Digitales:  <ul style="list-style-type: none"> • Se configura tabla de Salidas Digitales 	

MODBUS Config: Donde estará la configuración de conexión Modbus esclavo del PLC.



- Esta subrutina contiene el bloque MODBUS IP CONFIG donde se configura el socket de conexión, Network ID en su defecto 1, un time out y retries que serán los tiempos de reintento en la conexión Modbus y un Status de salida representado por una Memory Bit (MB).
- El socket2 en Uniflex permite tener una comunicación Modbus Esclavo por defecto.
- El siguiente bloque será para abrir la conexión del Socket2 donde se configura la misma dirección IP del dispositivo y el siguiente, MODBUS SCAN se encarga de escanear las direcciones Modbus de lectura y escritura. SCAN permite leer direcciones enteras y las almacena en la Memoria de enteros (MI).

Programa de Usuario: Donde estará todo el programa principal de control.

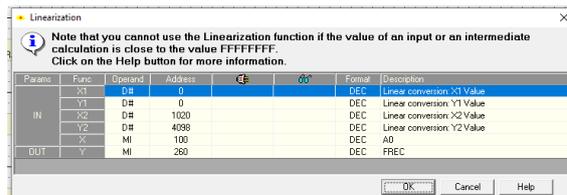


- Cuando la entrada I0 Sea positiva se acciona O2 que es el piloto de operación LOCAL
- M1 100 es la Memoria Entera donde se almacena el dato de la lectura Analógica

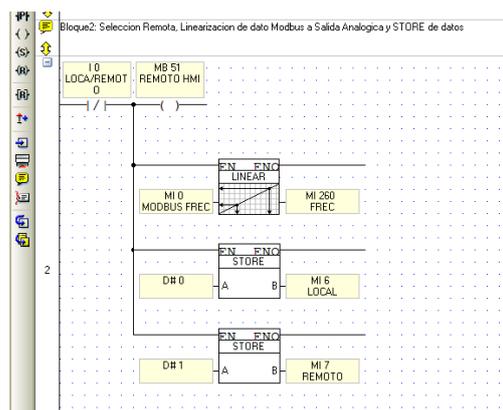
- Para escalar esta señal de entrada analógica AI Escrita en la memoria MI100 y enviarla a la salida analógica del PLC quien esta guardada en la memoria MI260 se debe considerar:

La resolución de AI es de 10 bits (0-10V, 0-20mA, 4-20mA) y el rango dado en el PLC es: (0-1023 Unid, 204-1024 Unid).

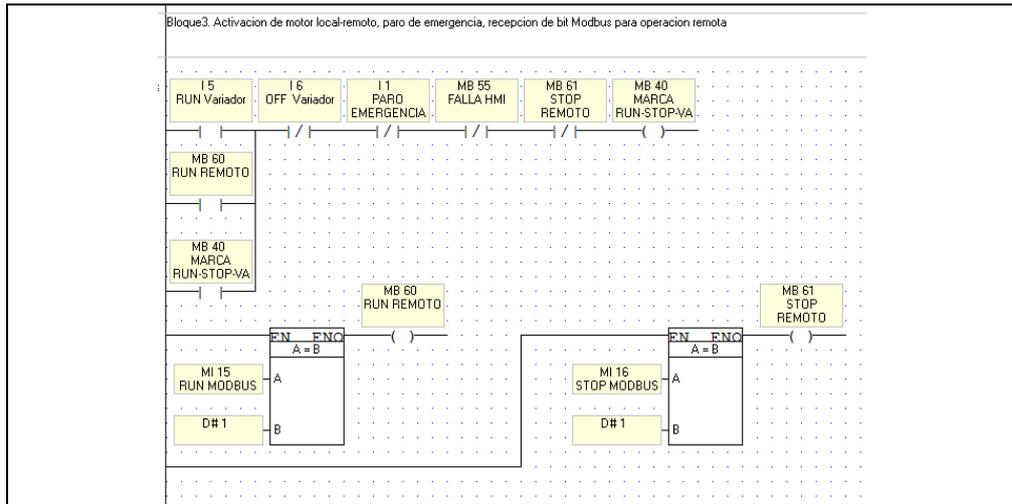
- Para escalar la señal de salida analógica AO se debe considerar:
La resolución de AO es de 12bits (0-10V, 0-20mA, 4-20mA) y el rango dado en el PLC es: (0-4095 Unid, 819-4095 Unid)



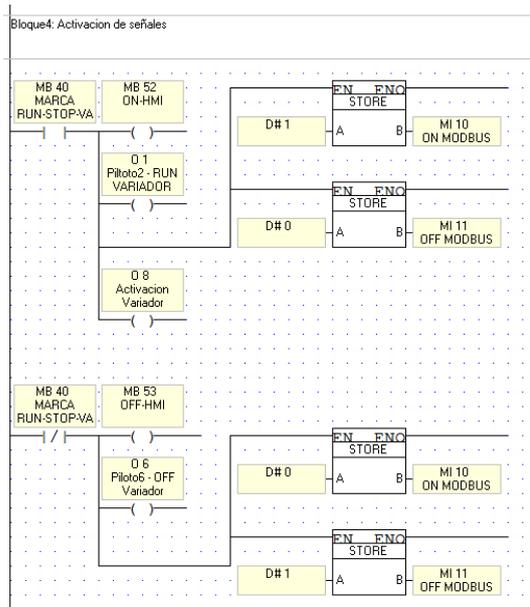
- La señal MI260 Que es la salida Analógica de Frecuencia, se escala para poder visualizarla en valores de 0 a 60Hz.
- La señal de I0 se envía a la dirección MI6 y MI7 que es el indicador de LOCAL/REMOTO de operación del motor se usa el bloque STORE Quien permite escribir un valor contenido en un operando o constante a otro operando.
- Cuando se seleccione la forma remota de operación, el dato de entrada a linealización proviene de la dirección Modbus MI0 quien recibirá el dato con valor de 0 a 60 y se lineará para pasar el dato de 12bits de 0 a 4098 con ajuste, que se escribe en la salida analógica A00 y genera la frecuencia del Motor.



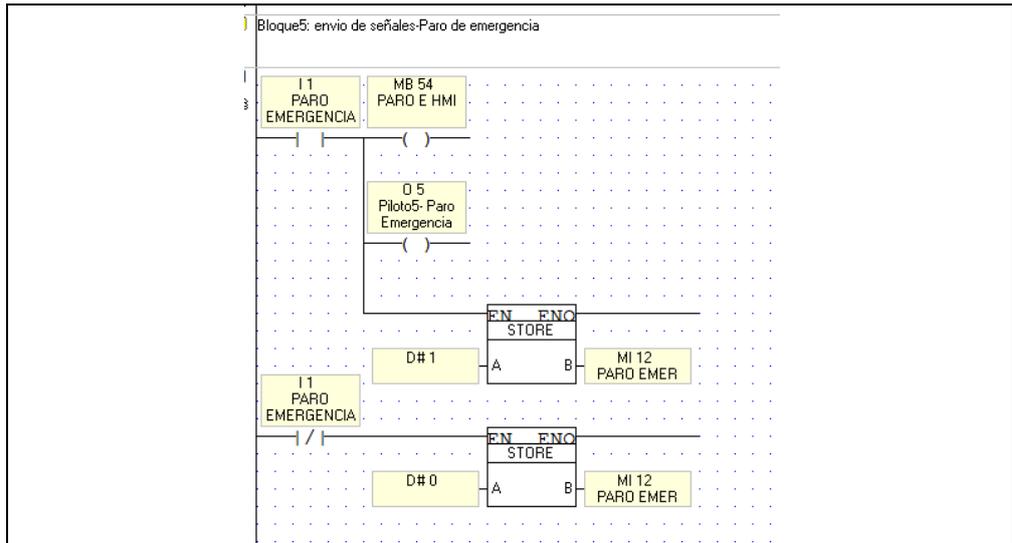
- El siguiente bloque será para activar o desactivar el variador consta de contactos abiertos y cerrados que pertenecen a fallas, paro de emergencia y datos que vienen de Modbus, estos datos Enteros son comparados con el bloque de comparación A=B para activar una bobina de activación o desactivación que se encargara de la acción remota del variador.



- En este bloque cuando se active la señal MB40 enviará un positivo al tablero mediante piloto O1 y hacia O8 que pertenece a la activación del motor, también con el bloque STORE se envía una señal positivo o negativo a las direcciones Modbus correspondientes.



- En este bloque cuando la entrada I1 se active, envía una señal positiva a O5 -piloto de emergencia, activa la señal MB54 para dar stop al bloque 3 y envía las señales mediante STORE a la dirección Modbus que corresponde.



- Finalmente, el bloque 6 se encargará de escalar una señal analógica que simulará sobre corriente, con el bloque de linealización escalamos la señal M101 con valores de (0-1020 Bits) a (0-7 Amp).

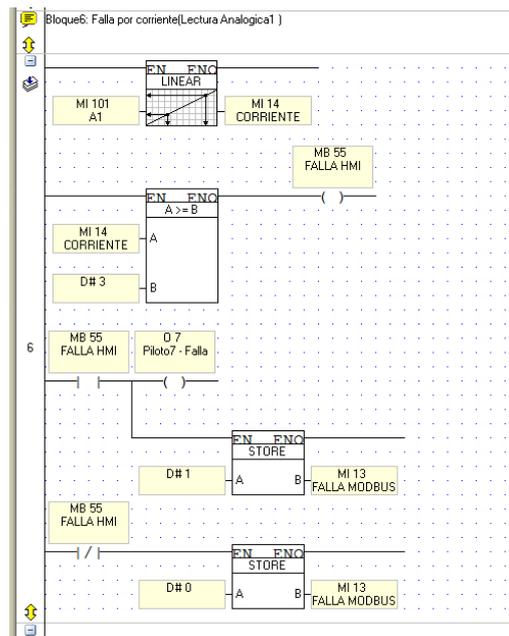
Linearization

Note that you cannot use the Linearization function if the value of an input or an intermediate calculation is close to the value FFFFFFFF.
Click on the Help button for more information.

Params	Func	Operand	Address	IN	OUT	Formal	Description
	X1	D#	0			DEC	Linear conversion: X1 Value
IN	Y1	D#	0			DEC	Linear conversion: Y1 Value
	X2	D#	1020			DEC	Linear conversion: X2 Value
	Y2	D#	7			DEC	Linear conversion: Y2 Value
	X	MI	101			DEC	A1
OUT	Y	MI	14			DEC	CORRIENTE

OK Cancel Help

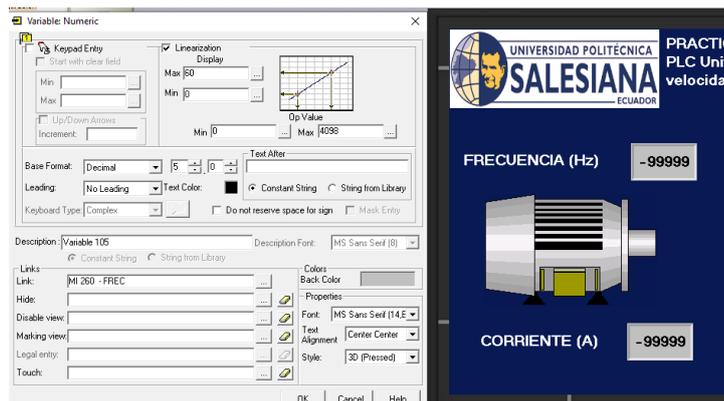
- También mandara un contacto al bloque3 y desactivara el RUN del variador y los datos serán enviados a las direcciones Modbus como correspondan.



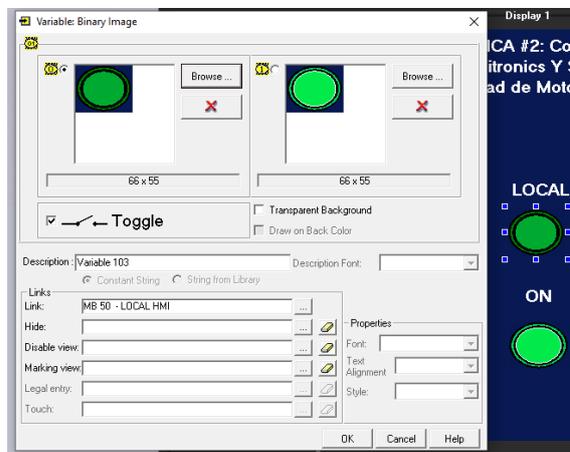
3. Se debe realizar el siguiente esquema para la pantalla HMI de Unitronics:



- La configuración de una variable numérica solo se debe enlazar el Link con la dirección de Memoria entera (MI), que a su vez también es de lectura y escritura Modbus según sea el caso, también nos permite tener acceso a varias configuraciones como escalar desde la variable el valor de esta misma.



- Cuando se trata de una Variable tipo Binary Image, se debe configurar la imagen cuando el valor de la variable sea verdadero y falso, luego asignarle un Link, para este caso se usan direcciones de memoria de bits (MB).

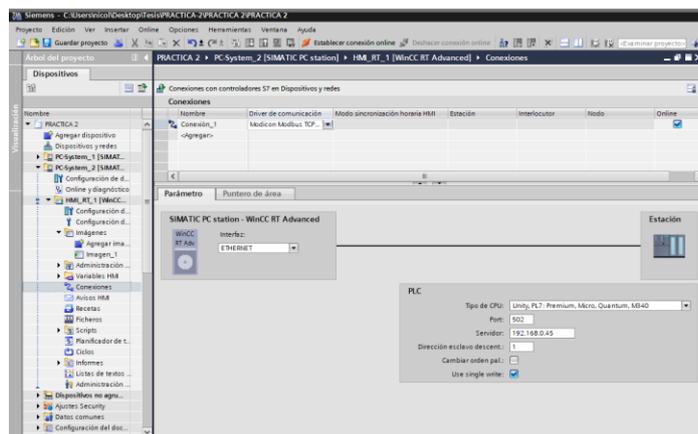


4. Configuración de Modbus TCP Master Wincc

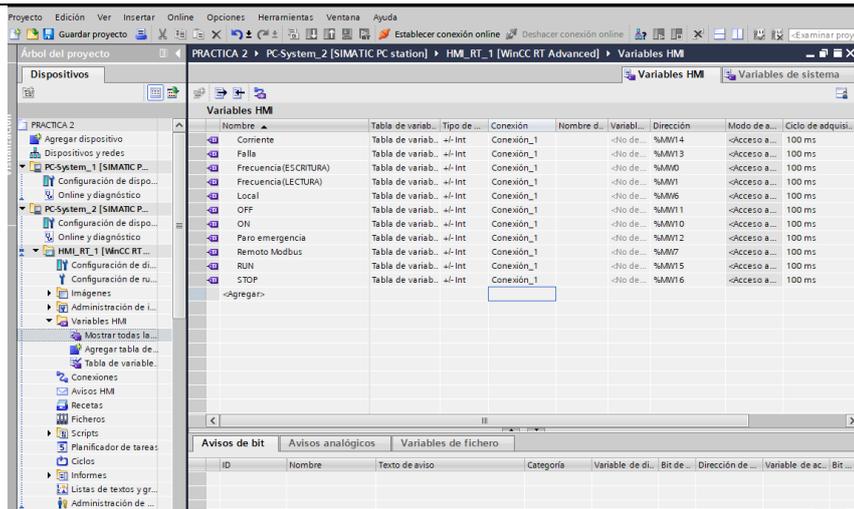
- Se crea un proyecto y en la ruta Agregar dispositivo → Sistema PC → SIMATIC HMI Application, escoger WinCC RT Advance.



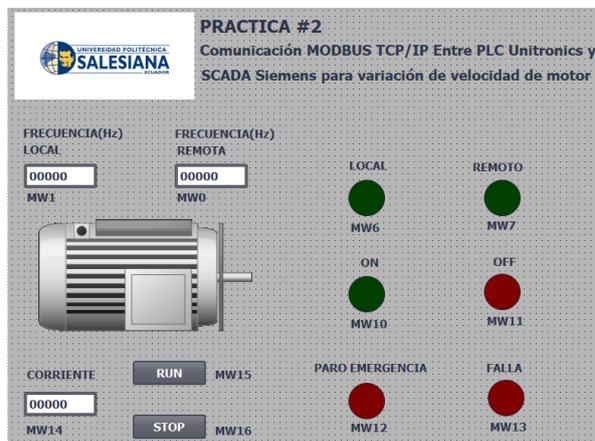
- Se realiza la siguiente configuración tomando en cuenta que la dirección IP y la dirección de esclavo deben ser las mismas configuradas en el PLC Unitronics que es: 192.168.0.45



- Se crea la tabla de variables asignando la conexión creada y las direcciones Modbus, que en este caso empiezan con MW y sus direcciones deben ser las mismas utilizadas para las acciones configuradas en el PLC Unitronics.
- Recordar que el ciclo de adquisición de datos debe ser 100ms para una lectura más rápida de datos.

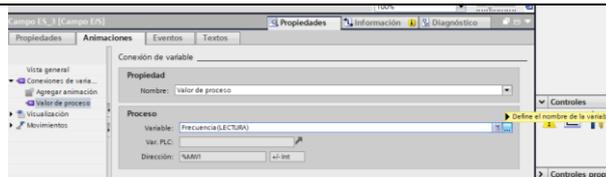


- Se diseña la pantalla en WinCC y se enlaza las direcciones Modbus según corresponda.



- Para una Entrada/Salida tipo campo se debe conectar la variable a una propiedad y en el Proceso conectar la variable ya configurada en la tabla de variables según corresponda a la dirección Modbus programada en el PLC Unitronics.

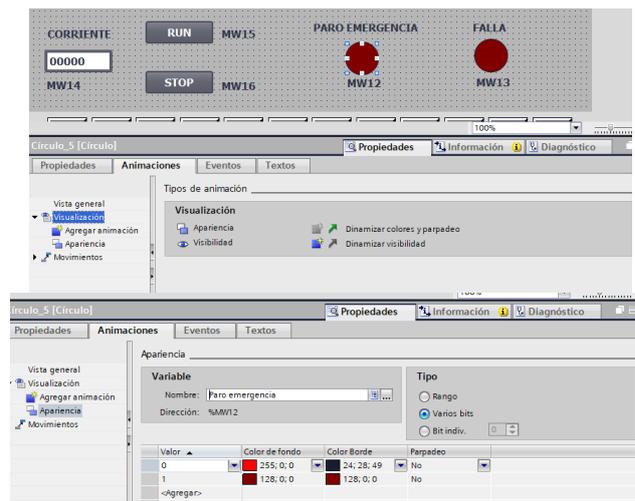




- Cuando es un elemento tipo Botón, se debe usar los Eventos y configurar el bit que se acciona según el tipo de evento seleccionado y la conexión con la variable Modbus.



- Para el caso de elementos básicos como figuras, textos o imágenes se debe configurar la Apariencia, donde se configura los colores del elemento cuando la variable tome un estado entre verdadero y falso y enlazar esta variable a la dirección Modbus según corresponda.



ACTIVIDADES POR DESARROLLAR

1. Dar seguimiento a todas las variables usando Online test para tener un acceso más rápido de la lectura y escritura de variables cuando se esté en fase de pruebas.
2. Acceder al PLC esclavo con un programa de PC o de Celular de Modbus TCP, Se lo puede descargar para Windows como “Modbus Poll” o en Android como “Poll Modbus”
3. Organizar mejor las variables Modbus para una mejor práctica como desarrollador.
4. Verificar el correcto funcionamiento del motor y contemplar fallas en la conexión TCP.

RESULTADOS OBTENIDOS: Se obtuvo la conexión inalámbrica entre Unitronics y WinCC, observando los indicadores tanto en el tablero como en las pantallas de Local y Remoto del tablero Esclavo, también de ingresar el valor de frecuencia y ser leído según corresponda en las pantallas como se muestra en las siguientes imágenes.

WinCC Runtime Advanced



Acceso remoto de pantalla HMI Unitronics



Lectura de Direcciones Modbus en PLC Unitronics desde programador.

Variable	Dir.	Acc.	Dir.	Acc.	Dir.	Acc.	Comentarios
I	Inputs						
Q	Outputs	M	0	✓	25	DEC	MODBUS FREQ
T	Timers	M	1	✓	47	DEC	ESCRITURA MODBUS FREQ.
MB	Memory Bits	M	2		0	DEC	
MB	Memory Bits	M	3		0	DEC	
ML	Memory Long	M	4		0	DEC	
DW	Double Word	M	5		0	DEC	
SB	System Bits	M	7	✓	0	DEC	LOCAL MODBUS
SL	System Long	M	8		0	DEC	REMOTO MODBUS
SDW	System Double Word	M	9		0	DEC	
SF	System Float	M	10	✓	0	DEC	ON MODBUS
SB	System Bits	M	10	✓	0	DEC	

CONCLUSIONES: En esta práctica se pudo realizar un arranque de motor de manera local mediante tablero de Unitronics y de forma remota desde SCADA WinCC también de variar la velocidad del motor desde cualquiera de las dos estaciones y ver los estados de motor en indicadores de pantalla y de tablero, todo esto gracias a la comunicación Modbus TCP/IP inalámbrica entre Unitronics y WinCC de Siemens.

RECOMENDACIONES:

- Para probar el Tablero Esclavo se recomienda descargarse una aplicación que pueda escanear las direcciones Modbus para verificar su conexión.
- Verificar mediante los indicadores Led de las antenas que existe una conexión entre estas.
- Verificar que la dirección IP de la PC este en el mismo rango de direcciones configuradas en las antenas previamente.

Docente/Técnico Docente: _____

Firma: _____

4.3 Práctica 3



FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES

CARRERA: Ingeniería Electrónica

ASIGNATURA: Redes III / Sistemas Microprocesados

NRO. PRÁCTICA:

03

TÍTULO PRÁCTICA: Comunicación MODBUS TCP/IP entre PLC Siemens y SCADA Labview para simulación de control de proceso llenado de tanques.

OBJETIVOS:

5. Implementar una red MODBUS TCP/IP en PLC Siemens.
6. Diseñar una red MODBUS TCP/IP entre PLC Siemens y Labview.
7. Uso de librería MODBUS TCP/IP en Labview
8. Crear un SCADA en Labview.

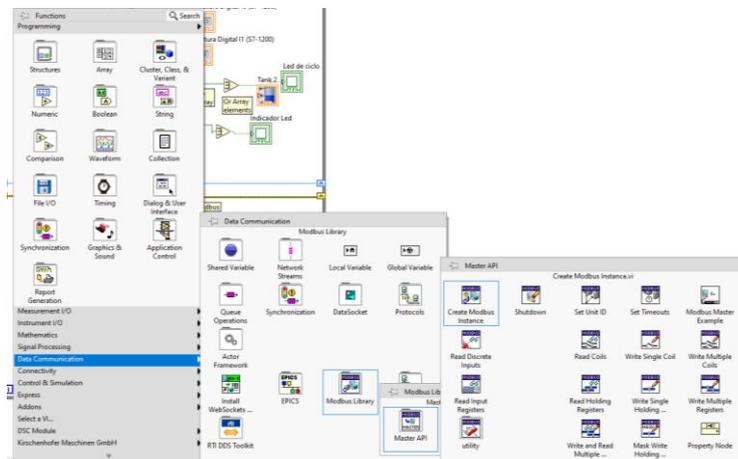
1. Introducción: El programa desarrolla una comunicación simple entre un maestro y un esclavo (Cliente y Servidor), el módulo de prácticas del laboratorio de PLC Siemens S7-1200 enviará y recibirá datos hacia la PC de usuario que visualizará los mismos mediante conexión MODBUS TCP/IP, diseñado en Labview.

2. Desarrollo en LabView:

Utilizaremos los siguientes bloques de programación:

- En la ruta: Data Communication / Modbus Library / Master API, Seleccionamos los bloques de programación siguientes:

Dev:

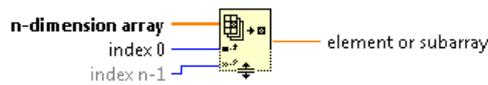


Create Modbus Instance: Es el bloque con el cual podremos inicializar una comunicación Modbus TCP/IP, en este caso Master permite conectarse a cualquier dirección IP de esclavo, el esclavo será configurado con una dirección IP desde el PLC Siemens S7-1200

Read Inputs Registers: Es el bloque que permitirá leer registros de entrada desde el PLC S7-1200, se configura dando una dirección de lectura inicial y final.

Write Multiple Registers: Es el bloque que permitirá escribir múltiples registros dada una dirección inicial.

Index Array Function: Desde Modbus los datos vienen en un arreglo de datos, este bloque de función cambia el tamaño de este arreglo obteniendo así la dirección n del elemento deseado.



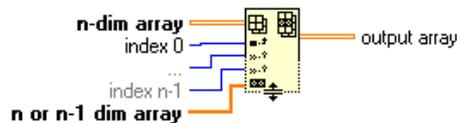
Number to boolean Array: Este bloque de función permite convertir un valor entero a un arreglo booleano, quiere decir que si leemos estados decimales como un 1 o 0 lo transforma a dato booleano para poder ser usado en elementos de indicadores booleanos en el programa.



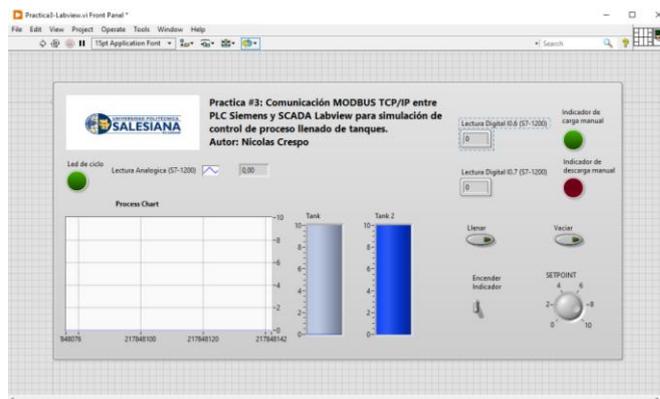
Or Array Elements Function: Retorna valores lógicos si los valores que vienen de un Boolean array son falsos o verdaderos según correspondan.



Insert into Array Function: Este bloque permite insertar elementos múltiples y enviarlos en forma de un array, y nos sirve en esta práctica para enviar cada dirección con su valor a los registros de Modbus TCP/IP.



Desarrollo del Front Panel de Labview.

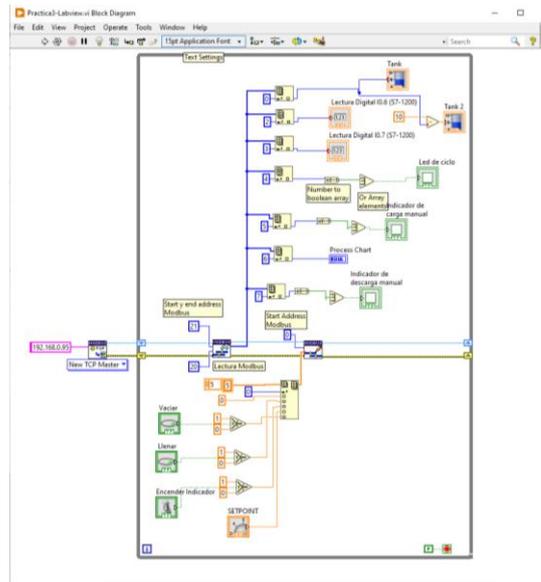


Como podemos observar se tienen indicadores de lectura digital, indicadores led digitales, y un visualizador de lectura analógica que provienen desde el PLC S7-1200, y se tienen botones para llenar o vaciar los tanques que serán programados en el PLC Siemens cuando el switch este colocado en forma remota y local, a su vez se debe ingresar un valor de set que indicará hasta que nivel el tanque debe llenarse y vaciarse.

Desarrollo de diagrama de bloques:

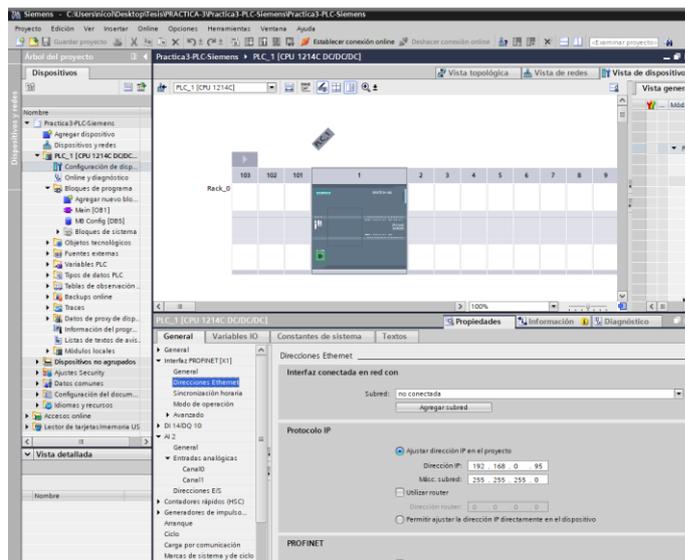
El diagrama de bloques que se presenta, está constituido por una estructura de While Loop que permitirá que los datos se muestren mientras el proceso está en ejecución

en un bucle de repetición, se usan los bloques ya mencionados como el de comunicación Modbus, en el cual se establece el programa como Master de Modbus TCP/IP y se leen 20 direcciones desde M21 Start Address Modbus, se convierten los datos y se separa el array para ser leídos por indicadores programados en la pantalla principal o front panel, cuando se escribe desde Labview se usa un constructor de array que nos permitirá ingresar varios elementos y enviarlos por medio de un solo bloque de escritura y que se empieza a escribir desde la dirección M0 configurada como Start Address Modbus.

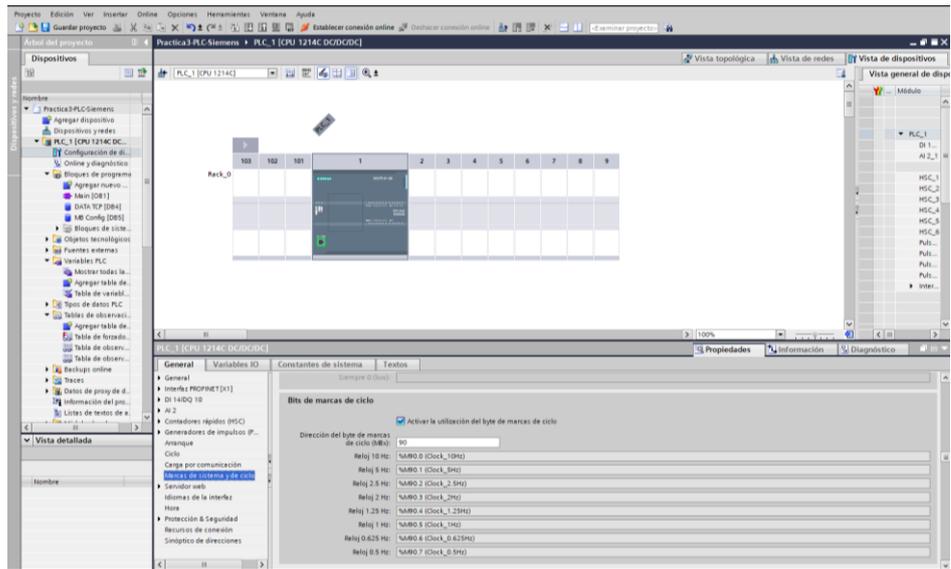


3. Desarrollo en TIA Portal:

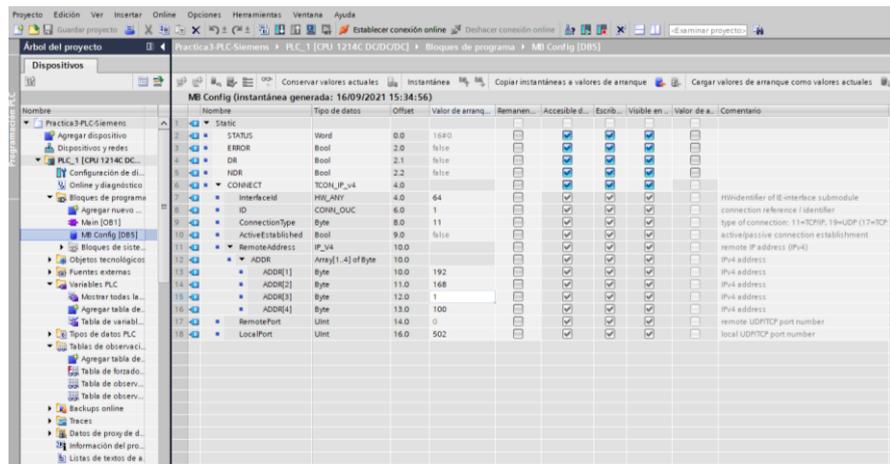
- **Configuración del dispositivo:** Se configura la dirección IP y máscara de subred.



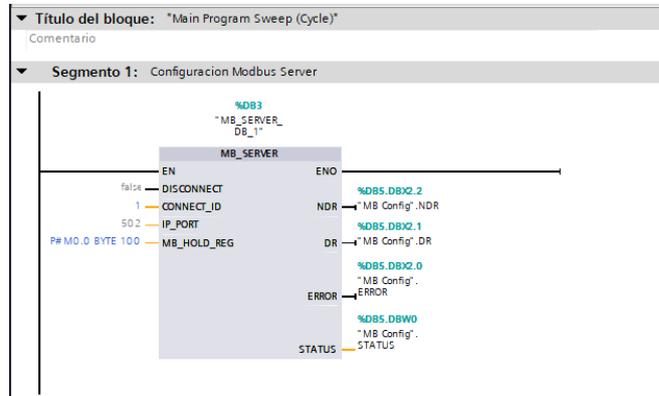
- **Configuración marcas de sistema de ciclo:** Se selecciona la marca M90.5 que es el clock de 1 Hz (1 segundo)



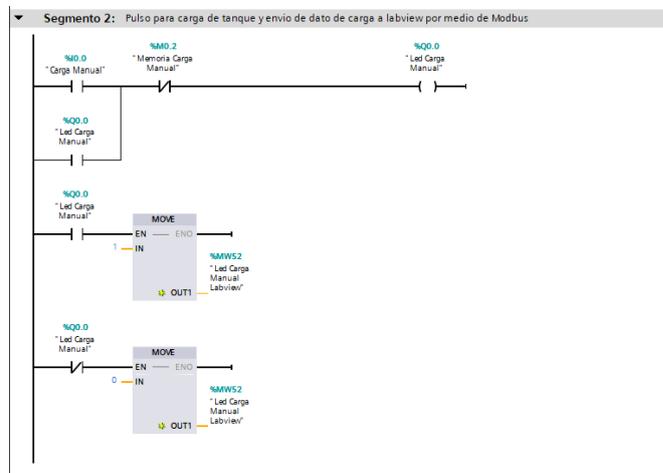
- **Configuración MB:** El siguiente bloque de función muestra los valores de interfaz, se configura el ID = 1, una conexión tipo TCP, dirección IP propia del protocolo Modbus y local port como defecto 502.



- **Configuración Modbus Server:** Lectura con P# M0.0 desde la dirección M0 que almacena 100 BYTES para su lectura, así mismo para la lectura de los valores de Status y de los errores que se configuró anteriormente en MB config.



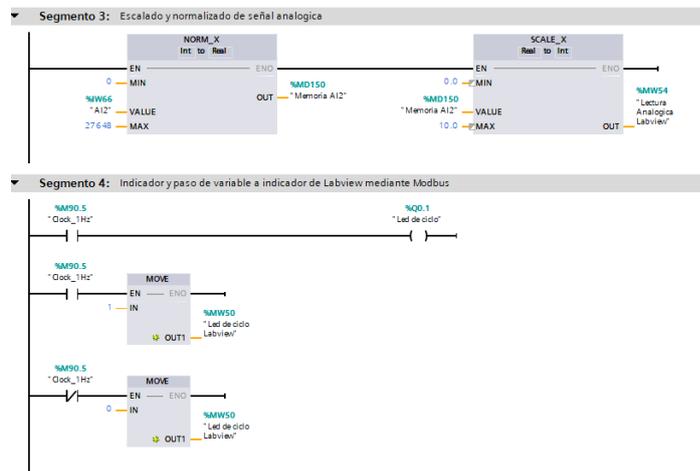
- **Configuración de pulso:** El valor de pulso manual se visualiza en salida Q0.0, como indicador LED. Luego, el dato se envía hacia Modbus a la dirección MW52 para ser vista desde SCADA Labview como un indicador de carga de tanque. Adicionalmente, la memoria de carga manual se desactiva cuando el contador llega al límite configurado por el usuario que será seteado desde Labview y leído en Siemens.



- **Escalado y normalizado de señal analógica:** La función NORM normaliza el dato de la variable de entrada en escala lineal. De esto, se define los parámetros o rangos MIN y MAX que van a reflejar la escala. Así, como mínimo desde cero hasta el valor máximo que es 27648, que va a depender de la tarjeta de entrada analógica para este caso el bloque NORM sirve para normalizar un valor entero a un valor REAL. El dato de salida se guarda en memoria MD150, el dato será enviado a otro bloque para escalar el dato de 0 a 10 unidades, que servirá para lectura de MW54 que mostrará la gráfica puesta en Labview.

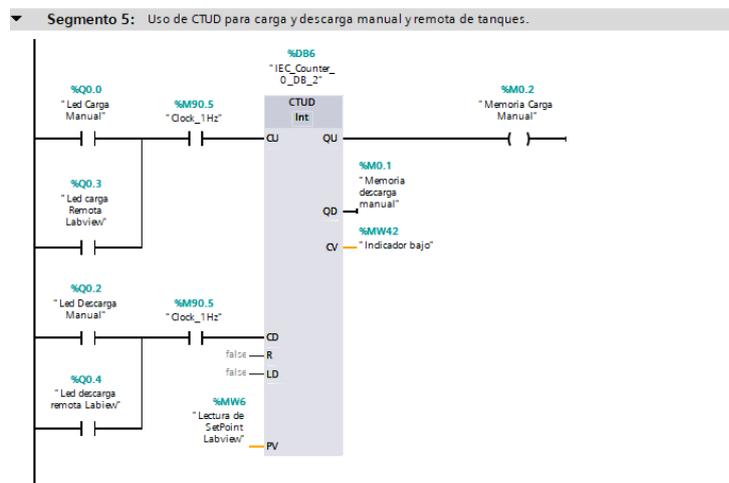
- **Indicador y paso de variable a indicador de Labview mediante Modbus:**

El contacto de ciclo de 1Hz pasa a encender un indicador en el tablero mediante LED de salida Q0.1, para él envío de datos 0 o 1 que indican el encendido o apagado del LED y ser leído en Labview mediante Modbus, se utiliza el bloque MOVE que traslada un valor REAL a la salida MW50 dependiendo si el contacto se abre o cierra.

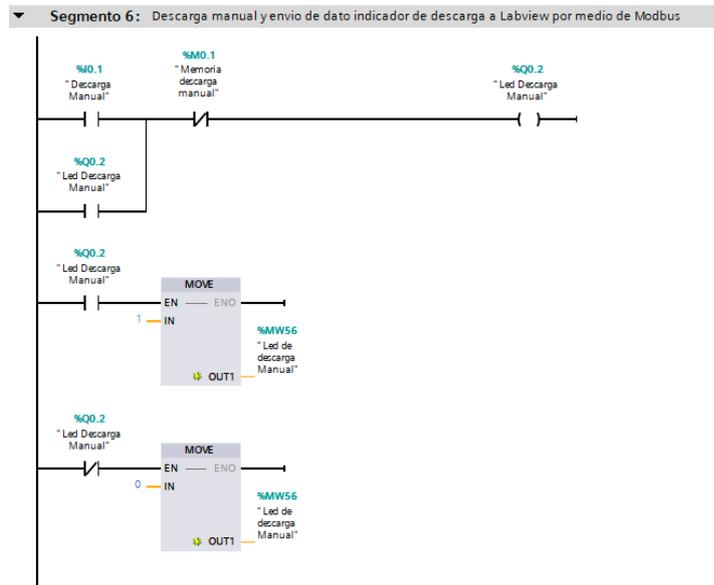


CTUD para carga y descarga manual y remota de tanques:

- Q0.0 pulso de carga manual se activa y el clock de 1 Hz hace el pulso para el contador, hasta el límite establecido por el usuario. Una vez que alcanza el límite se desactiva el contacto.
- Q0.3 pulso de carga remota desde Labview así mismo activa el mismo clock de 1 Hz y empieza la descarga hasta el límite establecido por usuario.
- Similarmente ocurre, con la descarga manual y remota, se activa el mismo clock de 1 Hz y ocurre la acción de descarga.
- En la dirección MW6 está la lectura de Labview que indica el Setpoint o hasta donde ocurre el llenado del tanque.
- Las direcciones M0.1 y M0.2 son los indicadores de nivel alto o bajo.

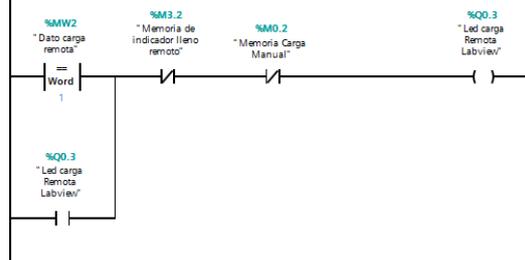


- **Descarga manual y envío de dato indicador a Labview por medio de Modbus:** Similarmente como en el segmento anterior se envía un pulso de descarga ya sea manual o remoto, cuando la memoria de descarga, M0.1, este en 0 se desactiva la descarga. Además, se usa el bloque MOVE para leer la dirección MW56 que es el indicador Led de descarga.

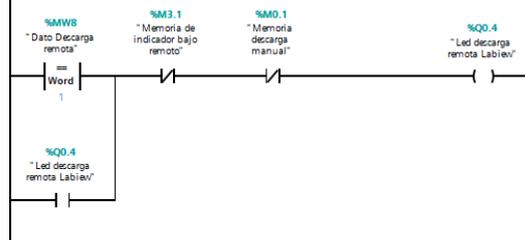


- **Pulsador de carga remota:** La dirección MW2 que viene desde Labview, si valor es 1 (encendido) se envía al Led de carga remota en Siemens, Q0.3. Es decir, este control se puede realizar desde el tablero, así como desde Labview.
- **Pulsador de descarga remota:** similarmente como el segmento anterior se muestra la activación de la descarga que puede ser controlada desde tablero o desde Labview. Esto se desactiva cuando valores de QD y QU están activas.

▼ Segmento 7: Pulsador de carga remota, dato que viene desde labview por medio de Modbus

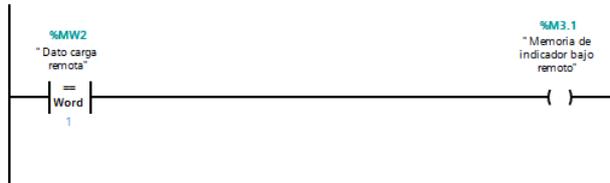


▼ Segmento 8: Pulsador de descarga remota, dato que viene desde labview por medio de Modbus

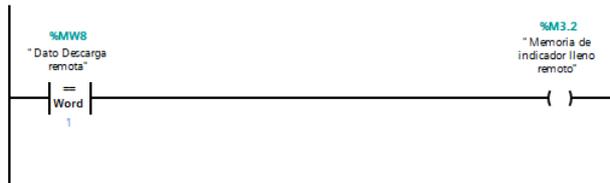


- Dato pulsador que cierra contacto de contador ascendente, es decir, cuando ocurre el pulso ya sea de carga luego la descarga se desactiva, y lo contrario es cierto. Así, se evita que los dos contactos se activen causando conflicto.
- Recibo de datos para encendido de indicador remoto desde Labview, para encender Q0.6 desde Labview que es un indicador.

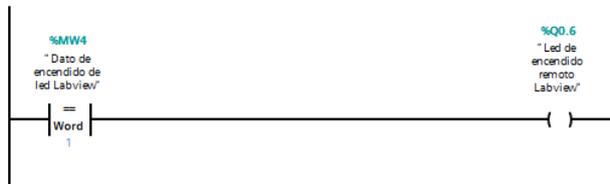
▼ Segmento 9: Dato de pulsador que cierra contacto de contador ascendente



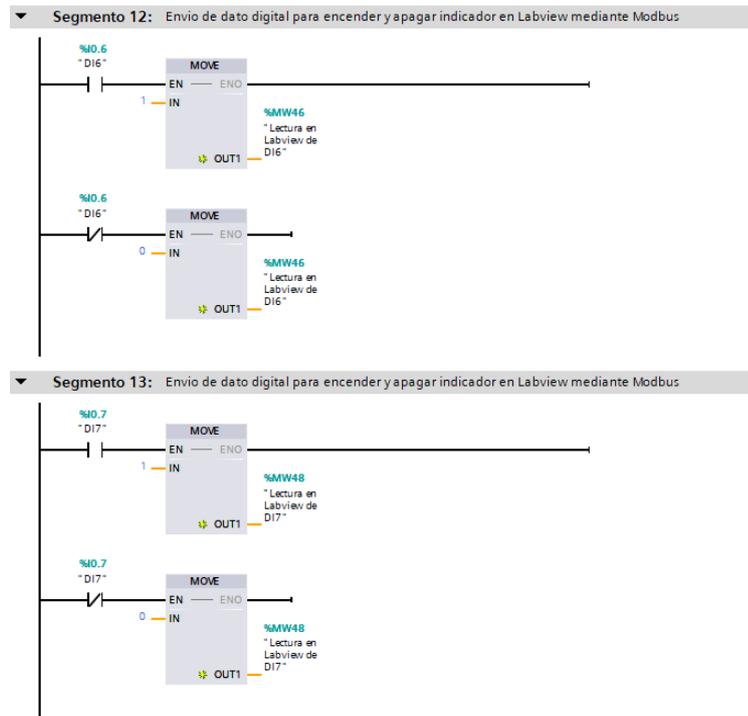
▼ Segmento 10: dato de pulsador de labview que cierra contacto de contador descendente



▼ Segmento 11: Recibo de dato para encendido de indicador remoto desde Labview



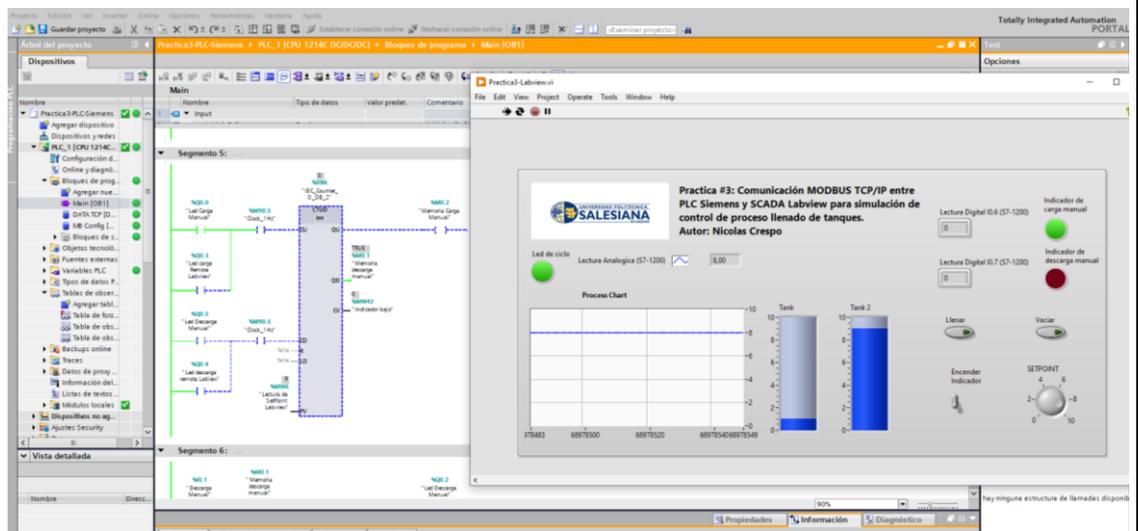
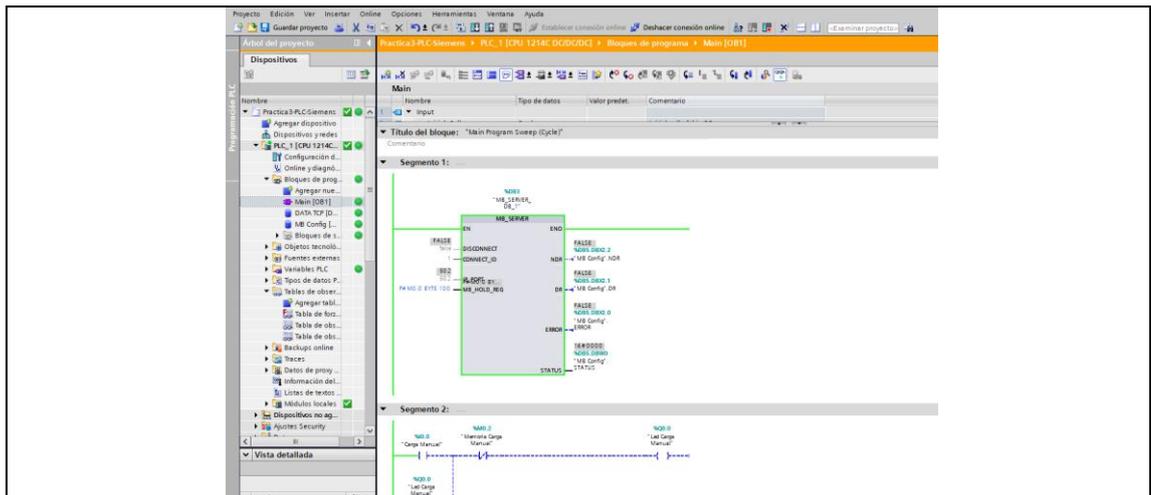
- Envío de dato digital para encender y apagar indicador en Labview mediante Modbus: Se envía desde I0.6, un interruptor en el tablero, para lectura en Labview.
- Así mismo, un interruptor desde el tablero, I0.7, para lectura en Labview.



ACTIVIDADES POR DESARROLLAR

1. Acceder al PLC esclavo con un programa de PC o de celular de Modbus TCP, se lo puede obtener desde Windows como “Modbus Poll” o en Android como “Poll Modbus”.
2. Contemplar fallas en las conexiones TCP.
3. Realizar conexiones para obtener más variables físicas desde PLC 1200 para profundizar el análisis con parámetros relacionados a aplicaciones de conexión con Modbus TCP/IP.

RESULTADOS OBTENIDOS: Se creó una comunicación entre el programa LabView como Modbus master y el PLC siemens como servidor Modbus en el cual se simula el llenado de un tanque y el vaciado del mismo de manera manual y remota. Así mismo, se ha configurado un setpoint de llenado, y se recibió un indicador intermitente con un led de ciclo como indicador. En LabView se pudo ver como se realiza el proceso de SCADA para visualizar los datos que provienen desde PLC Siemens S7-1200.



CONCLUSIONES: Se implementó una comunicación cliente servidor que permite leer y escribir desde cualquier dispositivo sea desde PC con Labview o desde PLC Siemens, además del diseño de una red local entre el PLC Siemens y Labview, donde se desarrolló una interfaz para visualizar los datos que muestra el monitoreo del llenado y vaciado de los tanques tanto remoto y local, con Setpoint configurado por el usuario, además de las mediciones de los procesos en el tiempo, y por último la configuración de un indicador led de ciclo que sirve como indicador de proceso, indicadores de lecturas y escrituras digitales.

RECOMENDACIONES:

- Verificar que la dirección IP de la PC este en el mismo rango de direcciones configuradas en las antenas previamente.
- Podemos verificar mediante una aplicación que lea los datos de Modbus mediante PLC siemens que envía los datos.
- Verificar mediante los indicadores Led de las antenas que existe una conexión entre estas.
- Se recomienda Crear una tabla de observación de variables para poder visualizar desde que dirección %MW se puede leer y escribir, ya que el formato de lectura y escritura de variables en Siemens es distinto ya que maneja registros de lectura y escritura con diferente direccionamiento de función.

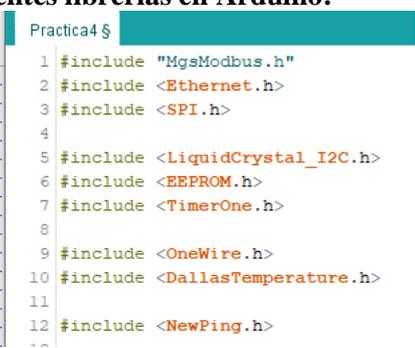
The screenshot displays the 'Dispositivos' (Devices) table in the SIMATIC Manager. The table lists various variables and their configurations. The columns are: 'Nombre' (Name), 'Dirección' (Address), 'Formato visualiza.' (Display format), 'Valor de observac.' (Observation value), 'Valor de forzado' (Forced value), 'Comentario' (Comment), and 'Comentario de variable' (Variable comment). The table contains 24 rows of data, including variables like 'Data carga remota', 'Data de encendido de Ind.', 'Lectura de Señal de Labview', and 'Data Descarga remota'.

Nombre	Dirección	Formato visualiza.	Valor de observac.	Valor de forzado	Comentario	Comentario de variable
1	NAM0	DEC	\$12			
2	"Data carga remota"	NAM2	DEC	0		
3	"Data de encendido de Ind."	NAM4	DEC	0		
4	"Lectura de Señal de Labview"	NAM6	DEC	8		
5	"Data Descarga remota"	NAM8	DEC	0		
6	Configuración d...	NAM10	DEC	0		
7	Online y diagn...	NAM12	DEC	0		
8	Bloques de prog...	NAM14	DEC	0		
9	Agregar nue...	NAM16	DEC	0		
10	Mód (DIP1)	NAM18	DEC	0		
11	Datos TCP ID...	NAM20	DEC	5		
12	MB Config [...]	NAM22	DEC	0		
13	Bloques de s...	NAM24	DEC	0		
14	Objetos tecnol...	NAM26	DEC	0		
15	Fuentes externas	NAM28	DEC	0		
16	Variables PLC	NAM30	DEC	0		
17	Monitor read...	NAM32	DEC	0		
18	Agregar tabl...	NAM34	DEC	0		
19	Tabla de vari...	NAM36	DEC	0		
20	Tipos de datos P...	NAM38	DEC+/-	1		
21	Tablas de obser...	NAM40	DEC	1		
22	Agregar tabl...	NAM42	DEC+/-	0		
23	Tabla de forz...	NAM44	DEC	8		
24	Tabla de obs...	NAM46	DEC	0		
25	Tabla de obs...	NAM48	DEC	0		
26	Backups online	NAM50	DEC	1		
27	Traces	NAM52	DEC	0		
28	Datos de proy...	NAM54	DEC	8		
29	Información del...	NAM56	DEC+/-	0		
30	Listas de textos	NAM58	DEC	5		
31		NAM60	DEC	5		
32		NAM62	DEC	0		
33		NAM64	DEC	0		

Docente/Técnico Docente: _____

Firma: _____

4.4 Práctica 4

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados	
NRO. PRÁCTICA:	04	TÍTULO PRÁCTICA: Comunicación MODBUS TCP/IP entre PLC Unitronics y Arduino Ethernet para lectura de sensores desde HMI Unitronics y arranque de motor variando su velocidad.	
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS TCP/IP en PLC Unitronics. 2. Diseñar una red MODBUS TCP/IP entre PLC Unitronics y Arduino. 3. Realizar lectura de sensores en Arduino. 4. Crear un HMI en PLC Unitronics Para visualización de sensores. 			
1. Introducción: El programa desarrolla una comunicación entre un maestro y un esclavo (Cliente y Servidor), el módulo Unitronics será el Maestro Modbus TCP/IP a la que se conectará el módulo Esclavo Arduino. Desde Unitronics se leerá los datos de sensores conectados a la banda transportadora, también se podrá manejar actuadores que estén conectados a Arduino, adicional se podrá desde Arduino variar la velocidad del motor conectado al módulo de Unitronics. Se debe considerar colocar alarmas visuales de fallo de sistema.			
1. Programación en Arduino: Se utilizarán las siguientes librerías en Arduino:			
DEV:			
	<ul style="list-style-type: none"> • MgsModbus será la librería que contiene la configuración y puesta en marcha de Modbus TCP/IP para Arduino. • Ethernet y SPI: Serán las que permitirán la conexión TCP/IP mediante el Shield Ethernet. • LiquidCrystal_I2C permitirá la conexión con el LCD mediante protocolo I2C la cual nos permitirá usar solo dos cables de conexión y ahorrar espacio en entradas o salidas físicas de Arduino. • TimerOne y EEPROM: TimerOne es una librería para uso de interrupciones internas en Arduino, se la utilizara para ejecutar la función del Sensor de temperatura DS18B20, ya que este tarda 780ms aproximadamente en dar una respuesta haciendo lenta la ejecución del 		

main principal lo que provoca que el main principal tarde en enviar o recibir mensajes Modbus TCP, en esta función de interrupción se guardara el dato en una dirección de la memoria EEPROM para ser leída en el main principal y haciendo que la ejecución del programa tenga el menor retraso posible de ejecución.

- OneWire y DallasTemperature: El protocolo One Wire que usa Dallas Temperature para su sensor de temperatura, permite usar varios sensores conectados a un solo cable permitiendo tener varias resoluciones de lectura para este sensor de 9,10,11 y 12bits.
- NewPing: librería que contiene la configuración de lectura de los sensores ultrasónicos.

Asignación inicial de variables:

```
--
13 #include <NewPing.h>
14 #define TRIGGER_PIN 24
15 #define ECHO_PIN 23
16 #define MAX_DISTANCE 200
17 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
18
19 LiquidCrystal_I2C lcd(0x27, 20, 4);
20 OneWire oneWire(ONE_WIRE_BUS);
21 DallasTemperature sensors(&oneWire);
22
23 MgsModbus Mb;
24 byte mac[] = {0xFE, 0x62, 0xC7, 0xDC, 0xE8, 0xC4 };
25 IPAddress ip(192, 168, 0, 40);
26 IPAddress gateway(192, 168, 0, 1);
27 IPAddress subnet(255, 255, 255, 0);
28
29
30
31
32
33 int pinzumbador = 28; // pin del zumbador
34 int sensorIR1 = 22;
35 int sensorIR2 = 27;
36 int IR1;
37 int IR2;
38
39 int Selector1 = 30;
40 int Selector2 = 31;
41 int Sel1;
42 int Sel2;
43
44 int PinEmergencia = 29;
45 int ParoEmergencia;
46
47 int Piloto1 = 36;
48 int Piloto2 = 37;
49 int Piloto3 = 38;
50 int Piloto4 = 39;
51 int Piloto5 = 40;
52 int Piloto6 = 41;
53
54 int PhotoRes;
55 int Potenciometro1;
56 int Potenciometro2;
57 int Frecuencia;
58 int motor = 7;
59 volatile unsigned long temp = 0;
60
```

- Se define el bus por el cual se van a conectar los sensores DS18B20, el pin ECHO y TRIGGER del sensor ultrasónico, la máxima distancia que tiene el sensor ultrasónico como rango, y luego se configura el sonar con las variables ya declaradas.

- Se asigna la dirección HEX por defecto 0x27 del I2C para el LCD y el tipo de LCD para este caso con dimensiones de 20x4.
- Se asigna el pin ya declarado del sensor de temperatura y se reasigna esta variable para DallasTemperature.
- Se realiza la configuración de direcciones Ethernet y la MAC del dispositivo, además de declarar la variable de Mb para Modbus.
- Finalmente se declaran las variables y se asignan los pines de entradas y salidas, para esto ver la tabla de entradas y salidas de Arduino en el Anexo: Arduino.

Configuración inicial de programa:

```

60
61 void setup() {
62
63   Timer1.initialize(5000000);
64   Timer1.attachInterrupt(SensorTemp);
65
66   lcd.init();
67   lcd.backlight();
68   lcd.setCursor(1, 0);
69   lcd.print("Modulo 3 - Arduino");
70
71   Serial.begin(9600);
72   Serial.println("Serial interface started");
73   Ethernet.begin(mac, ip, gateway, subnet); // start e
74   Serial.println("Ethernet interface started");
75
76   sensors.begin();
77
78   pinMode(Selector1 , INPUT);
79   pinMode(Selector2 , INPUT);
80   pinMode(PinEmergencia , INPUT);
81   pinMode(Piloto1 , OUTPUT);
82   pinMode(Piloto2 , OUTPUT);
83   pinMode(Piloto4 , OUTPUT);
84   pinMode(Piloto3 , OUTPUT);
85   pinMode(Piloto5 , OUTPUT);
86   pinMode(Piloto6 , OUTPUT);
87   pinMode(motor , OUTPUT);
88   pinMode(pinzumbador , OUTPUT);
89
90   Piloto3 = 0;
91   Piloto5 = 0;
92   Piloto6 = 0;
93
94 }
95

```

- Se configura el TIMER en 5 segundos y se le asigna el nombre de la función donde estará el programa de la interrupción.
- Se inicializa el LCD, se enciende la luz de fondo y se escribe un mensaje colocando el cursor en la posición 1,0.
- Se inicializa el puerto serial y el Ethernet, colocando la dirección mac, ip, Gateway, subnet, anteriormente ya declaradas.
- Se inicializa el sensor de temperatura.

- Se establece los pines de entrada y salida, Output para salidas digitales e Input para entradas digitales.
- Se inicializa las salidas Piloto3,4 y 5 en '0' o falso.

Programa Interrupción:

```

214 void SensorTemp(void)
215 {
216     sensors.requestTemperatures();
217     temp = sensors.getTempCByIndex(0);
218     EEPROM.write(0, temp);
219 }

```

- En esta función se colocará la adquisición de datos del sensor de temperatura DS18B20 usando las funciones básicas dadas en la librería de Dallas temperatura y guardamos el dato en la dirección 0 de la memoria EEPROM de Arduino para luego ser leída en el programa principal.
- Esta interrupción será ejecutada cada 5 segundos.

Programa Principal

```

96 void loop() {
97     //Lectura sensor de temperatura
98     int valuetemp = EEPROM.read(0);
99     Mb.MbData[0] = valuetemp ;
.00
.01     if (valuetemp > 40) {
.02         digitalWrite(Piloto4, true);
.03     } else {
.04         digitalWrite(Piloto4, false);
.05     }
.06     //Lectura sensor Ultrasonico
.07     unsigned int uS = sonar.ping();
.08     Mb.MbData[1] = uS / US_ROUNDTRIP_CM;
.09

```

- Se declara una variable para leer la dirección 0 de la memoria EEPROM donde previamente se guarda la información de temperatura en el programa interrupción, esto se realiza ya que el sensor DS18B20 se tarda 780 ms aproximadamente en enviar el dato hacia Arduino, esto causa que el programa principal se demore ese tiempo, haciendo que algunas funcionalidades como leer los datos de Modbus se vean con retardos.
- Se crea una condición de alarma de temperatura con una condicional If, y luego se enciende o se apaga la luz piloto 4 como estado de alarma.
- Se declara la variable uS como tipo unsigned int, que significa que tomará datos de entre 0 a 65,535 que no tiene signo
- Sonar.ping() es el pulso que será enviado y que se genera en la librería de NewPing, y luego se convierte el dato leído para ser leído en unidades de cm, a su vez se escribe el dato en la dirección 1 de Modbus.

```

109
110 //Lectura de Modbus Unitronics switch Local/Remoto
111 int data = Mb.MbData[21];
112 int data2 = Mb.MbData[24];
113 // Mensajes en LCD
114 lcd.setCursor(0, 1);
115 lcd.print("Mando Motor:");
116 if (data2 == 1) {
117     lcd.setCursor(13, 1);
118     lcd.print("PLC Uni");
119 } else {
120     lcd.setCursor(13, 1);
121     lcd.print("Arduino");
122 }
123 lcd.setCursor(0, 2);
124 lcd.print("Frec. Unitronics:");
125 lcd.setCursor(18, 2);
126 lcd.print(data);
127 if (data < 10) {
128     lcd.setCursor(18, 2);
129     lcd.print("0");
130     lcd.setCursor(19, 2);
131     lcd.print(data);
132 }
133 lcd.setCursor(0, 3);
134 lcd.print("Frec. Arduino:");
135 lcd.setCursor(18, 3);
136 lcd.print(Frecuencia);
137
138 if (Frecuencia < 10) {
139     lcd.setCursor(18, 3);
140     lcd.print("0");
141     lcd.setCursor(19, 3);
142     lcd.print(Frecuencia);
143 }

```

- Se reciben los datos enteros de la dirección Modbus 21 y 24, el primero corresponde al dato de la frecuencia del motor que se escribe desde Unitronics, y el segundo es el estado de escritura de la frecuencia (Mando de Motor) ya sea desde Arduino o Desde Unitronics.
- Se crean las condiciones para seleccionar el mensaje de mando de motor en el LCD: PLC Unitronics – Arduino.
- lcd.setCursor establece el cursor en la posición: (columna, fila) y con print se escriben los mensajes a visualizar.
- Se crea una condición para que el dato de la frecuencia no se sobrescriba y se actualice el valor correctamente, como ejemplo; sin esta condición cuando el dato sea 10 y luego 9 se visualizará de esta forma: ‘90’ para esto la condición dice que cuando el valor sea < a 10 el cursor se ubique una unidad más a la derecha escribiendo el valor como: ‘09’.

```

144 //Lecturas Analógicas y envíos de datos Modbus a PLC
145 PhotoRes = analogRead(A0);
146 Mb.MbData[2] = analogRead(0);
147
148 Potenciometro1 = analogRead(A1);
149 Mb.MbData[3] = analogRead(1);
150
151 Potenciometro2 = analogRead(A2);
152 Frecuencia = map(Potenciometro2, 0, 1020, 0, 60);
153 Mb.MbData[7] = Frecuencia;
154
155 //Lecturas de entradas digitales
156 Sel1 = digitalRead(Selector1);
157 Sel2 = digitalRead(Selector2);
158 IR1 = digitalRead(sensorIR1);
159 IR2 = digitalRead(sensorIR2);
160 ParoEmergencia = digitalRead(PinEmergencia);
161

```

- Para la lectura analógica de Arduino se utiliza la línea de código: analogRead(AI), y luego se escribe el dato a la dirección Modbus que corresponda.
- La función “map” permite linearizar un numero de un rango a otro y convertirá según el valor de lectura analógica de Arduino con resolución de 10bits (0-1023), para este caso el valor de salida será de 0 a 60 con unidad de Hz que será enviada al PLC Unitronics mediante Modbus en la dirección 7. Configuración: map(value, fromLow, fromHigh, toLow, toHigh). Dónde: Value es el valor a mapear, fromLow es el límite inferior del rango actual del valor, fromHigh el límite inferior del rango objetivo del valor, y toHigh el límite superior del rango objetivo del valor.
- Se asigna una variable a las lecturas digitales mediante el código digitalRead()

```

162 //Condiciones de programa
163 if (ParoEmergencia == HIGH) {
164   Mb.MbData[8] = 1;
165 } else {
166   Mb.MbData[8] = 0;
167 }
168
169 if (Sell == HIGH) {
170   Mb.MbData[4] = 1;
171 } else {
172   Mb.MbData[4] = 0;
173 }
174
175 if (Sel2 == HIGH) {
176   Mb.MbData[5] = 1;
177 } else {
178   Mb.MbData[5] = 0;
179 }
180
181 if (IR1 == HIGH || IR2 == LOW ) {
182   digitalWrite(pinzumbador, true);
183 } else {
184   digitalWrite(pinzumbador, false);
185 }
186 if (IR1 == HIGH) {
187   Mb.MbData[6] = 1;
188 }
189 if (IR1 == LOW) {
190   Mb.MbData[6] = 0;
191 }
192 if (IR2 == HIGH) {
193   Mb.MbData[8] = 0;
194 }
195 if (IR2 == LOW) {
196   Mb.MbData[8] = 1;
197 }

```

- Se establecen las condiciones básicas del programa que es indicar si un valor toma el estado de 1 o 0, en este caso HIGH o LOW, y ese valor enviarlo al PLC Unitronics mediante Modbus en las direcciones puestas en el programa, también se puede indicar si una variable o también la otra pueden hacer que un valor tome el estado de 1 o 0 colocando el símbolo de or de esta manera: ||

```

198
199     digitalWrite(Piloto1 , bitRead( Mb.MbData[22], 0));
200     digitalWrite(Piloto2 , bitRead( Mb.MbData[23], 0));
201
202     int a = bitRead( Mb.MbData[23], 0);
203     if (a == 1) {
204         analogWrite(motor, 90);
205     } else {
206         analogWrite(motor, 0);
207     }
208
209     Mb.MbsRun();
210
211 }
212

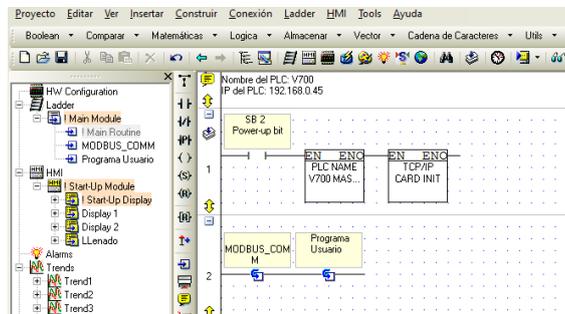
```

- Por último se lee el dato Modbus desde Unitronics usando la función bitRead(), para escribir el valor de True or False en la función de digitalWrite(). Estos dos se visualizarán como luces piloto.
- En la última condición se establece el encendido o apagado del motor de la banda transportadora, pero usando escritura analógica para variar su velocidad y que tiene como función: analogWrite (). El valor escrito como '90' puede ser establecido de 0 a 255 que define la velocidad en bits y corresponde al voltaje de salida de 0 a 5V.
- Mb.MbsRun() define el tipo de conexión Modbus TCP/IP lo que significa que esta conexión es de tipo Modbus Server.

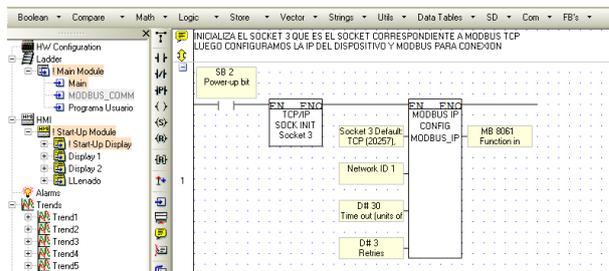
3. Programación en PLC Unitronics.

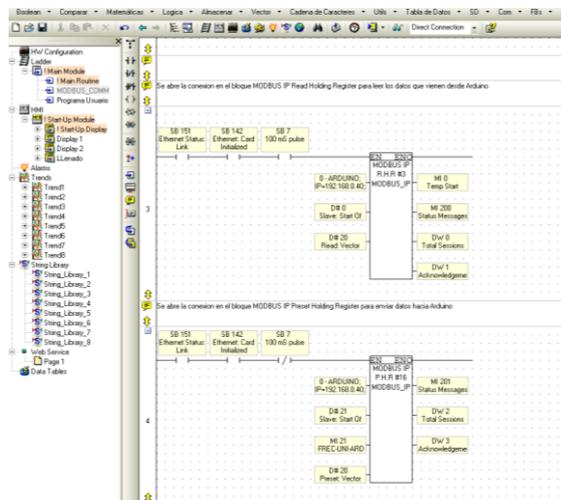
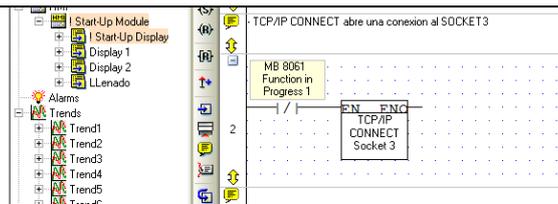
Se crearán 3 Subrutinas en la programación de PLC Unitronics:

- **Configuración:** Donde estará el nombre del PLC en este caso V700 MASTER y la dirección IP del dispositivo: 192.168.0.45



- **Configuración Modbus:** Donde está la configuración de Modbus Master del PLC y la conexión con el esclavo Arduino.





- Cuando la señal de Pulso SB7 sea verdadera se activará este bloque, que consiste en leer los registros enviados desde Arduino

-Se configura la IP con la de Arduino: 192.168.0.40

- Para este caso se leerá desde el vector 0 hasta el Vector 20 que son las direcciones Modbus IP

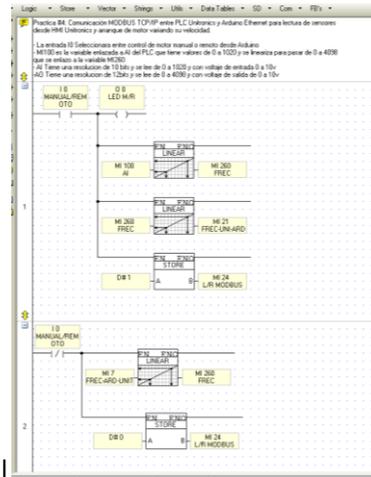
- El bloque empezara a leer las direcciones desde MIO.

-SB7 Es una señal de Pulso de 100ms que estará en contacto abierto

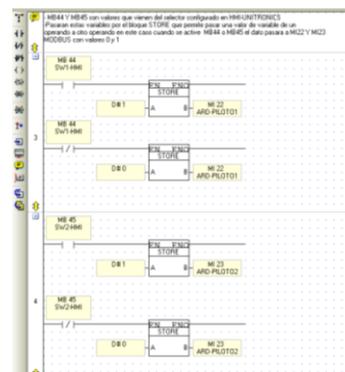
Programa de Usuario:

- Se define el tipo operación de control entre motor manual o remoto desde el Arduino por medio de la entrada IO. Así, cuando IO es positivo se acciona O0 que es el indicador de control manual.
- Luego, para escalar la señal de entrada analógica AI, escrita en MI 100, se envía hacia a la salida analógica del PLC que tiene valores entre 0 a 1020 unidades y es linealizada para pasar de 0 a 4098 unidades, guardada en la memoria MI 260, se debe considerar los siguiente:
 - La resolución del AI es de 10 bits (0-10V, 0-20mA, 4-20mA) y el rango en el PLC es tal que: (0-1020 Unid, 204-1024 Unid).
- Así mismo, para escalar la señal analógica A0 se debe considerar: La resolución de AO es de 12 bits (0-10V, 0-20 mA, 4-20mA) y el rango dado en el PLC es: (0-4098 Unid)
- La salida analógica de frecuencia, señal MI260, se escala para poder visualizarla en valores de 0 a 60 Hz.

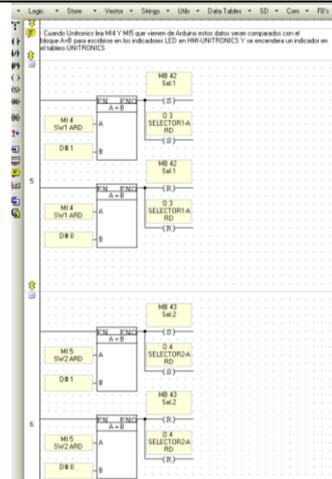
- La señal de I0 se envía a la dirección MI21 y MI24 que es el indicador de MANUAL/REMOTO de operación del motor, se usa el bloque STORE quien permite escribir un valor contenido en un operando o constante a otro operando.
- Por otro lado, cuando se selecciona la forma remota de operación, el dato de entrada a ser linealizado proviene de la dirección Modbus MI100 quien recibirá el dato con valor de 0 a 60 (Hz) y se linealiza para pasar el dato de 12 bits de 0 a 4098 (unidades) Con ajuste, que se escribe en la salida analógica AO0 y genera la frecuencia del motor



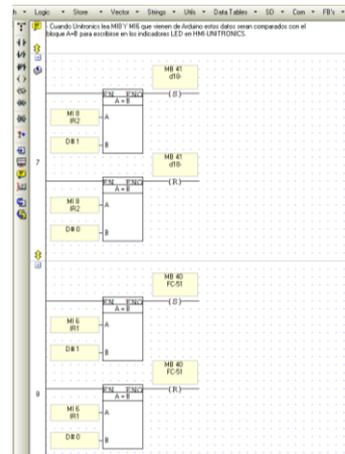
- En el siguiente bloque los valores del selector configurado en HMI-UNITRONICS, MB44 y MB45, enviarán las variables de encendido y apagado hacia el bloque STORE, donde los valores de una variable (MB) pasa hacia otro operando MI, cuando se active una de estas señales el valor será enviado a las direcciones Modbus correspondientemente. Luego, irán hacia el módulo de Arduino.



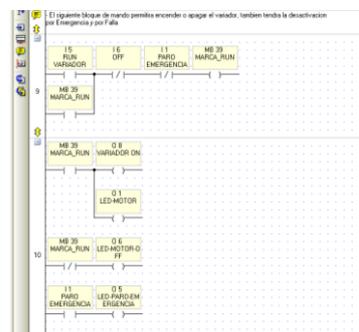
- Cuando Unitronics lea MI4 y MI5 que vienen de Arduino estos datos serán analizados con el bloque de comparación A=B para escribirse en los indicadores LED en HMI-UNITRONICS, así encendiendo un indicador en el tablero.



- Luego cuando UNITRONICS lea MI8 y MI6 que vienen de Arduino estos datos serán analizados en un bloque de comparación A=B para escribirse en los indicadores LED en HMI-UNITRONICS que será mostrado en el tablero.



- El siguiente bloque de mando será para activar o desactivar el variador, consta de contactos abiertos y cerrados que pertenecen a fallas y paro de emergencia.



ACTIVIDADES POR DESARROLLAR

1. Realizar conexiones para obtener más variables físicas desde PLC Unitronics para profundizar en el análisis con parámetros relacionados con las aplicaciones.
2. Dar seguimiento a todas las variables usando Online test para tener un acceso más rápido de la lectura y escritura de variables cuando se esté en fase de pruebas.
3. Verificar el correcto funcionamiento del motor y contemplar fallas en la conexión TCP.

RESULTADOS OBTENIDOS:

Se obtuvo la lectura de variables del sensor mediante una interrupción sin que existan retardos, comparado con el sistema configurado con Arduino que toma más tiempo. Además, como se muestra en la figura debajo podemos apreciar la interfaz de Unitronics en funcionamiento para un caso particular, obtenemos la temperatura del medio a 25°, muestra la lectura de los diferentes sensores como el de fotorresistencia, el sensor ultrasónico, sensor el de lectura analógica y la frecuencia a la que va el motor. Además, de los indicadores LED que muestra los selectores que se encienden cuando un indicador es activado.



CONCLUSIONES: En esta práctica se realizó un sistema de comunicación entre el cliente, Modbus TCP/IP, conectado a un servidor de Arduino, donde de manera remota mediante un tablero de Unitronics recibe la dirección IP de Arduino, donde se pudieron obtener los datos de los sensores conectados al Arduino. Además, que se pudo controlar la velocidad del motor desde Arduino o desde el PLC Unitronics.

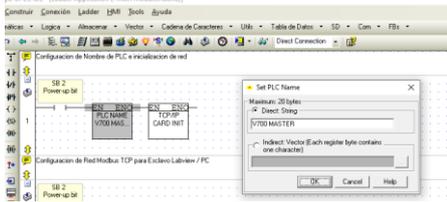
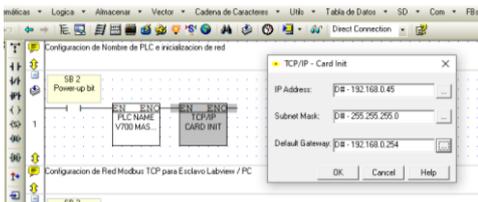
RECOMENDACIONES:

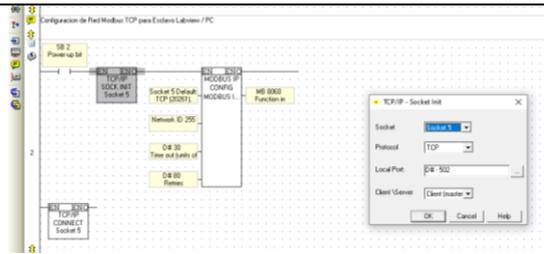
- Considerar colocar alarmas visuales de fallo de sistema
- Para probar el tablero cliente se recomienda descargar una aplicación que pueda escanear las direcciones Modbus para verificar la conexión.
- Verificar que la dirección IP tanto del Arduino y PLC Unitronics estén en el mismo rango de direcciones configuradas en las antenas previamente.
- Antes de encender los equipos asegurarse que todo está apagado, y en primer lugar energizar el variador de frecuencia, luego verificar que no exista falso contacto o que el enchufe de conexión no se encuentre flojo al momento de conectar.
- Para evitar interferencias recibidas dada la conexión con el Variador de frecuencia es recomendable desconectar los dispositivos USB del computador desde donde se controle el sistema.

Docente/Técnico Docente: _____

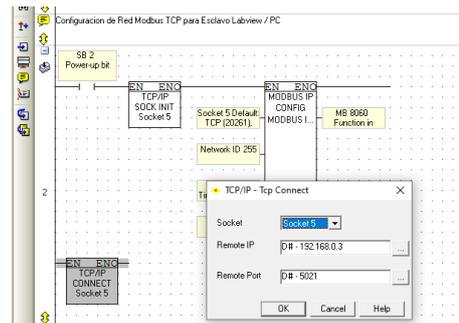
Firma: _____

4.5 Práctica 5

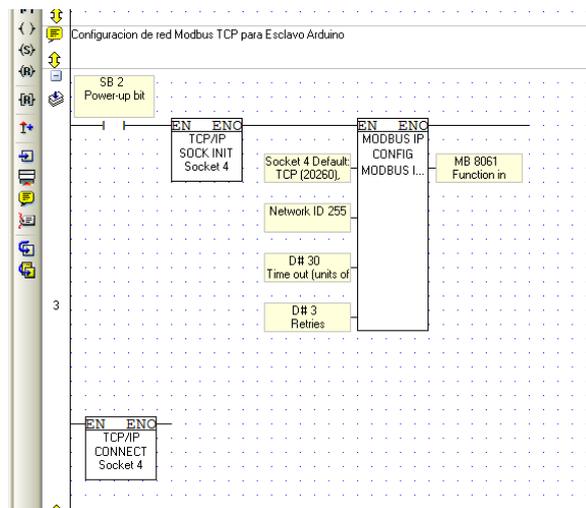
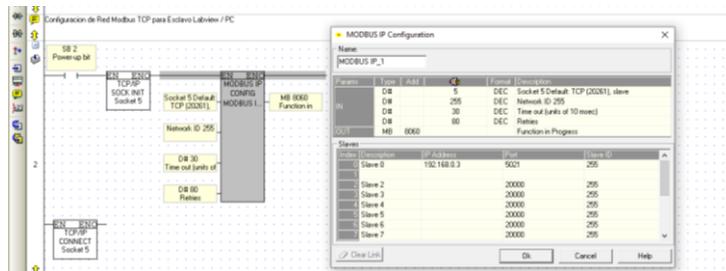
		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados
NRO. PRÁCTICA:	05	TÍTULO PRÁCTICA: Diseño de MODBUS TCP/IP entre PLC Siemens S7-1200, PLC Unitronics, Arduino Ethernet y PC.
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS TCP/IP en PLC Siemens S7-1200. 2. Diseñar una red MODBUS TCP/IP entre PLC Siemens S7-1200, PLC Unitronics, Arduino Ethernet y PC. 3. Interactuar entre los tres módulos enviando y recibiendo variables que puedan realizar acciones puntuales, como encendido de actuadores. 4. Uso de librería MODBUS TCP LabView 		
DEV:	1. Introducción: El programa desarrolla un sistema de monitoreo de variables Modbus TCP/IP, cuyas direcciones serán probadas desde monitor de variables en PLC Unitronics, tabla de variables en PLC Siemens, indicadores en Labview PC y LCD I2C o Monitor serial de Arduino.	
	2. Desarrollo en PLC Unitronics: <ul style="list-style-type: none"> • Se configura el nombre del PLC Y LA IP del dispositivo: 192.168.0.45 - 192.168.0.254 - 255.255.255.0 <div style="text-align: center;">   </div> <ul style="list-style-type: none"> • El socket 5 en Unitronics permite usar una conexión como cliente TCP. • El bloque MODBUS IP CONFIG es de configuración MODBUS cliente o master para establecer conexión con PC - LABVIEW 	

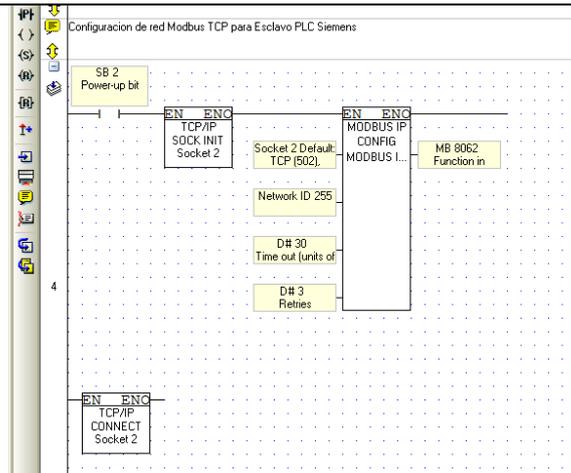


- La dirección IP 192.168.0.3 con puerto 5021 corresponderá a la dirección IP de la PC donde se configura Modbus en LABVIEW.

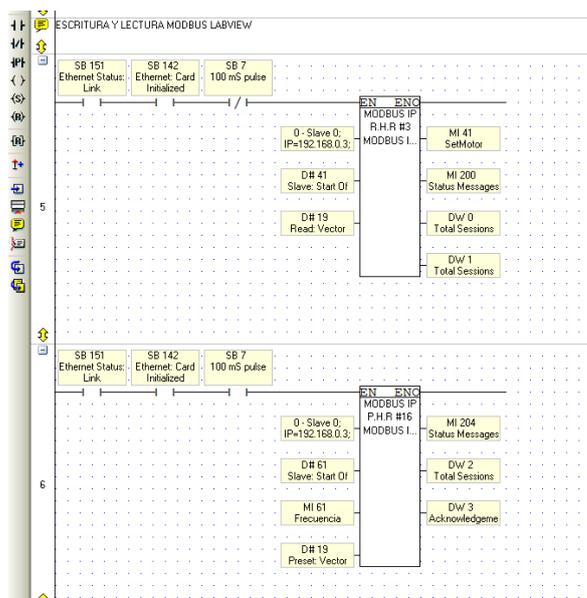


- Configuración de bloque Modbus TCP para dispositivos PC-LABVIEW, Arduino Ethernet, PLC S71200 de Siemens.
- La configuración se realiza con las direcciones IP de cada dispositivo.

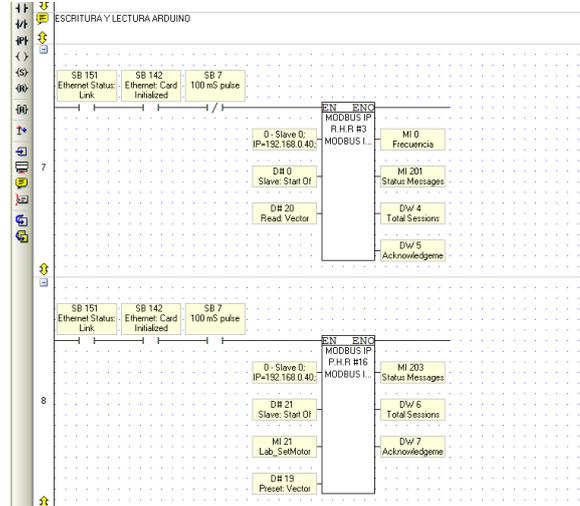




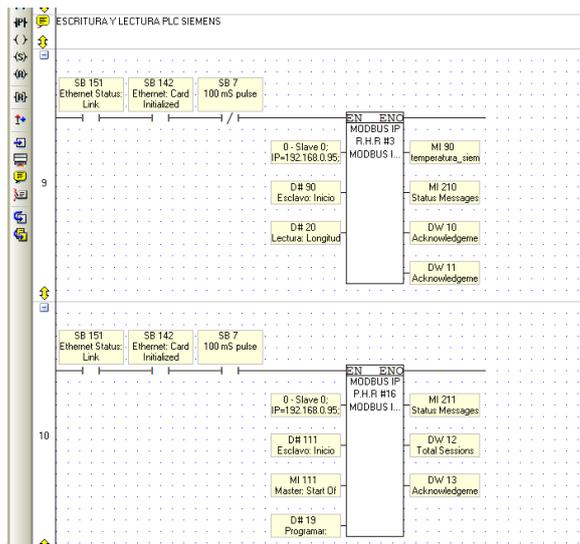
- Los siguientes bloques se usan para lectura y escritura en cada uno de los dispositivos.
- Se usa una señal de pulso que permitirá leer y escribir mensajes Modbus TCP, en Unitronics, solo se puede leer o escribir, pero no ambas, para ello se crea una instancia que permita saber si se está recibiendo información o se está escribiendo para detener una de estas acciones, pero como la comunicación debe ser en tiempo real se debe alternar la lectura y escritura usando un System Bit de 100 milisegundos, es decir que cada 100 mS se va a alternar las lecturas y escrituras entre dispositivos.
- **Lectura y Escritura Labview:** Las direcciones de lectura son: desde MI41 con vector de 19 datos que culmina en MI60. Las direcciones de escritura son: desde MI61 con vector de 19 datos que culmina en MI80. Estos datos de Modbus son los disponibles para PC – Labview.



- Lectura y Escritura Arduino:** Las direcciones de lectura son: desde MI0 con vector de 20 datos que culmina en MI20. Las direcciones de escritura son: desde MI21 con vector de 19 datos que culmina en M40. Estos datos de Modbus son los disponibles para Arduino.



- Lectura y Escritura PLC Siemens:** Las direcciones de lectura son: desde MI90 con vector de 19 datos que culmina en MI109. Las direcciones de escritura son: desde MI111 con vector de 19 datos que culmina en MI129. Estos datos de Modbus son los disponibles para PLC – Siemens. Recordar que estas direcciones Modbus pueden ser escaladas por el programador a su conveniencia.



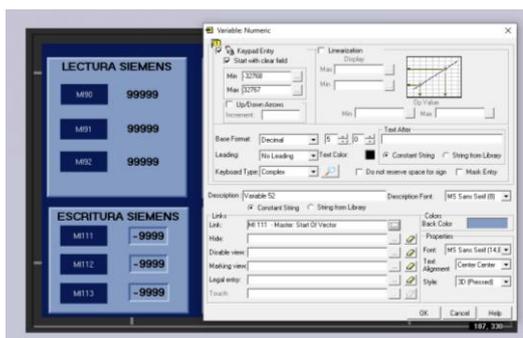
- Para la pantalla HMI Unitronics se debe crear seis divisiones con indicadores y controles numéricos, las cuales serán para Lectura y escritura en los dispositivos.



- Para configurar los indicadores numéricos HMI, se debe colocar en la pestaña “Link”, la dirección Modbus MI a leer.



- Para configurar los controles numéricos HMI se debe activar la pestaña Keypad Entry, de ser necesario se puede dar rango de entrada de valores por teclado y luego en la pestaña “Link”, Se debe colocar la dirección Modbus de escritura.



3. Programación y configuración de módulo esclavo Arduino

- **Incluir librerías necesarias en el proyecto y configurar la dirección IP del dispositivo Arduino:**

Como se puede observar, se declara la variable para MODBUS Mb, además se declara la dirección mac para identificar a Arduino como dispositivo de red y configuramos la dirección IP en la que se encontrará el dispositivo MODBUS, para este caso 192.168.0.40 como se había mencionado en el capítulo 3:

```
Practica-5 Arduino 1.8.19 (Windows Store 1.8.57.0)
Archivo Editar Programa Herramientas Ayuda
Practica-5
1 ///UNIVERSIDAD POLITECNICA SALESIANA
2 //TESIS DE GRADO DE INGENIERIA ELECTRONICA
3 //AUTOR: NICOLAS CRESPO DELGADO
4 //TEMA: PRACTICA 5: Diseño de MODBUS TCP/IP entre
5 //PLC Siemens S7-1200, PLC Unitrionics, Arduino Ethernet y PC.
6
7 //librerias
8 #include "MgsModbus.h"
9 #include <Ethernet.h>
10 #include <SPI.h>
11 #include <Wire.h>
12 #include <LiquidCrystal_I2C.h>
13
14
15 // Definición y configuración de comunicación Modbus TCP
16 //via puerto Ethernet
17 MgsModbus Mb;
18 byte mac[] = {0xFE, 0x62, 0x07, 0xDC, 0xE8, 0xC4 };
19 IPAddress ip(192, 168, 0, 40);
20 IPAddress gateway(192, 168, 0, 1);
21 IPAddress subnet(255, 255, 255, 0);
22
23 //Definición de dirección de LCD I2C, tipo de LCD
24 // Y pines de conexión de sensores ultrasonicos
25 LiquidCrystal_I2C lcd(0x27, 20, 4);
26
```

- **Declaración Variables de entradas y salidas.**

Para esta práctica se declara los potenciómetros que serán los datos variables a enviar hacia el Módulo Unitrionics mediante Modbus TCP.

Se debe declarar los pines de las luces pilotos.

Además, se declara una variable de tiempo anterior y se la inicializa en 0:

```
27 //Definición de variables y puertos IN OUT
28 int POT1;
29 int POT2;
30 int POT3;
31
32 int Piloto1 = 36;
33 int Piloto2 = 37;
34 int Piloto3 = 38;
35 int Piloto4 = 39;
36 int Piloto5 = 40;
37 int Piloto6 = 41;
38
39 unsigned long TiempoAnterior = 0;
```

- **Configuración de protocolos.**

Se programa los pines de luces piloto como salidas y se inicializan los protocolos de comunicación:

```
41 void setup() {
42   pinMode(Piloto1 , OUTPUT);
43   pinMode(Piloto2 , OUTPUT);
44   pinMode(Piloto4 , OUTPUT);
45   pinMode(Piloto3 , OUTPUT);
46   pinMode(Piloto5 , OUTPUT);
47   pinMode(Piloto6 , OUTPUT);
48   //Configuración e inicialización de protocolos
49   Ethernet.begin(mac, ip, gateway, subnet); // start ethernet int
50   Serial.println("Ethernet interface started");
51   Serial.begin(9600);
52
53   lcd.init();
54   lcd.backlight();
55
56 }
```

- **Escritura desde Arduino a Unitronics.**

El direccionamiento realizado en el PLC Unitronics usa los datos de Modbus del Mb0 al Mb20, Para esta práctica se usa M0, M1, M2, que corresponden a la lectura analógica de los potenciómetros.

También se encierra el dato de lectura en una variable para ser presentada en la pantalla LCD:

```
57
58 void loop()
59 {
60
61 //Escritura analogica en direcciones Modbus TCP
62 Mb.MbData[0] = analogRead(1);
63 Mb.MbData[1] = analogRead(2);
64 Mb.MbData[2] = analogRead(3);
65
66 POT1 = Mb.MbData[0];
67 POT2 = Mb.MbData[1];
68 POT3 = Mb.MbData[2];
69
```

- **Lectura desde Unitronics en Arduino.**

Para presentar los datos de lectura se usan las direcciones M21, M22, M23 y se imprime escribiendo el dato, tanto en LCD y en Monitor serial:

```
69
70 //LECTURA Arduino - Master - Unitronics
71 lcd.setCursor(0, 0);
72 lcd.print("LECTURA");
73
74 lcd.setCursor(0, 1);
75 lcd.print("MI21: ");
76 lcd.setCursor(6, 1);
77 lcd.print(Mb.MbData[21]);
78 Serial.print("MI21: ");
79 Serial.print(Mb.MbData[21]);
80
81 lcd.setCursor(0, 2);
82 lcd.print("MI22: ");
83 lcd.setCursor(6, 2);
84 lcd.print(Mb.MbData[22]);
85 Serial.print(" MI22: ");
86 Serial.print(Mb.MbData[22]);
87
88 lcd.setCursor(0, 3);
89 lcd.print("MI23: ");
90 lcd.setCursor(6, 3);
91 lcd.print(Mb.MbData[23]);
92 Serial.print(" MI23: ");
93 Serial.println(Mb.MbData[23]);
94
```

- **Visualización de valores de escritura en pantalla LCD.**

Los datos que se encerraron en variables POT1, POT2, POT3, son presentados en la pantalla LCD de la siguiente manera:

```
95 //ESCRITURA Arduino - Master - Unitronics
96 lcd.setCursor(10, 0);
97 lcd.print("ESCRITURA");
98
99 lcd.setCursor(10, 1);
100 lcd.print("MI0: ");
101 lcd.setCursor(15, 1);
102 lcd.print(POT1);
103
104 lcd.setCursor(10, 2);
105 lcd.print("MI1: ");
106 lcd.setCursor(15, 2);
107 lcd.print(POT2);
108
109 lcd.setCursor(10, 3);
110 lcd.print("MI2: ");
111 lcd.setCursor(15, 3);
112 lcd.print(POT3);
113
```

- **Actualización de datos en pantalla LCD.**

Se usa la función millis() que permite al Arduino ejecutar un retraso de tiempo para ejecutar otra acción sin interrumpir el programa principal, esta acción permite que cada 900 milisegundos la pantalla LCD se limpie, permitiendo actualizar los datos de lectura y escritura para su visualización, luego se debe ejecutar la función Mb.MbsRun() quien permite al Arduino funcionar como esclavo Modbus TCP.

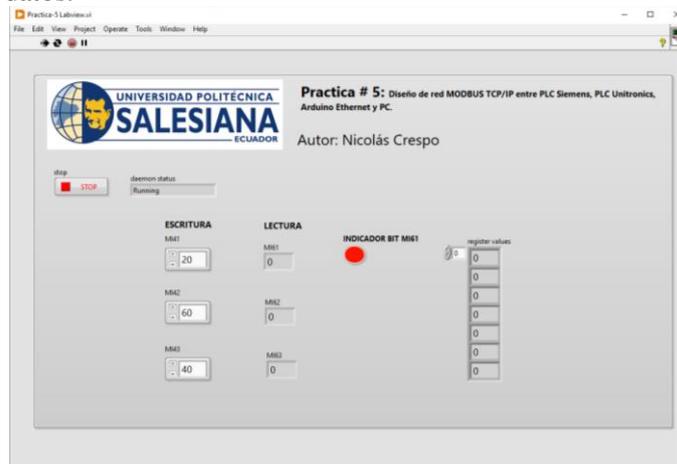
```

114
115 if (millis() - TiempoAnterior > 900) {
116     TiempoAnterior = millis();
117     lcd.clear();
118 }
119
120
121 Mb.MbsRun();
122
123 }

```

4. Desarrollo en Labview.

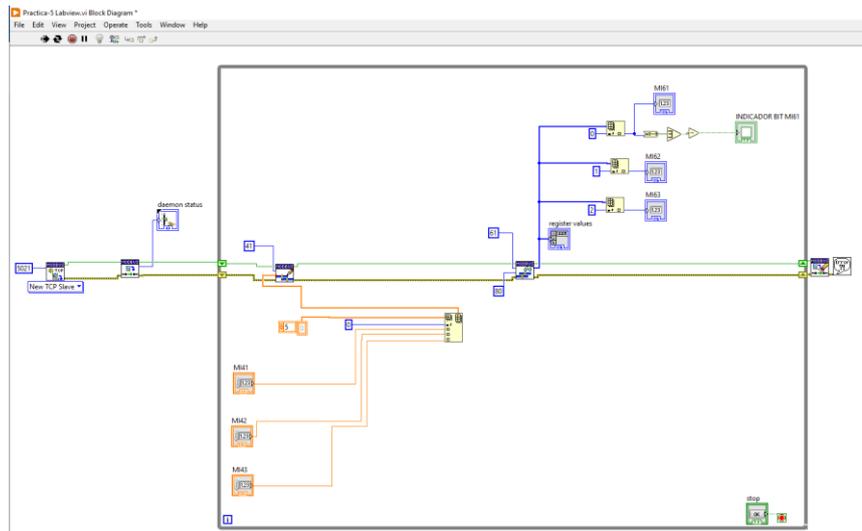
- Se realiza la pantalla colocando 3 controles de lectura y 3 indicadores de escritura, también un indicador LED que será solo para visualización cuando el dato Modbus sea tipo byte, es decir 1 o 0, también se coloca un indicador Array para visualización de los datos.



- En el diagrama de bloques la configuración creamos una instancia Modbus TCP esclavo, que permite usar la red de la computadora como red para comunicación Modbus TCP, para ello se debe configurar el dispositivo de red en la dirección IP: "192.168.0.3", para esta práctica.
- Tal como se realizó la práctica número 3, en el bloque de escritura se debe ingresar la dirección de inicio de escritura Modbus configurada en el PLC Unitronics, que es la dirección M41, luego se construye el array de datos a enviar que contiene los controles de escritura con el bloque llamado "Insert into Array Function".
- Para el bloque de lectura se coloca el rango de inicio de lectura y final de lectura de direcciones Modbus, en este caso, en el PLC Unitronics se configuro desde la dirección M61 hasta M80. Luego

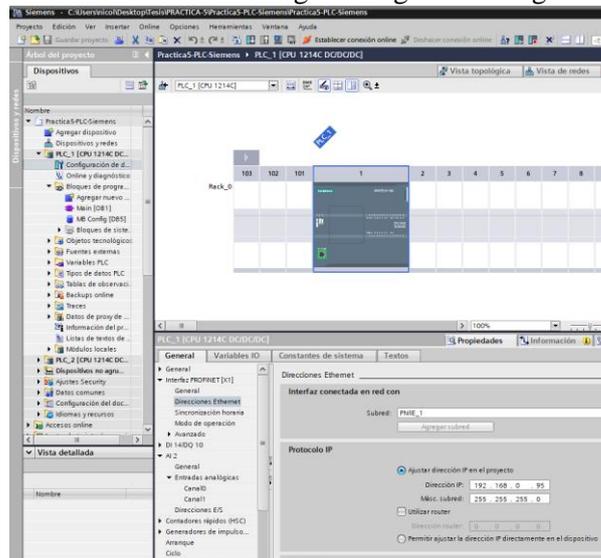
se usa el bloque “Index Array Function”, que permite cambiar el tamaño del arreglo de datos, a una dirección única deseada para poder manipular el dato como un dato entero en el programa.

- Para poder convertir el dato entero de lectura a un dato booleano, se usa el bloque “Number to boolean Array” que devuelve un número a un array booleano, luego en el bloque “Or Array Elements Function”, se retorna el dato booleano Array a un dato lógico de verdadero y falso. (Revisar práctica 3)



5. Desarrollo en Siemens

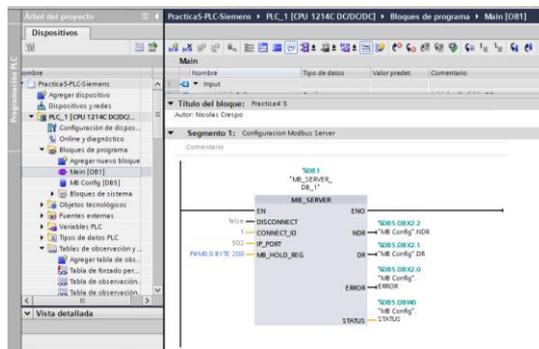
Se debe crear un proyecto con el PLC S7-1200 en TIA Portal, luego se configura la dirección IP del PLC según la siguiente imagen:



Luego se crea un Bloque de datos global, estos enlaces de datos sirven para tener visualización de la información de conexión Modbus en el bloque de Modbus Server de Siemens.

Nombre	Tipo de datos	Offset	Valor de amanq.	Remanen.	Accesible d...	Escrib...	Visible en...	Valor de s...	Comenta
STATUS	Word	0.0	16#0		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ERROR	Bool	2.0	false		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
DR	Bool	2.1	false		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
NR	Bool	2.2	false		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
CONNECT	TCOUP_4	4.0			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

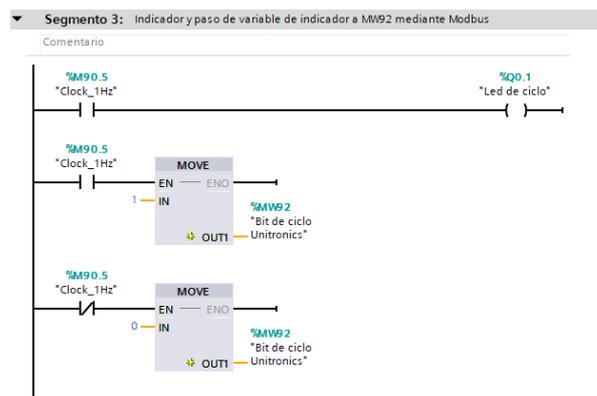
En el programa principal o Main colocamos y configuramos el bloque de comunicación MB_SERVER de la siguiente manera, donde MB_HOLDING_REG es el búfer de memoria (M) que usa ANY que es un puntero que según el patrón: P#dirección_de_bit "Tipo de datos" "Longitud" por ejemplo: P#M1000.0 WORD 500, para Lectura – Escritura que se tendrá para interactuar con las variables en los demás dispositivos:



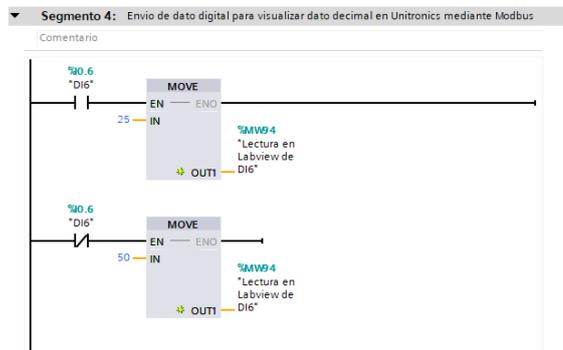
Uno de los datos a enviar desde el PLC Siemens en la dirección M90 es la lectura analógica IW66, que debe ser normalizada y escalada de la siguiente manera:



El siguiente dato será enviar un bit que corresponde a la señal de reloj interna de 1Hz del PLC y que su valor será enviado a la variable MW92 con el bloque Move:



De la misma manera, se envía un dato digital de la entrada I0.6, enviando valores decimales a la variable Modbus MW94.



Recordar que el tipo de datos que se utiliza es de Tipo Word, lo que permitirá compartir los datos de manera correcta entre dispositivos, además que el direccionamiento Modbus en Siemens es diferente, al usar siempre direcciones MW pares debido al tamaño de buffer que usa el CPU para direccionamiento de variables.

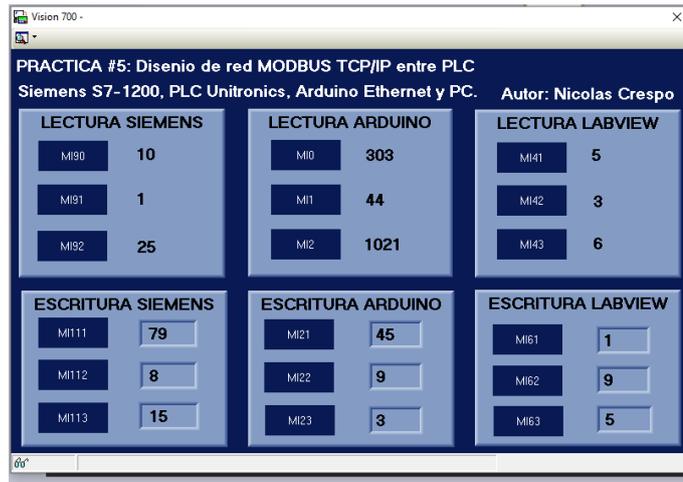
ACTIVIDADES POR DESARROLLAR

1. Acceder al PLC esclavo con un programa de PC o de celular de Modbus TCP, se lo puede obtener desde Windows como “Modbus Poll” o en Android como “Poll Modbus”.
2. Contemplar fallas en las conexiones TCP.
3. Enviar variables desde PC hasta Arduino, Siemens, o alternando el método de envío de datos entre dispositivos usando traspaso de variables de una dirección a otra dirección desde PLC Unitronics usando bloque Store.

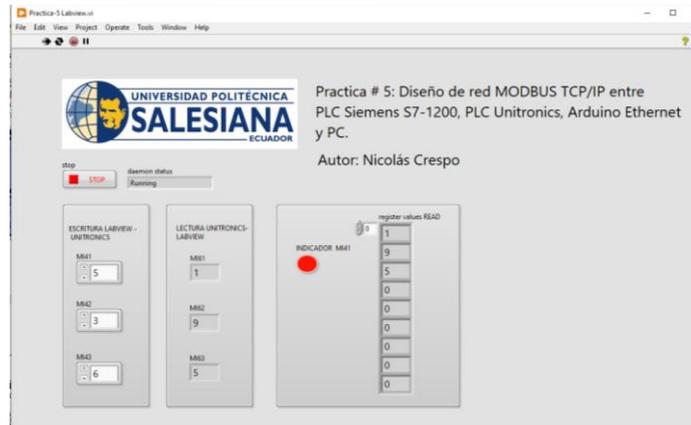
RESULTADOS OBTENIDOS:

Mediante LabView recibimos los datos de Unitronics, de Arduino, y del PLC Siemens sin problema y sin retraso, además que se pudo interactuar entre los tres. Así mismo, se leyeron datos de entrada y salida de los selectores de los tableros de prácticas.

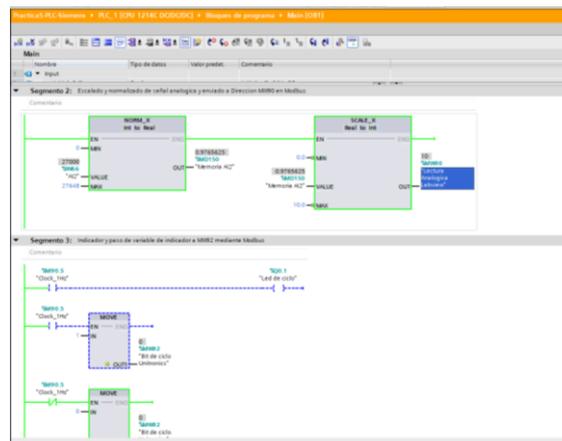
Pantalla HMI Unitronics:

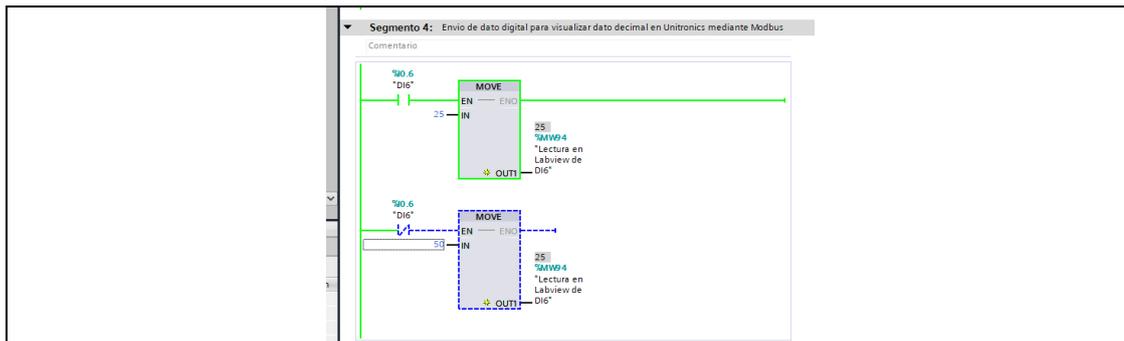


Pantalla Labview – PC



Bloques en funcionamiento y tabla de visualización de variables en PLC Siemens:





Según la tabla de observación podemos visualizar los 3 primeros datos que se escriben desde PLC Siemens y se pueden ver en pantalla HMI Unitronics, también desde MW180 se observa la lectura de la dirección M111 hasta M113 desde PLC Unitronics.

Prácticas PLC Siemens - PLC 1 (CPU 1214C-DC/DC/DC) - Tablas de observación y forzado

Nombre	Dirección	Formato visualiza.	Valor de observac.	Valor de forzado
"Lectura Analógica M0"	%MW80	DEC+/-	7	
"Led de ciclo M91"	%MW82	DEC+/-	1	
"Entrada digital M92"	%MW84	DEC+/-	25	
	%MW86	DEC+/-	0	
	%MW88	DEC+/-	0	
	%MW90	DEC+/-	0	
	%MW92	DEC+/-	0	
	%MW94	DEC+/-	0	
	%MW96	DEC+/-	0	
	%MW98	DEC+/-	0	
	%MW00	DEC+/-	0	
	%MW02	DEC+/-	0	
	%MW04	DEC+/-	0	
	%MW06	DEC+/-	0	
	%MW08	DEC+/-	0	
	%MW10	DEC+/-	0	
	%MW12	DEC+/-	0	
	%MW14	DEC+/-	0	
	%MW16	DEC+/-	0	
	%MW18	DEC+/-	0	
	%MW20	DEC+/-	0	
	%MW22	DEC+/-	79	
	%MW24	DEC+/-	8	
	%MW26	DEC+/-	15	
	%MW28	DEC+/-	0	
	%MW30	DEC+/-	0	
	%MW32	DEC+/-	0	
	%MW34	DEC+/-	0	

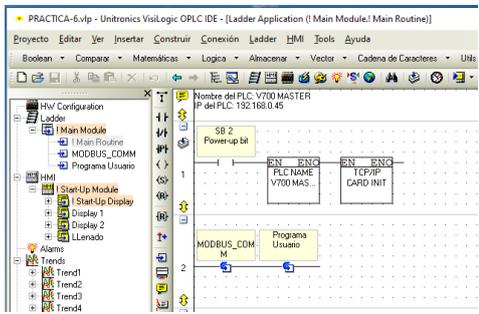
CONCLUSIONES: Se implementó una comunicación Modbus TCP, donde el PLC Unitronics configurado como PLC Master, recibe las conexiones de los dispositivos Arduino, PC Labview y PLC S7 1200, esta práctica servirá para futuras implementaciones con varios dispositivos en red Modbus TCP.

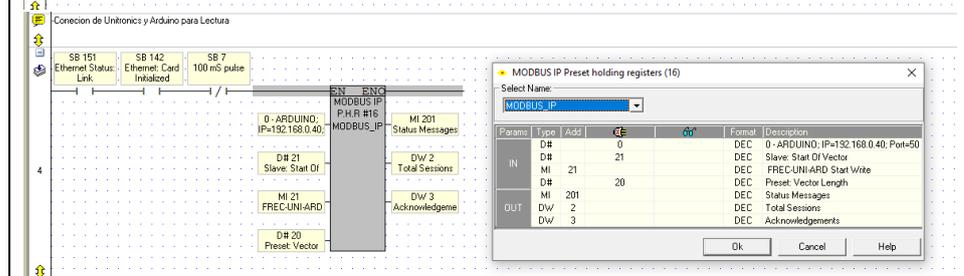
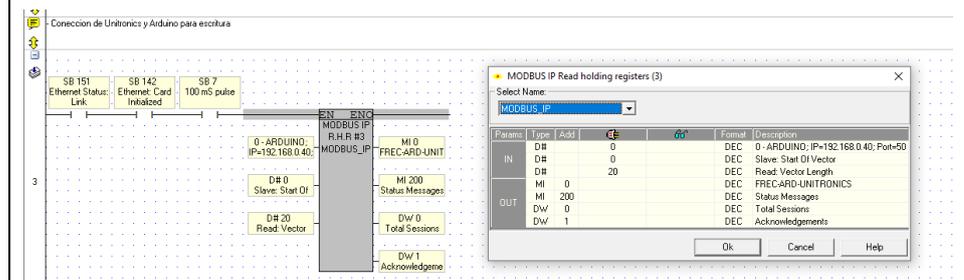
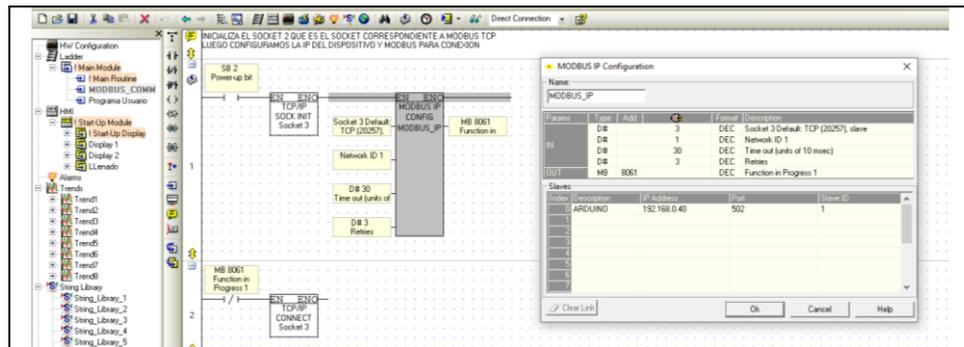
- RECOMENDACIONES:**
- Verificar que la dirección IP de la PC este en el mismo rango de direcciones configuradas en las antenas previamente.
 - Podemos verificar mediante una aplicación que lea los datos de Modbus mediante PLC Siemens que envía los datos.
 - Verificar mediante los indicadores Led de las antenas que existe una conexión entre estas.
 - Revisar si la dirección de red de la PC, es la misma configurada en el PLC master para lectura y escritura Labview.
 - Para probar el tablero cliente se recomienda descargar una aplicación que pueda escanear las direcciones Modbus para verificar la conexión.

Docente/Técnico Docente: _____

Firma: _____

4.6 Práctica 6

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados	
NRO. PRÁCTICA:	06	TÍTULO PRÁCTICA: Arranque de motor variando su velocidad desde estación Arduino Ethernet y Monitoreo desde HMI Unitronics.	
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS TCP/IP en PLC Unitronics y Arduino. 2. Arrancar el motor trifásico conectado a PLC Unitronics mediante Arduino 3. Crear control de marcha, paro de emergencia y falla de sistema mediante Arduino. 4. Crear una visualización grafica en HMI Unitronics. 			
DEV:	1. Introducción: El programa desarrolla una comunicación entre un maestro y un esclavo (Cliente y Servidor), el módulo Unitronics será el Maestro Modbus TCP/IP a la que se conectará el módulo Esclavo Arduino. Se debe desarrollar controles de mando de motor tanto remoto y local, Paros de emergencia y cambio de giro de motor.		
	2. Desarrollo en PLC Unitronics <ul style="list-style-type: none"> ● Configuración: Donde estará el nombre del PLC en este caso V700 MASTER y la dirección IP del dispositivo: 192.168.0.45 <div style="text-align: center;">  </div> <ul style="list-style-type: none"> ● Configuración Modbus: Donde está la configuración de Modbus Master del PLC y la conexión con el esclavo Arduino ● Cuando la señal de Pulso SB7 sea verdadera se activará este bloque, que consiste en leer los registros enviados desde Arduino ● Se configura la IP con la de Arduino: 192.168.0.40 ● Para este caso se leerá desde el vector 0 hasta el Vector 20 que son las direcciones Modbus IP ● El bloque empezará a leer las direcciones desde MIO. ● SB7 Es una señal de Pulso de 100 ms que estará en contacto abierto 		

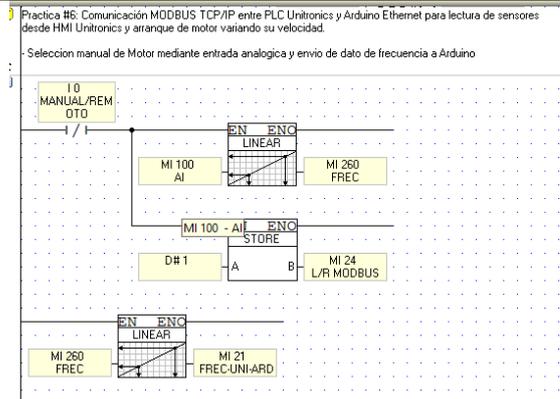


Programa de usuario:

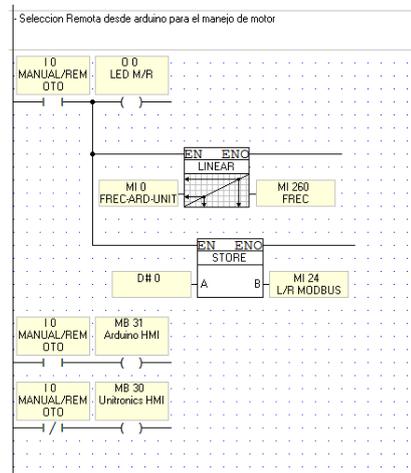
- La 5entrada IO Seleccionará entre control de motor manual o remoto desde Arduino.
- MI100 es la variable enlazada a AI del PLC que tiene valores de 0 a 1020 y se linealiza para pasar de 0 a 4098 que se enlazo a la variable MI260.
- AI Tiene una resolución de 10 bits y se lee de 0 a 1020 y con voltaje de entrada 0 a 10v
- AO Tiene una resolución de 12bits y se lee de 0 a 4098 y con voltaje de salida de 0 a 10v
- La salida analógica de frecuencia, señal MI260, se escala para poder visualizarla en valores de 0 a 60 Hz
- La señal de IO se envía a la dirección MI24 que es el indicador de MANUAL/REMOTO de operación del motor, se usa el bloque STORE quien permite escribir un valor contenido en un operando o constante a otro operando.

La entrada IO Seleccionará entre control de motor manual o remoto desde Arduino

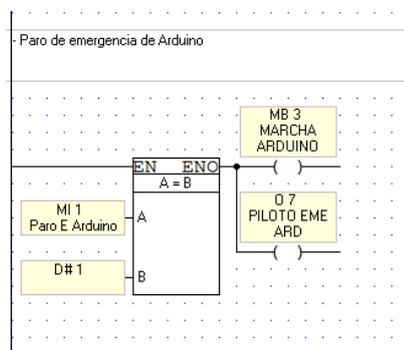
- MI100 es la variable enlazada a AI del PLC que tiene valores de 0 a 1020 y se linealiza para pasar de 0 a 4098 que se enlazo a la variable MI260
- AI Tiene una resolución de 10 bits y se lee de 0 a 1020 y con voltaje de entrada 0 a 10v
- AO Tiene una resolución de 12bits y se lee de 0 a 4098 y con voltaje de salida de 0 a 10v



- Por otro lado, cuando se selecciona la forma remota de operación, el dato de entrada a ser linealizado proviene de la dirección Modbus MI100 quien recibirá el dato con valor de 0 a 60 (Hz) y se linealiza para pasar el dato de 12 bits de 0 a 4098 (unidades) Con ajuste, que se escribe en la salida analógica A00 y genera la frecuencia del motor.

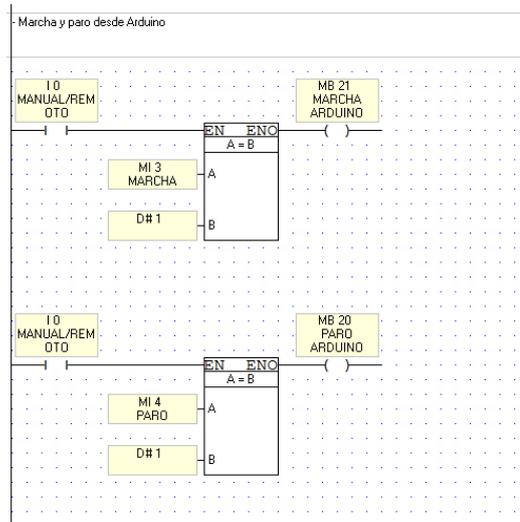


- Control de paro de emergencia desde Arduino, se activa el paro de emergencia y datos que vienen de Modbus, estos datos Enteros son comparados con el bloque de comparación A=B.

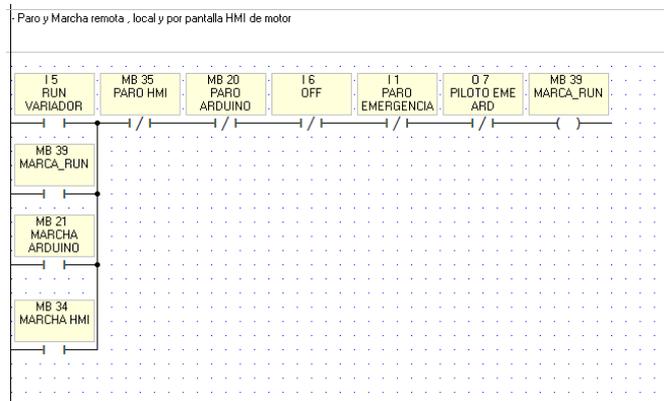


- Control de paro y marcha desde Arduino, los siguientes bloques muestran que una vez que se determina el control ya sea manual o remoto. Se recibe la señal es que comparada en un bloque de

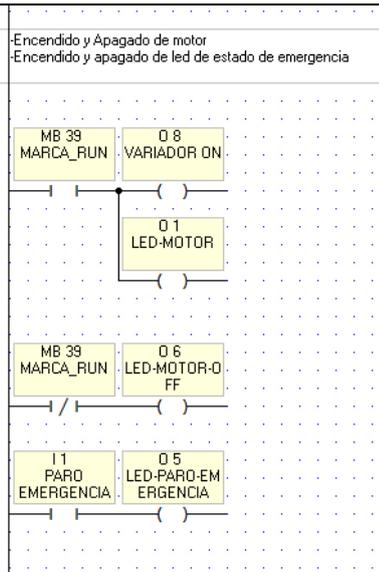
comparación $A=B$, para que ocurra la acción de marcha y paro del sistema.



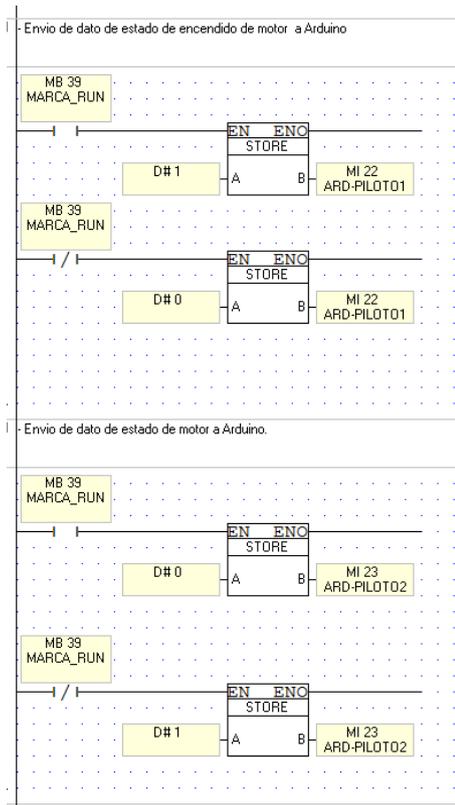
- El siguiente bloque será para activar o desactivar el variador consta de contactos abiertos y cerrados que pertenecen a fallas, paro de emergencia y datos que vienen de Modbus, estos datos Enteros son comparados con el bloque de comparación $A=B$ para activar una bobina de activación o desactivación que se encargará de la acción remota del variador.



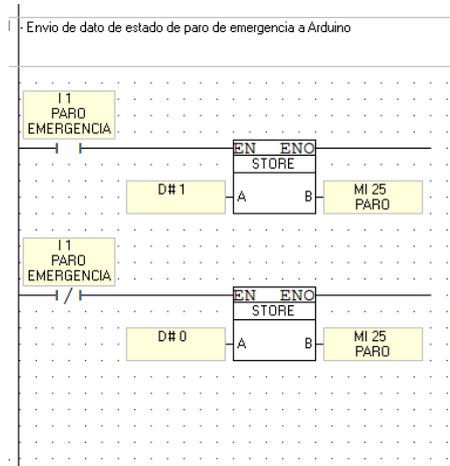
- Para este bloque se muestra la entrada MB39 que permite el encendido o apagado del motor. Además, sirve como indicador Led para observar el estado de emergencia.



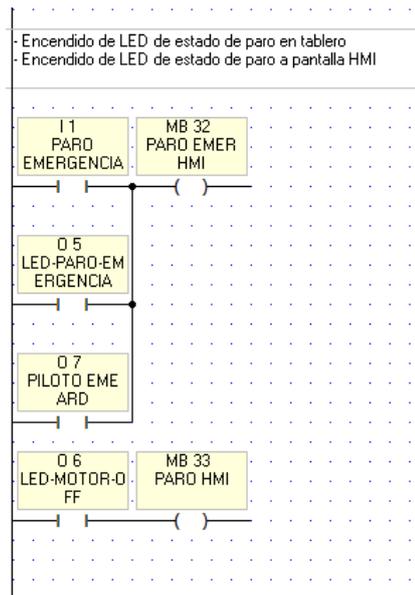
- A continuación, se muestra el envío de datos sobre el estado de encendido de motor hacia Arduino, ya sea que esté en estado 0 o 1.



- Similarmente, se muestra el envío de datos sobre el estado del paro de emergencia a Arduino, ya sea que esté en estado 0 o 1; apagado o encendido, respectivamente.



- El siguiente bloque contiene los indicadores LED sobre el estado de emergencia ya sea desde el tablero o desde la pantalla HMI.



- Cambio de giro de motor manual y remoto desde Arduino.

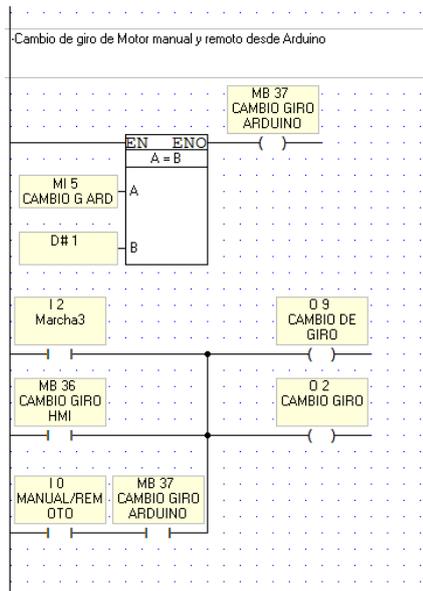


Tabla de Direcciones Modbus en PLC Unitronics desde programador.

Tabla lectura

	Dpr	Addr	Use	Format	Description
I	Inputs				
O	Outputs	MI 0	<input checked="" type="checkbox"/>	DEC	FREC-ARD-UNITRONICS
T	Timers	MI 1	<input checked="" type="checkbox"/>	DEC	Paro E Arduino
MB	Memory Bits	MI 2	<input type="checkbox"/>	DEC	
MI	Memory Integer	MI 3	<input checked="" type="checkbox"/>	DEC	MARCHA ARDUINO
ML	Memory Long	MI 4	<input checked="" type="checkbox"/>	DEC	PARO
DW	Double Word	MI 5	<input checked="" type="checkbox"/>	DEC	CAMBIO G ARD
XB	X Bits	MI 6	<input type="checkbox"/>	DEC	
XI	X Integer	MI 7	<input type="checkbox"/>	DEC	
XL	X Long	MI 8	<input type="checkbox"/>	DEC	
XDW	X Double Word	MI 9	<input type="checkbox"/>	DEC	
MF	Memory Float	MI 10	<input type="checkbox"/>	DEC	
SB	System Bits	MI 11	<input type="checkbox"/>	DEC	
SI	System Integer	MI 12	<input type="checkbox"/>	DEC	
SL	System Long	MI 13	<input type="checkbox"/>	DEC	
SDW	System Double Word	MI 14	<input type="checkbox"/>	DEC	
C	Counters	MI 15	<input type="checkbox"/>	DEC	
#	Signed Constants	MI 15	<input type="checkbox"/>	DEC	

Tabla escritura

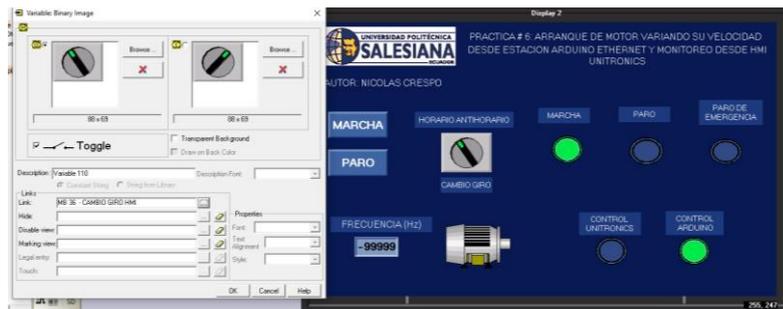
	Dpr	Addr	Use	Format	Description
I	Inputs				
O	Outputs	MI 20	<input type="checkbox"/>	DEC	
T	Timers	MI 21	<input checked="" type="checkbox"/>	DEC	FREC-UNI-ARD Start Write
MB	Memory Bits	MI 22	<input checked="" type="checkbox"/>	DEC	ARD-PILOT01
MI	Memory Integer	MI 23	<input checked="" type="checkbox"/>	DEC	ARD-PILOT02
ML	Memory Long	MI 24	<input checked="" type="checkbox"/>	DEC	L/R MODBUS
DW	Double Word	MI 25	<input checked="" type="checkbox"/>	DEC	PARO EMERGENCIA UNITRONICS
XB	X Bits	MI 26	<input type="checkbox"/>	DEC	
XI	X Integer	MI 27	<input type="checkbox"/>	DEC	
XL	X Long	MI 28	<input type="checkbox"/>	DEC	
XDW	X Double Word	MI 29	<input type="checkbox"/>	DEC	
MF	Memory Float	MI 30	<input type="checkbox"/>	DEC	
SB	System Bits	MI 31	<input type="checkbox"/>	DEC	
SI	System Integer	MI 32	<input type="checkbox"/>	DEC	
SL	System Long	MI 33	<input type="checkbox"/>	DEC	
SDW	System Double Word	MI 34	<input type="checkbox"/>	DEC	
C	Counters	MI 34	<input type="checkbox"/>	DEC	

Programación de pantalla HMI

- Cuando programamos el botón de marcha o paro, solo debemos enlazar la variable que anteriormente fue descrita como Marcha HMI, y que cuya dirección es MB34, de este modo continuamos con las siguientes.



- Para programación de imágenes binarias, se selecciona una imagen que represente a estado booleano en "0" o en "1", luego se enlaza el link con la dirección de memoria de bits para este caso MB36 que dará la orden de cambio de giro desde HMI, y se continúa realizando el mismo procedimiento para las imágenes animadas de nuestro programa.



- Para mostrar un dato de una variable, solo enlazamos el link con el dato de Memory Integer correspondiente (MI), en este caso leeremos la frecuencia.



3. Desarrollo en Arduino:

- MgsModbus será la librería que contiene la configuración y puesta en marcha de Modbus TCP/IP para Arduino.
- LiquidCrystal_I2C permitirá la conexión con el LCD mediante protocolo I2C la cual nos permitirá usar solo dos cables de conexión y ahorrar espacio en entradas o salidas físicas de Arduino.
- Ethernet y SPI: Serán las que permitirán la conexión TCP/IP mediante el Shield Ethernet.
- Se asigna la dirección HEX por defecto 0x27 del I2C para el LCD y el tipo de LCD para este caso con dimensiones de 20x4.
- Se realiza la configuración de direcciones Ethernet y la MAC del dispositivo, además de declarar la variable de Mb para Modbus.

```

Practica6 Arduino 1.8.19 (Windows Store 1.8.57.0)
Archivo Editar Programa Herramientas Ayuda

Practica6
1  ///UNIVERSIDAD POLITECNICA SALESIANA
2  ///TESIS DE GRADO DE INGENIERIA ELECTRONICA
3  ///AUTOR: NICOLAS CRESPO DELGADO
4  ///TEMA: PRACTICA 6: Arranque de motor variando su velocidad
5  ///desde estacion Arduino Ethernet y Monitoreo desde HMI Unitronics
6
7  //Librerias
8  #include "MgsModbus.h"
9  #include <LiquidCrystal_I2C.h>
10 #include <Ethernet.h>
11 #include <SPI.h>
12
13 //Definicion y direccionamiento de de Pantalla LCD I2C
14 //como pantalla de 20x4
15 LiquidCrystal_I2C lcd(0x27, 20, 4);
16
17 // Definicion y configuracion de comunicacion Modbus TCP
18 //via puerto Ethernet
19 MgsModbus Mb;
20 byte mac[] = {0xFE, 0x62, 0xC7, 0xDC, 0xE8, 0xC4 };
21 IPAddress ip(192, 168, 0, 40);
22 IPAddress gateway(192, 168, 0, 1);
23 IPAddress subnet(255, 255, 255, 0);
24

```

- Finalmente se declaran las variables y se asignan los pines de entradas y salidas, para esto ver la tabla de entradas y salidas de Arduino en el Anexo: Arduino.

```
24  
25 //Definicion de variables  
26 int Selector1 = 30;  
27 int Selector2 = 31;  
28 int Selector3 = 32;  
29 int Sel1;  
30 int Sel2;  
31 int Sel3;  
32  
33 int PinEmergencia = 29;  
34 int ParoEmergencia;  
35  
36 int Piloto1 = 36;  
37 int Piloto2 = 37;  
38 int Piloto3 = 38;  
39 int Piloto4 = 39;  
40 int Piloto5 = 40;  
41 int Piloto6 = 41;  
42 int Potenciometrol;  
43 int Frecuencia;  
44
```

Configuración inicial del programa:

- Se inicializa el LCD, se enciende la luz de fondo y se escribe un mensaje colocando el cursor en la posición 1,0.
- Se inicializa el puerto serial y el Ethernet, colocando la dirección mac, IP, Gateway, subnet, anteriormente ya declaradas.
- Se establecen los pines de entrada y salida, Output para salidas digitales e Input para entradas digitales.
- Se inicializan las salidas Piloto 3 en '0' o falso.
- Se configura PID mode en Automático.

```

45 void setup() {
46 |
47 //Configuracion e inicializacion de protocolos
48 //Configuracion de puertos de entrada y salidas digitales
49 lcd.init();
50 lcd.backlight();
51 lcd.setCursor(1, 0);
52 lcd.print("Modulo 3 - Arduino");
53
54 Serial.begin(9600);
55 Serial.println("Serial interface started");
56 Ethernet.begin(mac, ip, gateway, subnet);
57 Serial.println("Ethernet interface started");
58
59 pinMode(Selector1 , INPUT);
60 pinMode(Selector2 , INPUT);
61 pinMode(Selector3 , INPUT);
62 pinMode(PinEmergencia , INPUT);
63 pinMode(Piloto1 , OUTPUT);
64 pinMode(Piloto2 , OUTPUT);
65 pinMode(Piloto4 , OUTPUT);
66 pinMode(Piloto3 , OUTPUT);
67 pinMode(Piloto5 , OUTPUT);
68 pinMode(Piloto6 , OUTPUT);
69 pinMode(pinzumbador , OUTPUT);
70
71 }

```

- Lectura del dato de Modbus Unitronics para control Local/Remoto, y mensaje en pantalla LCD como indicador del estado.
- Se reciben los datos enteros de la dirección Modbus 21 y 24, el primero corresponde al dato de la frecuencia del motor que se escribe desde Unitronics, y el segundo es el estado de escritura de la frecuencia (Mando de Motor) ya sea desde Arduino o Desde Unitronics.
- Se crean las condiciones para seleccionar el mensaje de mando de motor en el LCD: PLC Unitronics – Arduino.
- lcd.setCursor establece el cursor en la posición: (columna, fila) y con print se escriben los mensajes a visualizar.
- Se crea una condición para que el dato de la frecuencia no se sobrescriba y se actualice el valor correctamente, como ejemplo; sin esta condición cuando el dato sea 10 y luego 9 se visualiza de esta forma: ‘90’ para esto la condición dice que cuando el valor sea < a 10 el cursor se ubique una unidad más a la derecha escribiendo el valor como: ‘09’. Usamos esta condición como técnica de actualización de mensaje de LCD, la más común es usar la función lcd.clear().

```

73 void loop() {
74
75 //Lectura de Modbus Unitronics switch Local/Remoto
76 int data = Mb.MbData[21];
77 int data2 = Mb.MbData[24];
78
79 // Mensajes en pantalla LCD
80 lcd.setCursor(0, 1);
81 lcd.print("Mando Motor:");
82
83 if (data2 == 1) {
84     digitalWrite(Piloto2 , 0);
85     lcd.setCursor(13, 1);
86     lcd.print("PLC Uni");
87 } else {
88     digitalWrite(Piloto2, 1);
89     lcd.setCursor(13, 1);
90     lcd.print("Arduino");
91 }
92
93 lcd.setCursor(0, 2);
94 lcd.print("Frec. Unitronics:");
95
96 lcd.setCursor(18, 2);
97 lcd.print(data);
98
99 if (data < 10) {
100     lcd.setCursor(18, 2);
101     lcd.print("0");
102     lcd.setCursor(19, 2);
103     lcd.print(data);
104 }
105
106 lcd.setCursor(0, 3);
107 lcd.print("Frec. Arduino:");
108 lcd.setCursor(18, 3);
109 lcd.print(Frecuencia);
110

```

- Para la lectura analógica de Arduino se utiliza la línea de código: `analogRead(AI)`, y luego se escribe el dato a la dirección Modbus que corresponda.
- La función “map” permite linealizar un número de un rango a otro y convertirá según el valor de lectura analógica de Arduino con resolución de 10bits (0-1023), para este caso el valor de salida será de 0 a 60 con unidad de Hz que será enviada al PLC Unitronics mediante Modbus en la dirección 7. Como se menciona en la práctica anterior, la sintaxis de la función map es: `map(value, fromLow, fromHigh, toLow, toHigh)`.
- Se asigna una variable a las lecturas digitales mediante el código `digitalRead()`

```

--
118 //Lecturas Analógicas y envios de datos Modbus a PLC
119 Potenciometro1 = analogRead(A1);
120 Frecuencia = map(Potenciometro1, 0, 1023, 0, 60);
121 Mb.MbData[0] = Frecuencia;
122
123 //Lectura de entradas digitales
124 Sel1 = digitalRead(Selector1);
125 Sel2 = digitalRead(Selector2);
126 Sel3 = digitalRead(Selector3);
127 ParoEmergencia = digitalRead(PinEmergencia);
128

```

- Establecemos las condiciones básicas del programa que es indicar si un valor toma el estado de 1 o 0, en este caso HIGH o LOW, y ese valor enviarlo al PLC Unitronics mediante Modbus en las direcciones puestas en el programa, también se puede indicar si una variable o también la otra pueden hacer que un valor tome el estado de 1 o 0.

```

129 //Paro de emergencia Arduino
130 if (ParoEmergencia == HIGH) {
131     Mb.MbData[1] = 0;
132     digitalWrite(Piloto6 , 0);
133 } else {
134     digitalWrite(Piloto6 , 1);
135     Mb.MbData[1] = 1;
136 }
137
138 //Marcha y paro de motor
139 if (Sel1 == HIGH) {
140     Mb.MbData[3] = 1;
141
142 } else {
143     Mb.MbData[3] = 0;
144 }
145
146 if (Sel2 == HIGH) {
147     Mb.MbData[4] = 1;
148
149 } else {
150     Mb.MbData[4] = 0;
151 }
152
153
154 // Inversion de giro
155 if (Sel3 == HIGH) {
156     digitalWrite(Piloto3 , 1);
157     Mb.MbData[5] = 1;
158 } else {
159     digitalWrite(Piloto3 , 0);
160     Mb.MbData[5] = 0;
161 }
162
163

```

- Por último, se lee el dato Modbus desde Unitronics usando la función bitRead(), para escribir el valor de True or False en la función de digitalWrite(). Estos dos se visualizarán como luces piloto.
- Mb.MbsRun() define el tipo de conexión Modbus TCP/IP lo que significa que esta conexión es de tipo Modbus Server.

```

163
164 //Encendido de indicadores de estado de motor
165 digitalWrite(Piloto1 , bitRead( Mb.MbData[22], 0));
166 digitalWrite(Piloto4 , bitRead( Mb.MbData[23], 0));
167 digitalWrite(Piloto5 , bitRead( Mb.MbData[25], 0));
168
169 //Configuracion de Modbus como Esclavo
170 Mb.MbsRun();
171 }

```

ACTIVIDADES POR DESARROLLAR

1. Realizar conexiones para obtener más variables físicas desde PLC Unitronics para profundizar en el análisis con parámetros relacionados con las aplicaciones.
2. Dar seguimiento a todas las variables usando el test Online test para tener un acceso más rápido de la lectura y escritura de variables cuando se esté en fase de pruebas.
3. Verificar el correcto funcionamiento del motor y contemplar fallas en la conexión TCP.

RESULTADOS OBTENIDOS:

Se pudo obtener el control de marcha y paro desde HMI Unitronics por medio de interruptores, comparado con el sistema configurado con Arduino que toma más tiempo. Además, como se muestra en la figura debajo se puede apreciar la interfaz de HMI en funcionamiento del sistema, se obtuvo el cambio de giro de motor, muestra en pantalla el valor de la frecuencia del motor en Hz. Además, de los indicadores LED que muestra el estado de marcha, paro, paro de emergencia, y el tipo de control ya sea por medio de Unitronics o Arduino.



CONCLUSIONES:

Este trabajo mostró un sistema de comunicación entre el cliente, Modbus TCP/IP, conectado a un servidor de Arduino, donde de manera remota mediante un tablero de HMI Unitronics recibe la dirección IP de Arduino, donde se logró controlar marcha y paro del motor trifásico. Además, se pudo controlar la velocidad del motor desde Arduino o desde el PLC Unitronics, este material para uso como material didáctico para estudiantes entiendan el control de paro y marcha y demás elementos básicos de control de motor.

RECOMENDACIONES:

- Considerar colocar alarmas visuales de fallo de sistema
- Para probar el tablero cliente se recomienda descargar una aplicación que pueda escanear las direcciones Modbus para verificar la conexión.
- Verificar que la dirección IP tanto del Arduino y PLC Unitronics estén en el mismo rango de direcciones configuradas en las antenas previamente.
- Antes de encender los equipos asegurarse que todo está apagado, y en primer lugar energizar el variador de frecuencia, luego verificar que no exista falso contacto o que el enchufe de conexión no se encuentre flojo al momento de conectar.

Docente/Técnico Docente: _____

Firma: _____

4.7 Práctica 7

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados / Teoría de control	
NRO. PRÁCTICA:	07	TÍTULO PRÁCTICA: Simulación de llenado de tanque usando el controlador Arduino Ethernet y su librería PID para control de motor mediante variador de velocidad.	
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS TCP/IP entre PLC Unitronics y Tablero Arduino. 2. Crear un control PID para llenado de tanque controlado desde Tablero Arduino. 3. Interactuar con variables y graficas de control desde PLC Unitronics. 4. Controlar motor variando su velocidad dada la respuesta de control desde tablero Arduino. 			
Dev:	1. Introducción: El programa desarrolla un control PID de llenado de tanque que es desarrollado en el tablero Arduino, el PLC Unitronics leerá las variables como: Setpoint, Input, Output, Kp, Ki, Kd, entre otras, y se ajustará la variable de salida para poder controlar el Motor mediante variador de velocidad simulando que este es quien funciona como bomba centrífuga, cabe recalcar que los dos motores conectados a Arduino son de 12VDC y serán controlados desde tablero Arduino. Considerar colocar un accionador de descarga de tanque, y un accionador de activación de perturbación para probar la estabilidad del control PID.		
	2. Desarrollo en Arduino: <ul style="list-style-type: none"> ● MgsModbus será la librería que contiene la configuración y puesta en marcha de Modbus TCP/IP para Arduino. ● LiquidCrystal_I2C permitirá la conexión con el LCD mediante protocolo I2C la cual nos permitirá usar solo dos cables de conexión y ahorrar espacio en entradas o salidas físicas de Arduino. ● Ethernet y SPI: Serán las que permitirán la conexión TCP/IP mediante el Shield Ethernet. ● NewPing: librería que contiene la configuración de lectura de los sensores ultrasónicos. ● PID: librería que contiene la configuración que permitirá el control cuando ocurran errores o perturbaciones en el sistema. ● Math: esta librería permite obtener funciones matemáticas. <pre> Practica7.s 1 ///UNIVERSIDAD POLITECNICA SALESIANA 2 //TESIS DE GRADO DE INGENIERIA ELECTRONICA 3 //AUTOR: NICOLAS CRESPO DELGADO 4 //TEMA: PRACTICA 7: Simulación de llenado de tanque usando el 5 //controlador Arduino Ethernet y su librería PID para control 6 //de motor mediante variador de velocidad. 7 8 //Librerías 9 #include "MgsModbus.h" 10 #include <LiquidCrystal_I2C.h> 11 #include <Ethernet.h> 12 #include <SPI.h> 13 #include <PID_v1.h> 14 #include <math.h> 15 #include <NewPing.h> 16 </pre>		

Definición de pines, configuraciones y asignación de variables:

- Se define el bus por el cual se van a conectar los sensores HC-SR04, el pin ECHO y TRIGGER del sensor ultrasónico, la máxima distancia que tiene el sensor ultrasónico como rango, y luego se configura el sonar con las variables ya declaradas.
- Se asigna la dirección HEX por defecto 0x27 del I2C para el LCD y el tipo de LCD para este caso con dimensiones de 20x4.
- Se realiza la configuración de direcciones Ethernet y la MAC del dispositivo, además de declarar la variable de Mb para Modbus.

```
16
17 //Definición de pines y configuración de sensor Ultrasonico
18 #define TRIGGER_PIN 16
19 #define ECHO_PIN 15
20 #define MAX_DISTANCE 200
21 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
22
23 //Definición y direccionamiento de de Pantalla LCD I2C
24 //como pantalla de 20x4
25 LiquidCrystal_I2C lcd(0x27, 20, 4);
26
27 // Definición y configuración de comunicación Modbus TCP
28 //via puerto Ethernet
29 MgsModbus Mb;
30 byte mac[] = {0xFE, 0x62, 0xC7, 0xDC, 0xE8, 0xC4 };
31 IPAddress ip(192, 168, 0, 40);
32 IPAddress gateway(192, 168, 0, 1);
33 IPAddress subnet(255, 255, 255, 0);
34
```

- Finalmente se declaran las variables y se asignan los pines de entradas y salidas, para esto ver la tabla de entradas y salidas de Arduino en el Anexo: Arduino.

```
35 //Definición de variables
36 int PinEmergencia = 29;
37 int FaroEmergencia;
38
39 int Selector1 = 30;
40 int Selector2 = 31;
41 int Sel1;
42 int Sel2;
43
44 int Piloto1 = 36;
45 int Piloto2 = 37;
46 int Piloto3 = 38;
47 int Piloto4 = 39;
48 int Piloto5 = 40;
49 int Piloto6 = 41;
50
51 double Potencimetro1;
52 double Potencimetro2;
53 double Potencimetro3;
54 double Potencimetro4;
55
56 double pkp;
57 double pkd;
58 double pki;
59
60 int Frecuencia;
61 int motor1 = 6;
62 int motor2 = 5;
63
64 int distancia;
65
66 double Setpoint, Input, Output;
67 PID myPID(Input, Output, Setpoint, 2, 5, 1, DIRECT);
68
```

Configuración inicial del programa:

- Se inicializa el LCD, se enciende la luz de fondo y se escribe un mensaje colocando el cursor en la posición 1,0.
- Se inicializa el puerto serial y el Ethernet, colocando la dirección mac, IP, Gateway, subnet, anteriormente ya declaradas.
- Se establecen los pines de entrada y salida, Output para salidas digitales e Input para entradas digitales.

- Se inicializa las salidas Piloto 3 en '0' o falso.
- Se configura PID mode en Automático.

```

69 void setup()
70 {
71   //Configuracion e inicializacion de protocolos
72   //Configuracion de puertos de entrada y salidas digitales
73   led.initt();
74   led.backlight();
75   led.setCursor(1, 0);
76   led.println("Modulo 3 - Arduino");
77
78   Ethernet.begin(mac, ip, gateway, subnet);
79   Serial.println("Ethernet interface started");
80
81   Serial.begin(9600);
82
83   pinMode(Selector1, INPUT);
84   pinMode(Selector2, INPUT);
85   pinMode(PinEmergencia, INPUT);
86   pinMode(Piloto1, OUTPUT);
87   pinMode(Piloto2, OUTPUT);
88   pinMode(Piloto4, OUTPUT);
89   pinMode(Piloto3, OUTPUT);
90   pinMode(Piloto5, OUTPUT);
91   pinMode(Piloto6, OUTPUT);
92   pinMode(motor1, OUTPUT);
93   pinMode(motor2, OUTPUT);
94
95   Piloto3 = 0;
96   myPID.SetMode(AUTOMATIC);
97
98
99 }

```

Programa principal:

- Se asigna una variable a las lecturas digitales mediante el código digitalRead(), que serán: descarga de tanque, perturbación dada al sistema y un paro de emergencia.
- La función “map” permite linealizar un número de un rango a otro y convertirá según el valor de lectura analógica de Arduino con resolución de 10bits (0-1023), para este caso el valor de salidas serán el SetPoint con unidad en centímetros (cm), las constantes Kp, Ki, Kd que serán los parámetros para la función PID. Sintaxis: map(value, fromLow, fromHigh, toLow, toHigh).

```

101 void loop() {
102   //Adquisicion de datos digitales y analogicos
103   Sel1 = digitalRead(Selector1);
104   Sel2 = digitalRead(Selector2);
105   ParoEmergencia = digitalRead(PinEmergencia);
106   Potencimetro1 = analogRead(A1);
107   Potencimetro2 = analogRead(A2);
108   Potencimetro3 = analogRead(A3);
109   Potencimetro4 = analogRead(A4);
110   pkp = map(Potencimetro1, 0, 1020.00, 0, 100.00);
111   pki = map(Potencimetro2, 0, 1020, 0, 100);
112   pkd = map(Potencimetro3, 0, 1020, 0, 100);
113   Setpoint = map(Potencimetro4, 0, 1020, 2, 20);
114

```

- La función PID recibirá los parámetros asignados de map, que pueden ser modificados por el usuario, mediante el potenciómetro ingresando los datos de SetPoint, Kp, Ki, Kd.
- Adquisición de datos del nivel del tanque de líquido mediante el sensor ultrasónico para leer la distancia y obtener el nivel de líquido, se realiza una ganancia negativa para visualizar la diferencia entre un tanque 1 y tanque 2.
- Configuración de entrada y cómputo de entrada PID, cálculo de distancia o nivel del líquido en el tanque, y conversión de dato usando la función map para obtener el dato de salida analógica digital con valor de 255 a 60 Hz de salida para obtención de frecuencia del motor en módulo Unitronics.

```

115 //Parametros de valores de PID dados por el usuario
116 //mediante potenciómetros
117 myPID.SetTunings(pkp, pki, pkd);
118
119 //Adquisición de datos del nivel de tanque
120 //mediante sensor Ultrasonico HC-SR04
121 unsigned int uS = sonar.ping();
122 distancia = uS / US_ROUNDTRIP_CM;
123
124 //Configuración de entrada y cómputo de PID
125 Input = distancia;
126 myPID.Compute();
127
128 // resta de nivel de tanque con distancia sensada
129 // para ajuste y visualización de tanque 2
130 int dist = 20 - distancia;
131
132 // Conversión de dato de salida para obtención de
133 // la frecuencia de motor en módulo Unitronics
134 int Frec = map(Output, 0, 255, 0, 60);

```

- Los datos de los valores linealizados, distancia, frecuencia, y setpoint serán enviados a PLC Unitronics mediante Modbus en las diferentes direcciones, como se muestra en la imagen. El dato de la distancia se ha ingresado una ganancia negativa, esto se obtiene mediante la práctica midiendo con regla la distancia del líquido respecto a la distancia del sensor y obtener una lectura real. Además, se definen las posiciones del cursor y los diferentes mensajes que aparecen en la pantalla LCD.
- lcd.setCursor establece el cursor en la posición: (columna, fila) y con print se escriben los mensajes a visualizar.

```

136 //Envío de datos a Unitronics
137 Mb.MbData[0] = Input - 1;
138 Mb.MbData[2] = pkp;
139 Mb.MbData[3] = pki;
140 Mb.MbData[4] = pkd;
141 Mb.MbData[5] = Setpoint;
142 Mb.MbData[6] = dist;
143 Mb.MbData[7] = Frec;
144
145 // Mensajes en pantalla LCD
146 lcd.setCursor(9, 1);
147 lcd.print("SP:");
148 lcd.setCursor(13, 1);
149 lcd.print(Setpoint);
150
151 lcd.setCursor(9, 3);
152 lcd.print("OUT:");
153 lcd.setCursor(13, 3);
154 lcd.print(Output);
155
156 lcd.setCursor(9, 2);
157 lcd.print("Dist:");
158 lcd.setCursor(14, 2);
159 lcd.print(Input - 1);
160
161 lcd.setCursor(0, 1);
162 lcd.print("kp:");
163 lcd.setCursor(3, 1);
164 lcd.print(pkp);
165
166 lcd.setCursor(0, 2);
167 lcd.print("ki:");
168 lcd.setCursor(3, 2);
169 lcd.print(pki);
170
171 lcd.setCursor(0, 3);
172 lcd.print("kd:");
173 lcd.setCursor(3, 3);
174 lcd.print(pkd);

```

- Establecemos las condiciones básicas del programa que son para dar marcha al sistema o paro de emergencia tanto desde estación Arduino como de estación Unitronics, la condición indica que: si paro de emergencia (Contacto normalmente cerrado), es igual a 1 lógico, Paro HMI igual a 0 (Bit de botón recibido desde Unitronics por Modbus), Selector1 igual a 0 (Selector de descarga de tanque), se escribe al motor la salida Output generada por el cálculo del PID, caso contrario la salida o escritura analógica de motor1 se escribe en 0, así apagando el motor que controla el nivel PID. También se escribe bits digitales a las

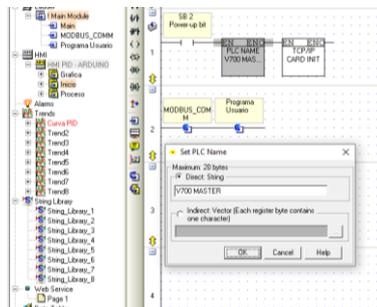
salidas de luces indicadoras del tablero, tanto Arduino y Unitronics que recibe una señal de “0” o “1” mediante dirección Modbus M1.

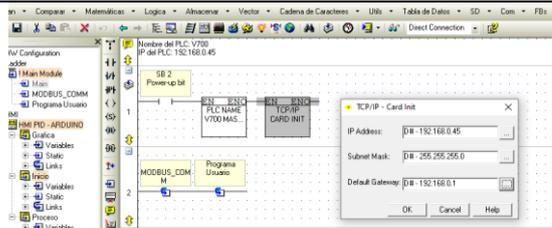
- Las siguientes condiciones serán para establecer el paro de los motores desde Arduino a Unitronics y viceversa, así como para la descarga del tanque, paro de emergencia y control de descarga. Se establece condición para el control del motor de descarga, indica si el Selector 1 de descarga esta activo, o la distancia es mayor al Setpoint (Control para descarga automática y ajuste PID), y paro HMI es igual cero, y Selector2 (Selector para encender perturbación al sistema), es igual a cero, el motor de descarga se activa, caso contrario se desactiva.
- Mb.MbsRun() define el tipo de conexión Modbus TCP/IP lo que significa que esta conexión es de tipo Modbus Server.

```
175
176 // Adquisición de dato de paro de emergencia desde pantalla
177 //HMI de Modulo de Unitronics en direccion Modbus M2
178 int paroHMI = Mb.MbData[22];
179
180 //Condicion de parada de motores en sistema
181 if (ParoEmergencia == HIGH and paroHMI == 0 and Sel1 == 0 ) {
182 //Envia de dato de parada de motor desde paro de Arduino a Unitronics
183 Mb.MbData[1] = 0;
184 digitalWrite(Piloto4, LOW);
185 digitalWrite(Piloto1, HIGH);
186 analogWrite(motor1, Output);
187 } else {
188 analogWrite(motor1, 0);
189 Mb.MbData[1] = 1;
190 digitalWrite(Piloto4, HIGH);
191 digitalWrite(Piloto1, LOW);
192 }
193 //Condicion de descarga de tanque, paro de emergencia y control de descarga
194 if ( Input - 1 > 3 ) {
195 if ( Sel1 == 1 or Input - 1 > Setpoint and paroHMI == 0 and Sel2 == 0 ) {
196 analogWrite(motor2, 70 );
197 digitalWrite(Piloto4, HIGH);
198 } else {
199 analogWrite(motor2, 0);
200 digitalWrite(Piloto4, LOW);
201 }
202 } else {
203 analogWrite(motor2, 0);
204 }
205 //Activa Perturbacion
206 if ( Sel2 == 1 ) {
207 analogWrite(motor2, 10);
208 }
209
210
211 // Arranca modulo como Modbus Server.
212 Mb.MbsRun();
213 delay(10);
214 }
```

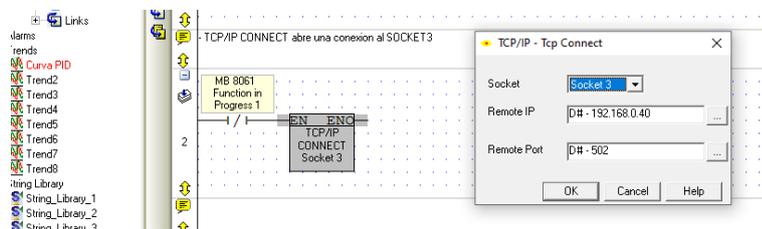
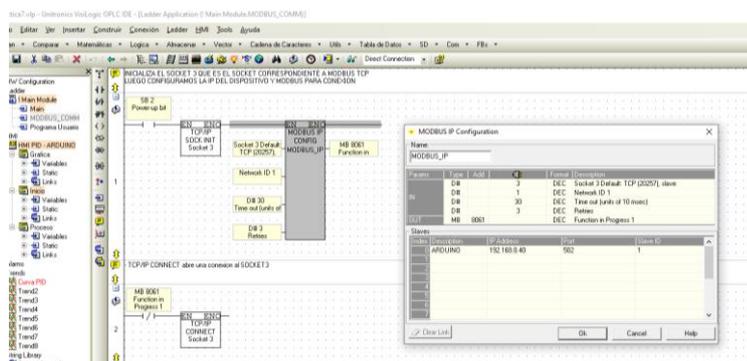
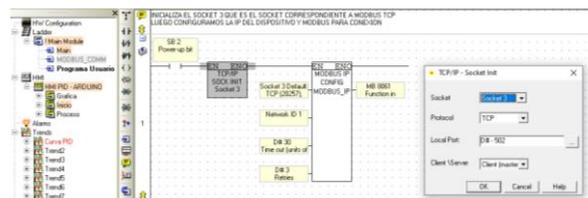
3. Desarrollo en Unitronics:

Configuración: Donde estará el nombre del PLC en este caso V700 MASTER y la dirección IP del dispositivo: 192.168.0.45

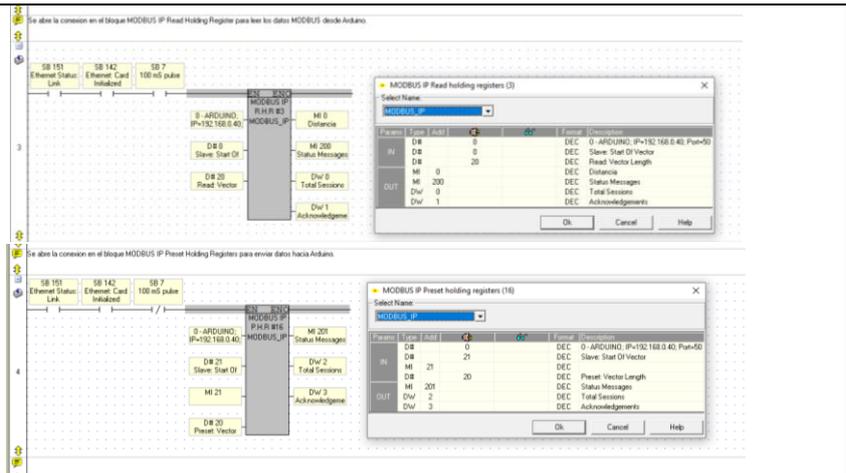




Configuración Modbus: Dónde está la configuración de Modbus Master del PLC y la conexión con el esclavo Arduino con IP: 192.168.0.40 y puerto Modbus 502 abriendo un socket de comunicación TCP.



- Cuando la señal de Pulso SB7 sea verdadera se activará este bloque, que consiste en leer los registros enviados desde Arduino
- Se configura la IP con la de Arduino: 192.168.0.40
- Para este caso se leerá desde el vector 0 hasta el Vector 20 que son las direcciones Modbus IP.
- A partir de la dirección de vector 21 se escribirá datos desde Unitronics hacia esclavo Arduino.
- El bloque empezará a leer las direcciones desde MIO.
- SB7 Es una señal de Pulso de 100 ms que estará en contacto abierto y que permitirá alternar la lectura y escritura Modbus TCP.
- En la siguiente imagen se presenta los bloques y configuración tanto de Preset Holding Registers y Read Holding Register.

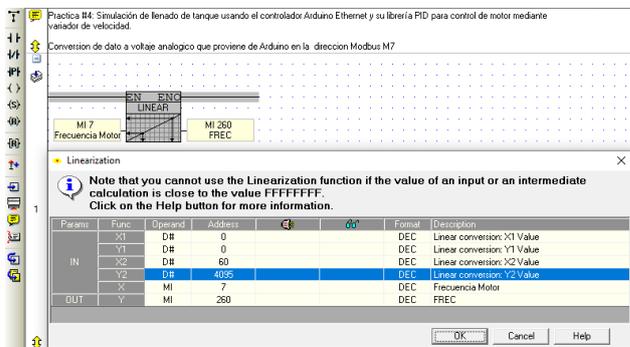


- Se crea una tabla de variables de memoria entera (MI), donde se guardarán los datos de escritura desde la dirección (MI21) y lectura Modbus desde dirección (MI0 a MI20), para esta práctica pueden ser cualquier dirección MI que estén en este rango.
- Recibimos los valores desde Arduino como: Distancia, paro de emergencia que viene desde Arduino, Kp, Ki, Kd, Setpoint, Distancia de tanque 2, y frecuencia de motor.

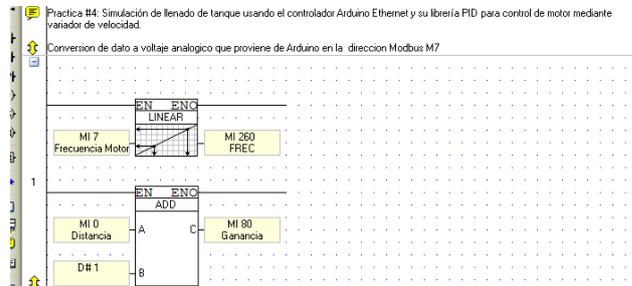
Inputs	Outputs	Distancia	Emergencia
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10
11	11	11	11
12	12	12	12
13	13	13	13
14	14	14	14
15	15	15	15
16	16	16	16
17	17	17	17
18	18	18	18
19	19	19	19
20	20	20	20
21	21	21	21
22	22	22	22
23	23	23	23
24	24	24	24
25	25	25	25
26	26	26	26
27	27	27	27
28	28	28	28
29	29	29	29

Programa de usuario: Donde estará todo el programa principal de control.

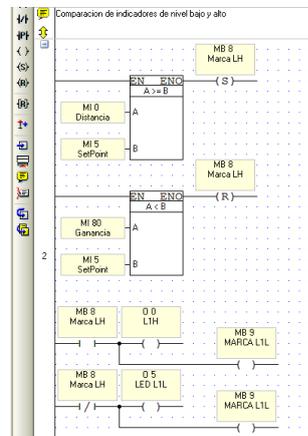
- MI7 es la Memoria Entera donde se almacena el dato de la lectura de la frecuencia de motor que viene desde Arduino.
- La señal MI260 que es la salida Analógica de Frecuencia, se escala para poder visualizarla en valores de 0 a 60Hz.



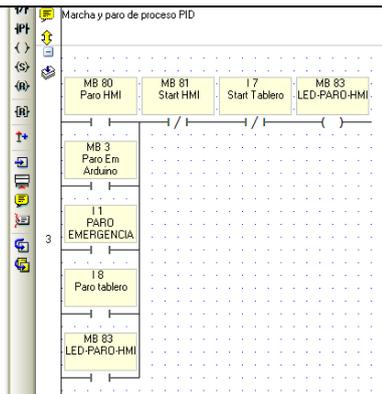
- El dato de entrada proviene de la dirección Modbus MI0 quien recibirá el dato con valor de la distancia y se suma uno con el bloque ADD de suma y obtener una ganancia.



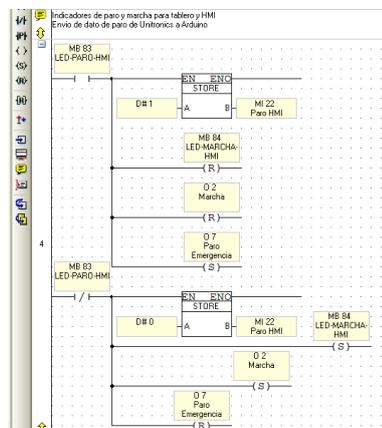
- Los siguientes bloques son de comparación que mostraran el nivel de los indicadores ya sean bajo y alto.



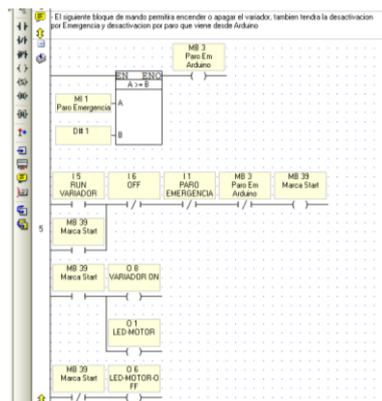
- Bloque de paro y marcha desde HMI Unitronics, la marcha es controlada solo desde Unitronics y el paro puede ser controlado de ambos, Arduino y Unitronics. Además, de indicadores de LED cuando ocurre paro en ambos programas.



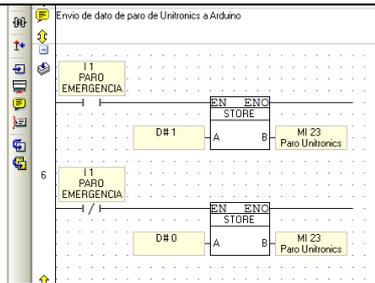
- El siguiente bloque controla los indicadores LED que se mostrarán en el tablero, el envío de datos de paro de Unitronics a Arduino para que Arduino desactive o active los motores y encienda el indicador de paro y marcha.



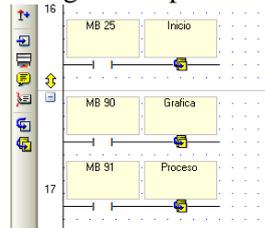
- A continuación, se muestra el bloque de mando que permitirá encender o apagar el variador, que incluirá la desactivación por emergencia y desactivación por paro que viene desde Arduino.



- El siguiente bloque controla el paro de emergencia de apagado o encendido. De Unitronics hacia Arduino.

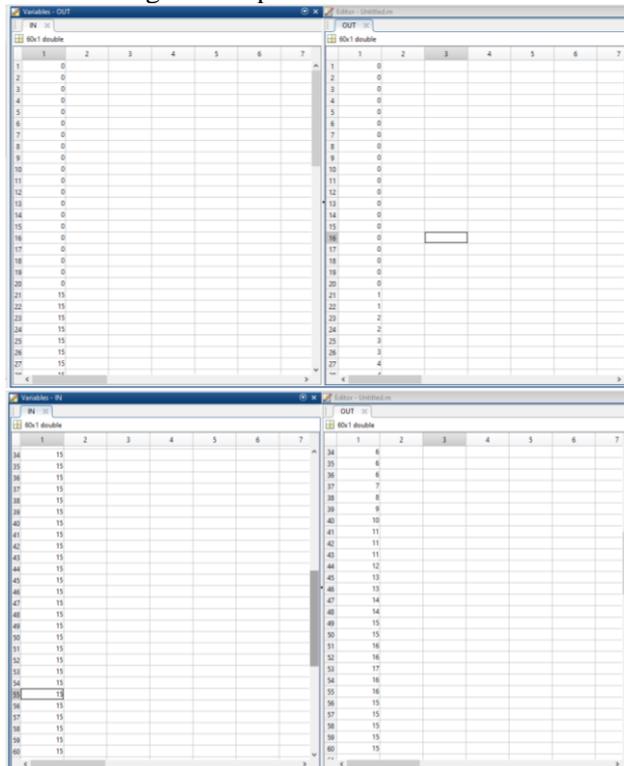


- Para navegar entre gráfica y proceso e inicio tenemos las marcas configuradas en los botones de anterior y siguiente en la pantalla HMI que nos enviaran a las siguientes pantallas.



4. Obtención de modelo matemático con Matlab.

- Se agrega una entrada escalón creando una nueva variable de tipo array en Matlab, en este caso la entrada escalón representa al Setpoint que se ha de configurar en 15 y la salida el nivel del tanque que va en incremento hasta llegar al Setpoint.

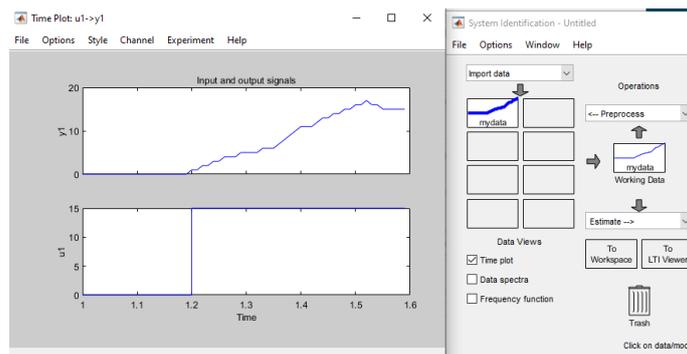
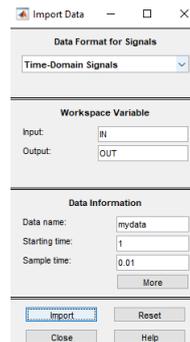


Como podemos observar han sido agregados 60 datos tanto de entrada y de salida y que han sido estimados según la observación del llenado de esta práctica en físico.

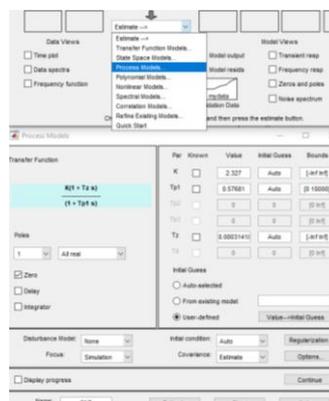
- Por comando abrir la ventana Ident (System identification Toolbox) desde la ventana de comandos de Matlab y seguimos la siguiente secuencia de pasos:

1. Abrir “Import data” y seleccionamos “Time domain data...” para estimar valores en el dominio del tiempo.

2. En la ventana que se abrió, agregar la variable de entrada y salida, IN – OUT y un tiempo de muestreo de 0.01, que es el tiempo de ejecución del módulo de Arduino donde se encuentra el control PID.

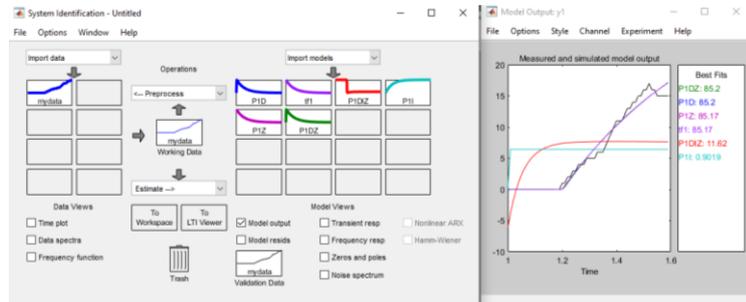


3. Luego en “Estimate” escogemos la opción “Process Models” donde debemos realizar estimaciones tanto como con el retardo o delay, agregar polos o zeros al sistema con “Zero” y agregar integradores que serán parte de la estructura de nuestra función de transferencia según las pruebas realizadas.



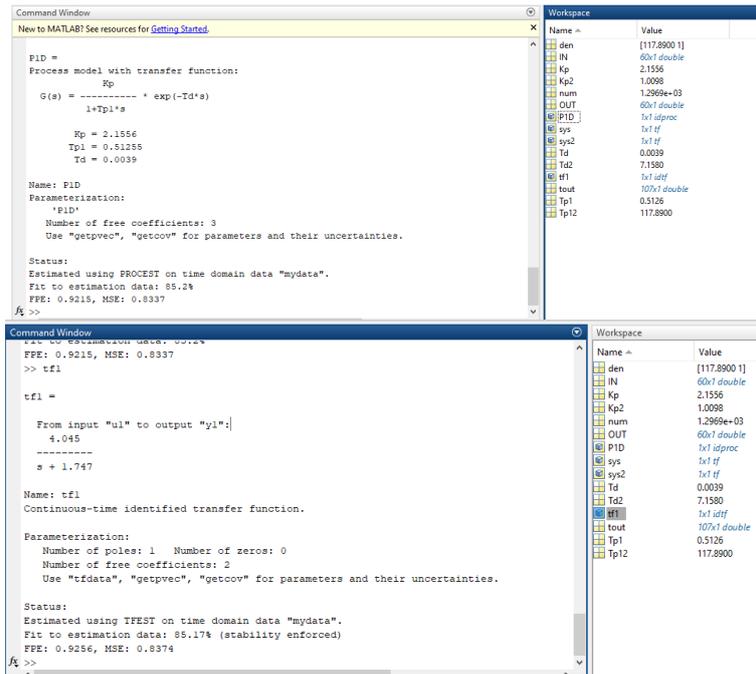
4. Se seleccionan varios formatos de función de transferencia dando check en las casillas de Zero, Delay e integrador para obtener varios

modelos y poder seleccionar la que tenga un porcentaje alto de estimación, esto quiere decir que la función de transferencia es estable respecto al tiempo.

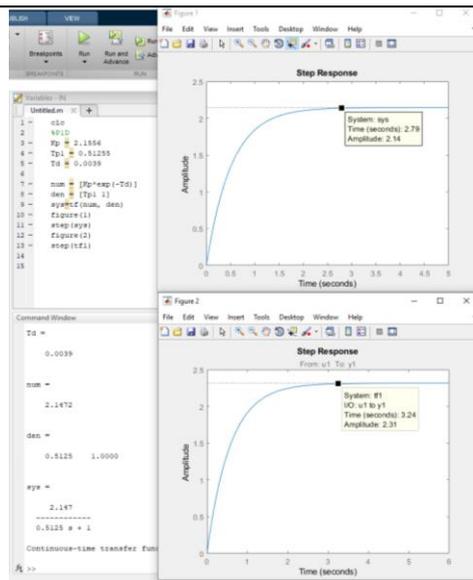


Como podemos observar, se tiene una mejor estimación en PID, PIDZ, y tf1, con que se trabajaran importando estos valores al Workspace de Matlab. (tf1 se obtuvo de estimar la función usando Transfer Function Models que es otra manera de utilizar una herramienta de estimación de funciones de transferencia).

5. Observamos nuestra función de transferencia, tanto de PID y tf1, arrastrando el dato a Command Window.

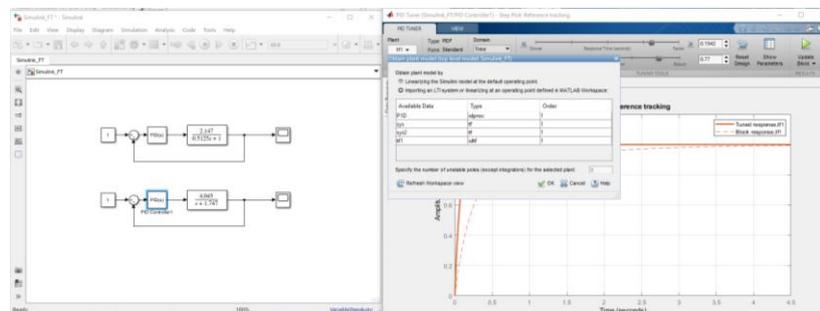


6. Creamos un pequeño Script para poder observar la Ft de PID y graficarla conjunto a tf1 y poder hacer un análisis entre gráficas.

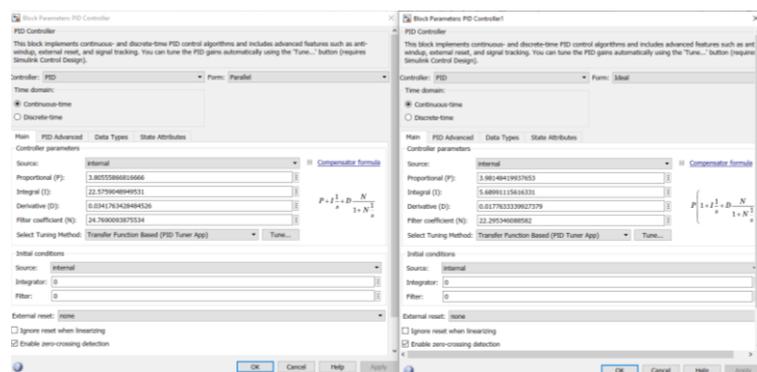


Como podemos observar, ambas graficas son similares al ser vistas desde una respuesta al escalón unitario, la primera “sys” se estabiliza en 2,79 segundos mientras que “tf1” alcanza la consigna en 3.24 segundo. De esto podemos determinar cuál función nos conviene para realizar un análisis de PID.

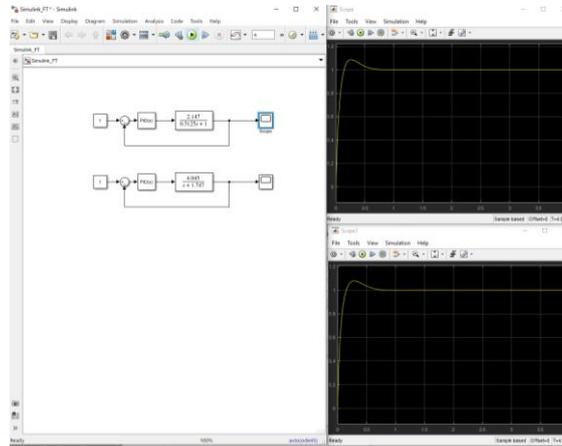
7. Se crea un control con Simulink usando “sys” y “tf1”, luego se obtiene los valores PID del bloque. Se comparan las gráficas y se ajusta el control lo más conveniente posible, luego ajustar los valores PID encontrados en el Módulo de Arduino con los potenciómetros de Kp, Ki y Kd.



Luego de actualizar los bloques PID, se tienen los siguientes valores hallados tanto para PID y tf1:



Al obtener las dos graficas tenemos que son similares, esto lo podemos verificar al hacer pruebas con los valores encontrados en Arduino tanto con la planta “PID” y “tf1”, y encontrar el mejor valor PID considerando el tiempo de estabilización, sobre impulsos, y perturbaciones.



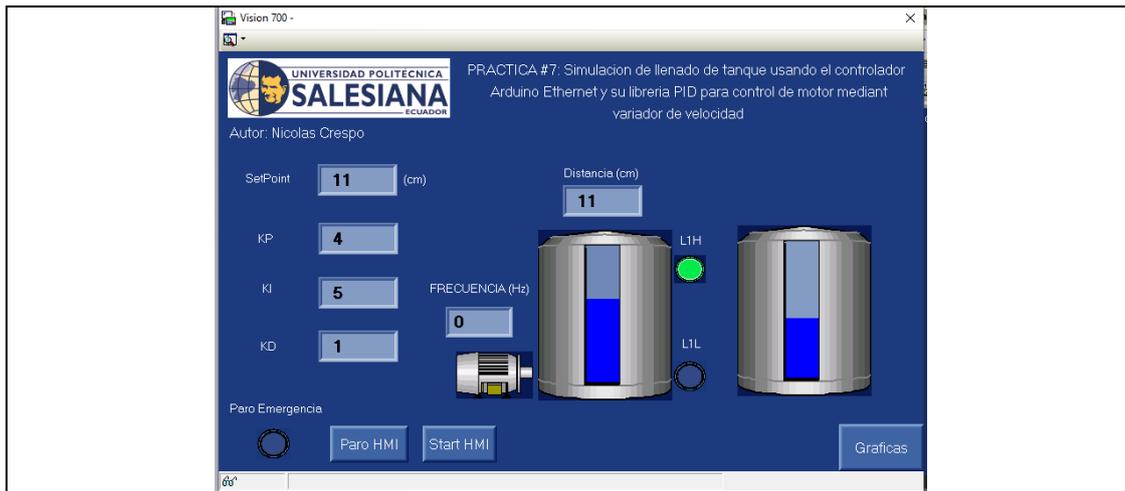
ACTIVIDADES POR DESARROLLAR

- Crear alarmas de nivel bajo, medio y alto.
- Programar el PID mediante Labview mandando los datos hacia Arduino.
- Sintonizar lazo PID usando como herramienta Matlab, obtener datos de entrada y salida y comparar los datos obtenidos de K_p , K_i , K_d mediante análisis y método de tanteo.
- Crear un informe detallado de los resultados obtenidos del llenado de tanque.

RESULTADOS OBTENIDOS:

De esta práctica se pudo sintonizar un lazo PID con los parámetros K_p K_i K_d (valores), estos se los obtuvo por método de tanteo y por el método de adquisición de datos e ingreso de los mismos usando Matlab, donde se consiguió la función de transferencia y los valores de los parámetros mencionados. Además, los tiempos de llenado van a ser mayores o menores en dependencia del método que se utilice. Adicionalmente, cuando ocurren perturbaciones el sistema trata de compensar el nivel aumentado el valor de Output, así mientras un motor está vaciando el tanque el otro motor estará tratando de mantener el nivel ingresado en el Setpoint. Finalmente, se mantuvo una coherencia entre el OUTPUT de Arduino y la frecuencia al motor conectado en el módulo de Unitronics.

La pantalla que se muestra a continuación tiene como valores K_p , K_i , K_d seteados en Arduino, con los valores hallados durante la obtención de la función de transferencia y el proceso de ajuste de PID en Matlab, la segunda imagen muestra en grafica que el control fue eficiente llegando al SetPoint, aunque también se colocó valores usando el método de tanteo para tener una mejor respuesta del sistema.



CONCLUSIONES: Se implementó una comunicación cliente servidor, además del diseño de una red entre Arduino y Unitronics, donde se desarrolló una interfaz, HMI Unitronics, para controlar de manera manual o remota el llenado y vaciado de los tanques, además de las mediciones del sensor las entradas, salidas y graficas del sistema que están mostrados en la interfaz.

RECOMENDACIONES:

- Trabajar en un rango menor de distancia cuando se use algún liquido ya que puede ocurrir un rebosamiento y dañar el motor.
- Antes de encender el motor trifásico, asegurarse que ocurra la estabilización en output, evitando caídas de frecuencias bruscas.
- Verificar mediante los indicadores Led de las antenas que existe una conexión entre estas.
- Verificar que la dirección IP de la de los dispositivos estén en el mismo rango de direcciones configuradas en las antenas previamente.

Docente/Técnico Docente: _____

Firma: _____

4.8 Práctica 8

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados	
NRO. PRÁCTICA:	08	TÍTULO PRÁCTICA: Diseño de arquitectura y control de tratamiento de agua potable simulado.	
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS TCP/IP en PLC Siemens. 2. Diseñar una red MODBUS TCP/IP entre PLC Siemens, PLC Unitronics, Labview y Arduino. 3. Entender el proceso de potabilización del agua. 4. Crear un sistema SCADA – HMI que gobierne los procesos de potabilización. 			
DEV:	<ol style="list-style-type: none"> 1. Se desea realizar un proceso de potabilización del agua que cumpla los siguientes procesos: <ol style="list-style-type: none"> 1. Captación del agua bruta. 2. Coagulación. 3. Floculación. 4. Sedimentación. 5. Decantación. 6. Filtración. 7. Desinfección 8. Depósitos de reserva y transporte. 		
	<ol style="list-style-type: none"> 2. Conceptos de proceso de potabilización: <ul style="list-style-type: none"> • Captación A través de la toma, el agua pasa a través de una rejilla que evita la entrada de residuos grandes. Luego el agua es pre tratada y se envía a un depósito de arena donde se acumula en estado flotante. El agua bruta sigue fluyendo hasta el interruptor seccionador, donde se distribuye simultáneamente a las tres unidades de producción. • Coagulación Al igual que cuando se toma agua de un río o arroyo, cuando se sedimenta el agua mineral, las partículas que componen la turbidez y el color son tan ligeras que no sedimentan. Estas partículas tienen carga negativa y las cargas del mismo signo se repelen entre sí, por lo que en estas condiciones no es posible reagruparlas para obtener partículas de mayor tamaño. Este proceso consiste en neutralizar estas partículas mediante la adición de una carga positiva con un producto denominado coagulante. Este proceso dura unos segundos y requiere mucha acción de parte del agitador para que aparezca Coagul se complemente con él con agua en el menor tiempo posible. El punto máximo de mezcla en el que se introduce el coagulante se denomina mezcla rápida • Floculación Después de que el agua se coagula, las partículas no tienen carga en la superficie y no hay interrupción para que las partículas se vuelvan a unir. Para hacer esto, el agua debe agitarse lentamente para que las partículas solidificadas choquen y 		

formen partículas más grandes llamadas FLOCS. Este proceso debe ejecutarse en condiciones muy bien controladas. Sacudir muy fuerte en esta etapa puede provocar la ruptura de los ojos ya formados, pero sacudir muy lentamente puede provocar la formación de ojos esponjosos y débiles complicados de sedimentar.

- **Sedimentación**

Este proceso es el primer paso efectivo en la separación de partículas del agua y se logra una turbidez y color reducidos en comparación con el agua cruda. La separación entre el líquido claro que flota en la superficie y el lodo que contiene la alta concentración de sólidos que se deposita por gravedad. Esto se debe a que su densidad es mayor que el líquido ya tratado.

- **Decantación**

La estructura más visible dentro de una planta de tratamiento de agua potable son los decantadores. Entonces, cuando se agrega productos químicos, floculantes y coagulantes, las partículas se combinan para formar otros grumos grandes que se hunden hasta el fondo del tanque de sedimentación. Los detalles instalados en una fábrica llamada pulsador son una de las innovaciones introducidas en el sistema de tratamiento de agua. Mantienen la suciedad en la suspensión central, ejecutan el movimiento de "respiración" que excreta los materiales excesivos en el búnker central y conduce este material al tratamiento. El agua superficial está limpia y lista para los pasos de purificación. Esta mejora acelera el proceso y mejora la eficiencia general de producción de agua potable.

- **Filtración**

Este es el paso final en el proceso de purificación del agua. Consiste en hacer pasar agua a través de un medio poroso que por lo general es arena, para eliminar los sólidos y otros elementos visibles en el agua. El filtro retiene partículas de baja densidad y las que quedan por alguna razón. El objetivo más resaltante es detener a los elementos patógenos. Si el filtro está obstruido, será necesario limpiarlo. El tiempo que se tarda en lavar el filtro dos veces seguidas se denomina ciclo de filtración.

- **Desinfección**

Se introduce cloro al agua para así hacer el líquido microbiológicamente inocuo. El cloro es el producto químico más común usado como desinfectante debido a su alta actividad oxidativa y residual. Su finalidad es destruir los microorganismos patógenos presentes en el agua: (Salmonella, Shigella, Vibrio cholerae, E. coli). Kori), protozoos y virus.

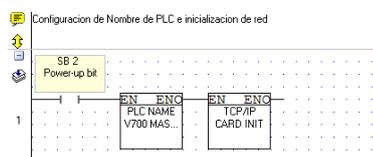
- **Depósitos de Reserva**

Luego de cumplidos estos procesos, el agua es apta para el consumo humano, el líquido final es almacenado en los tanques de reserva donde luego es llevada a las tuberías de distribución para la ciudad.

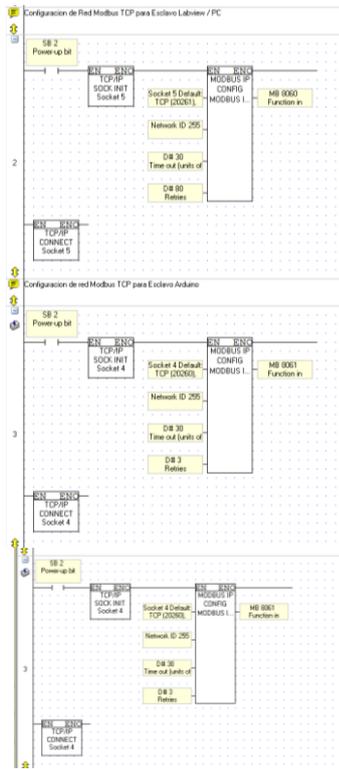
3. Diseñar una interfaz en PLC Unitronics que contenga los procesos:

1. Captación del agua bruta.
2. Coagulación.

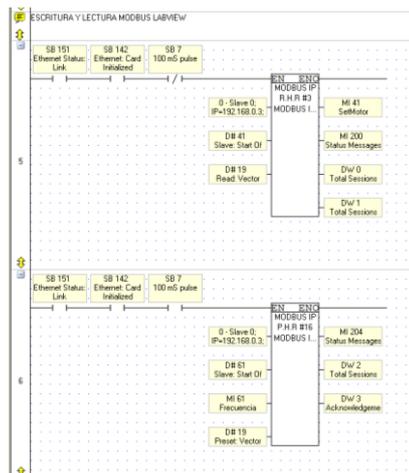
- Se configura el nombre del PLC y la dirección IP: 192.168.0.45.

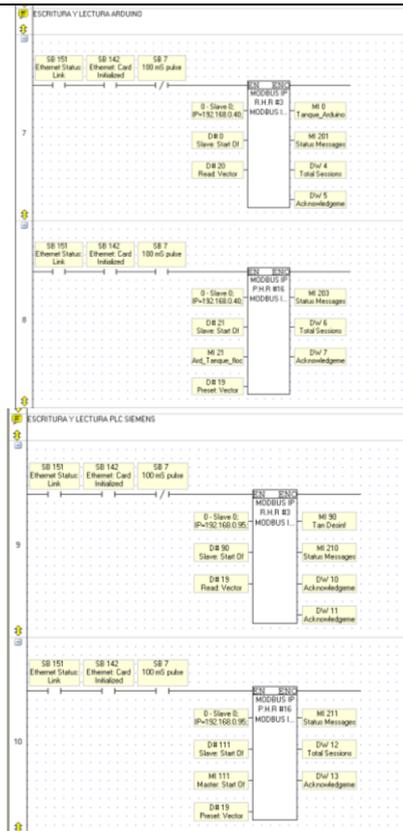


- Configuración de los dispositivos Esclavos Modbus TCP.

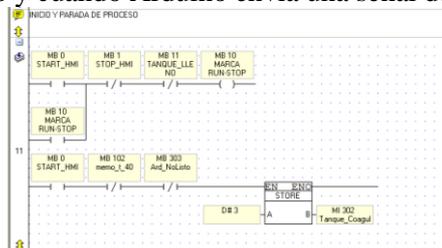


- Configuración de equipos esclavos para lectura y escritura Modbus.

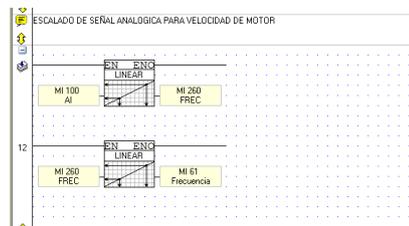




- Inicio y parada de llenado de tanques, se detiene cuando tanque se llena al valor dado y cuando Arduino envía una señal de proceso listo.

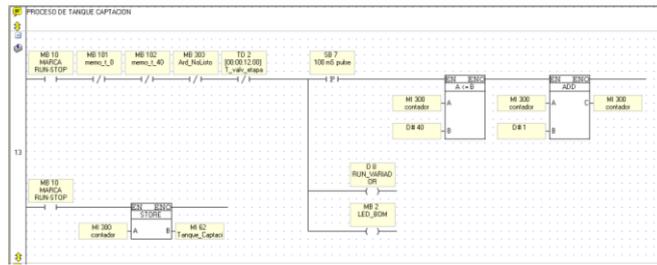


- Escalado de señal analógica y envío de variable frecuencia a motor y a visualización mediante Modbus.

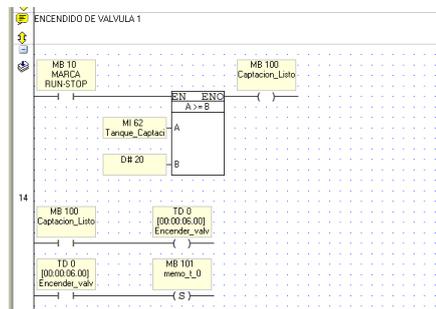


- El tanque captación se llena mientras el valor aún no se alcancen las 40 Unidades, y mientras Arduino no envíe señal de tanques llenos. Se enciende variador de frecuencia para simular el llenado y el contador se activa hasta alcanzar un valor de 40 Unidades que corresponden al nivel

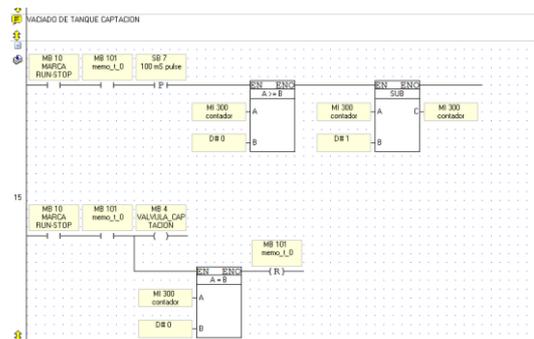
de tanque, a su vez la variable de contador se envía por Modbus a Labview para visualizar el nivel de tanque Captación.



- Cuando el nivel de tanque captación esté listo, se enciende un temporizador que espera 6 Segundos para encender la válvula uno, empezar a descargar el tanque captación y llenar el tanque coagulación.

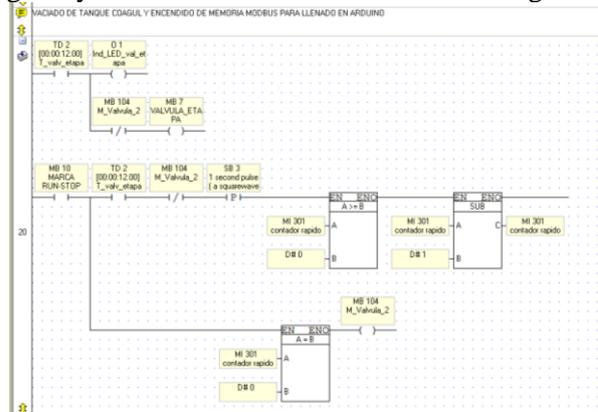


- El contador MI300 empieza a decrecer hasta que tome el valor de cero que corresponden al nivel de tanque de captación, y la válvula V1 se cierra y enciende la válvula V2. Este proceso se realiza a la vez, es decir que mientras el tanque captación se llena, el tanque coagulación se llena.

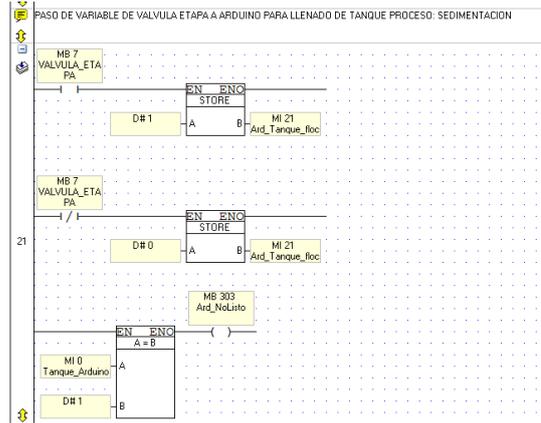


- El tanque coagulación se llena hasta 40 unidades cuando el tanque captación alcanza las 0 Unidades o es decir que se vacía. La variable de tanque coagulación se envía a dato Modbus hacia Labview.

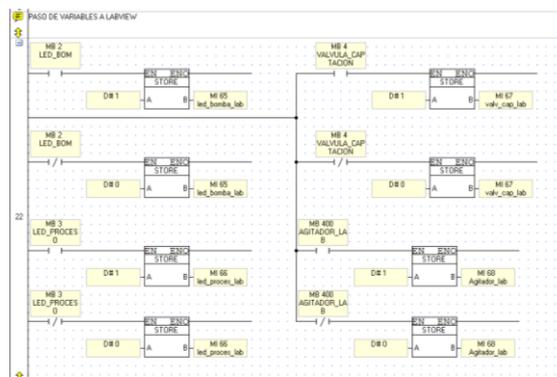
- Cuando la válvula etapa se enciende, el tanque coagulación empieza a descargarse y se detiene cuando el nivel de este llega a 0.

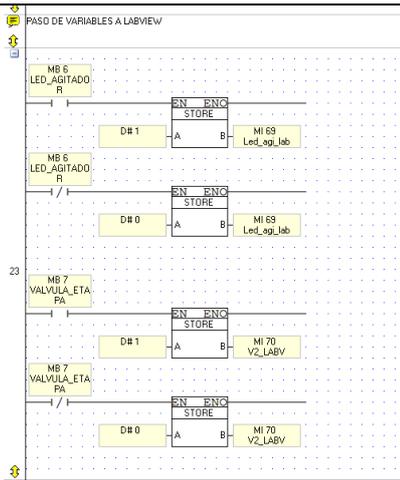


- Se pasa las variables de válvula etapa que indica a Arduino que puede empezar a llenar su tanque, también se recibe en MIO la señal que indica cuando el tanque en Arduino que corresponde al proceso de sedimentación esté listo para detener el proceso que se esté llevando a cabo en PLC Unitronics.

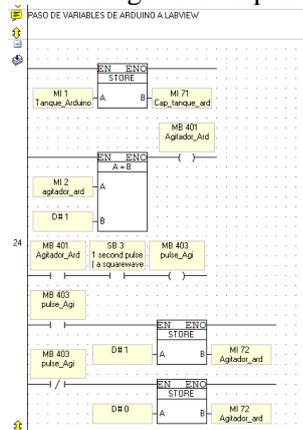


- Se envían las variables a Labview para tener visualización de todo el proceso en la PC.

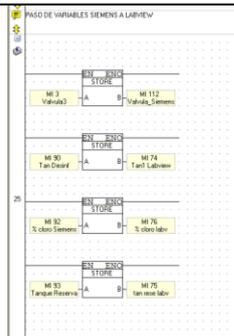




- Las variables que vienen de Arduino se envían a Labview, cuando el agitador se enciende desde Arduino, se coloca una señal de Pulso que simulará el movimiento del agitador en pantalla Labview.



- Se envía y se recibe las señales de PLC Siemens como el encendido de la válvula que da paso a proceso de siemens, este paso lo da Arduino, cuando active MI3, luego desde MI90 el valor de tanque desinfección, MI92 el valor de sensor simulado de cantidad de cloro en líquido, y por último el valor de llenado de tanque reserva para finalizar el proceso.



4. Diseñar una Interfaz en Arduino que contenga los procesos:

1. Floculación.
2. Decantación.

- Se usan las librerías necesarias para esta práctica, considerando una nueva librería llamada Crescer.h que activa temporizadores, su funcionamiento puede revisarse en el apartado de Anexos.

```

practica_8_new
1 //UNIVERSIDAD POLITÉCNICA SALESIANA
2 //TESIS DE GRADO DE INGENIERIA ELECTRONICA
3 //AUTOR: STOLIAS CHEPO DELGADO
4 //Tema: PRACTICA 8: Diseño de arquitectura de tratamiento de agua
5 //potable simulado
6
7 //librerias
8
9 #include "MgmoBus.h"
10 #include <Ethernet.h>
11 #include <SPI.h>
12 #include <Wire.h>
13 #include <liquidCrystal_I2C.h>
14 #include <Ultrasonic.h>
15 #include <Crescer.h>
16

```

- Se configura el proyecto en la dirección IP 192.168.0.40. Se configura la dirección de LCD y los pines de sensor ultrasónico para medir el nivel de tanque, también se configura los dos temporizadores a usar que corresponden a agitador y tiempo de espera de llenado de tanque.

```

17 // Definición y configuración de comunicación Modbus TCP
18 //via puerto Ethernet
19 MgmoBus Mb;
20 byte mac[] = {0xFF, 0x42, 0x42, 0x47, 0x4C, 0x48, 0x44 };
21 IPAddress ip(192, 168, 0, 40);
22 IPAddress gateway(192, 168, 0, 1);
23 IPAddress subnet(255, 255, 255, 0);
24
25 //Definición de dirección de LCD I2C, tipo de LCD
26 // Y pines de conexión de sensores ultrasónicos
27 liquidCrystal_I2C lcd(0x27, 20, 4);
28
29 Ultrasonic ultrasonico(14, 15);
30
31 Tempora temp1;
32 Tempora temp2;

```

- Se declaran las variables a usar en este proyecto.

```

33
34 int pul1 = 34;
35 int pul2 = 35;
36
37 int Piloto1 = 36;
38 int Piloto2 = 37;
39 int Piloto3 = 38;
40 int Piloto4 = 39;
41 int Piloto5 = 40;
42 int Piloto6 = 41;
43
44 int motor1 = 6;
45 int motor2 = 5;
46 int M_llenado;
47 int distancia;
48 int floc;
49 unsigned long TiempoAnterior1 = 0;
50 int Agitador = 0;
51 int Valvula3;

```

- Se definen los modos de uso de los pines en Arduino, se inicializan protocolos y se da el tiempo de retardo de los temporizadores, temp1 en 10 Segundos y temp2 en 4 Segundos.

```

53 void setup() {
54 //Configuracion e inicializacion de protocolos
55 //Configuracion de puertos de entrada y salidas digitales
56 pinMode(Piloto1 , OUTPUT);
57 pinMode(Piloto2 , OUTPUT);
58 pinMode(Piloto4 , OUTPUT);
59 pinMode(Piloto3 , OUTPUT);
60 pinMode(Piloto5 , OUTPUT);
61 pinMode(Piloto6 , OUTPUT);
62
63 Ethernet.begin(mac, ip, gateway, subnet); // start etehrn
64 Serial.println("Ethernet interface started");
65 Serial.begin(9600);
66
67 led.init();
68 led.backlight();
69
70 pinMode(motor1 , OUTPUT);
71 pinMode(motor2 , OUTPUT);
72 temp1.defiSP(10000);
73 temp2.defiSP(4000);
74 }
75

```

- Se escala la señal de distancia en 43 Unidades que son el nivel de tanque usado desde PLC Unitronics, la variable que viene desde Unitronics con Modbus, inicia el proceso de llenado de tanque en Arduino hasta que este sea menor igual a 3 Unidades.

```

76 void loop() {
77 //Lectura sensor Ultrasonico tanques
78 distancia = map(ultrasonic2.read(CM), 2, 17, 0, 43);
79
80
81 //Lectura Modbus Unitronics - Labview
82 int Unit_tanque_floc = Mb.MbData[21];
83
84 //Señal de Unitronics para llenar tanque en Arduino
85 if (Unit_tanque_floc == 1) {
86 Mb.MbData[0] = 1;
87 M_llenado = 1;
88 }
89
90 if (distancia <= 3) {
91 M_llenado = 0;
92 }
93
94 if (M_llenado == 1) {
95 temp1.Saida(0);
96 analogWrite(motor2, 255);
97 }
98
99 if (M_llenado == 0) {
100 analogWrite(motor2, 0);
101 }
102

```

- Cuando el nivel de tanque este entre 0 y 3 Unidades, se envía una señal a PLC Unitronics que corresponde a Arduino Listo para parar el proceso de descarga de tanque y luego si el motor 2 esta apagado, se activa el temporizador de 4 Segundos, una vez se cumpla el tiempo empieza el proceso de floculación.

```

102
103 if (distancia >= 0 and distancia <= 3 ) {
104 Mb.MbData[0] = 1;
105 Serial.println(temp2.CV);
106 if (digitalRead(motor2) == 0) {
107 temp2.Saida(1);
108 if (temp2.Saida(1) == 1) {
109 floc = 1;
110 }
111 } else {
112 temp2.Saida(0);
113 }
114 }
115
116

```

- Cuando la variable floc sea verdadera, el temporizador de 10 Segundos se activa, mientras el temporizador este activo, el agitador se enciende, una vez el tiempo se cumpla la válvula 3 se enciende y el motor 1 se enciende para pasar el líquido al siguiente tanque y proceso.

```

119 // Floculacion
120
121 if (floc == 1) {
122
123     lcd.setCursor(0, 2);
124     templ.Saida(1);
125
126     if (templ.Saida(1) == 0) {
127         digitalWrite(Pilotol, HIGH);
128         Agitador = 1;
129     }
130
131     if (templ.Saida(1) == 1) {
132         digitalWrite(Pilotol, LOW);
133         Agitador = 0;
134         analogWrite(motor1, 255);
135         Valvula3 = 1;
136     }
137 }

```

- Cuando la distancia sea mayor a 45 Unidades, el proceso floculación se detiene.

```

140 if (distancia >= 45 ) {
141     floc = 0;
142     Valvula3 = 0;
143     Mb.MbData[0] = 0;
144     analogWrite(motor1, 0);
145 }
146

```

- Los parametros del proceso se visualizan en la pantalla LCD.

```

152 // visualizacion en pantalla LCD
153 lcd.setCursor(0, 0);
154 lcd.print("Practica# 8");
155 lcd.setCursor(0, 1);
156 lcd.print("Nivel: ");
157 lcd.setCursor(8, 1);
158 lcd.print(42 - distancia);
159
160 lcd.setCursor(0, 2);
161 lcd.print("Motor: ");
162 if (M_llenado == 1) {
163     lcd.setCursor(8, 2);
164     lcd.print("Encendido");
165 } else {
166     lcd.setCursor(8, 2);
167     lcd.print("Apagado");
168 }
169
170 if (Agitador == 1) {
171     lcd.setCursor(0, 3);
172     lcd.print("AGITADOR: ON");
173 } else {
174     lcd.setCursor(0, 3);
175     lcd.print("AGITADOR: OFF");
176 }
177
178 Mb.MbsRun();

```

- Este cronometro permite que los datos en LCD se vayan actualizando.

```

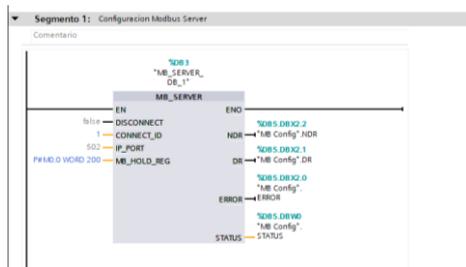
180 if (millis() - TiempoAnterior > 500) {
181     TiempoAnterior = millis();
182     lcd.clear();
183 }

```

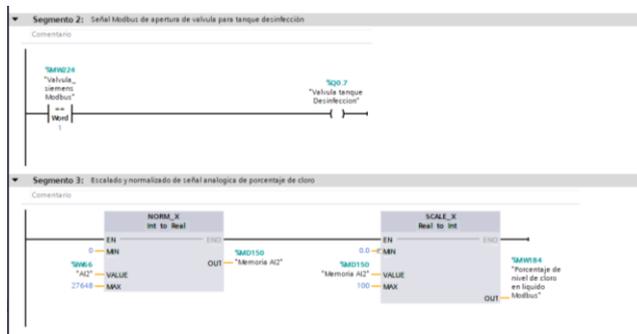
5. Diseñar la programación en PLC Siemens que tenga los siguientes procesos:

1. Desinfección
2. Depósitos de reserva y transporte.

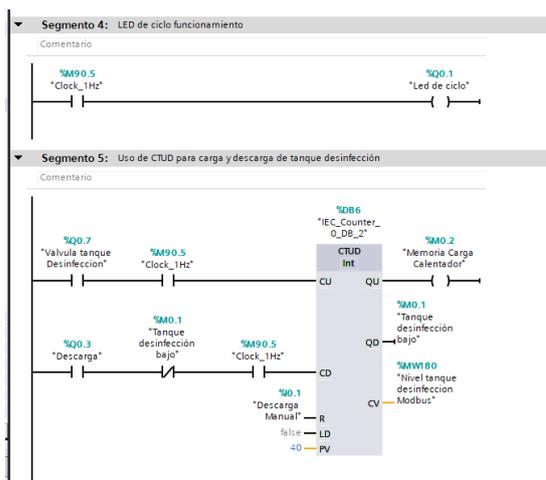
- Se configura el bloque Modbus en Siemens. La dirección IP del dispositivo debe ser 192.168.0.95



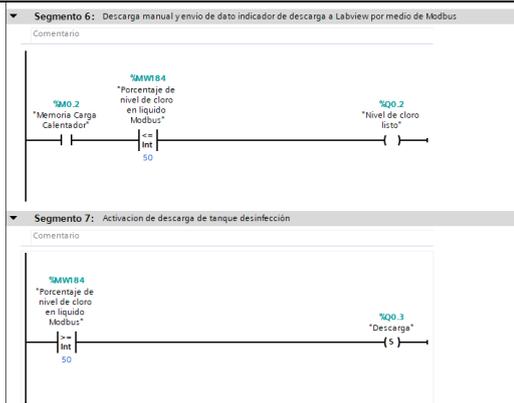
- La señal Modbus que viene en la dirección M112 y que se lee en PLC Siemens como dirección MW224, es la señal de inicio de proceso en PLC Siemens, o para el proceso es su apertura de válvula para empezar a llenar el tanque desinfección.
- Los bloques de Normalizado y Escalado obtienen la señal simulada de sensor de nivel de cloro en líquido que va de 0 a 100%.



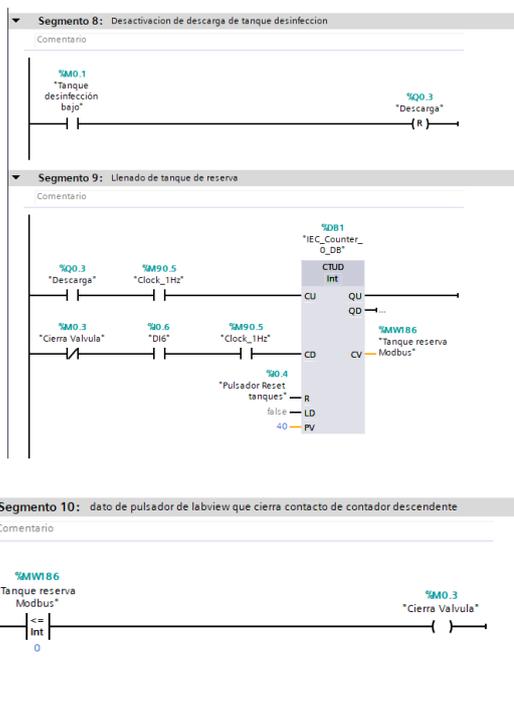
- Cuando la válvula tanque desinfección se active, el contador se activa empezando a contar hasta 40 Unidades, la señal de reloj da el pulso para el contador haciendo que el nivel de tanque crezca.



- Cuando el tanque desinfección se ha llenado hasta las 40 Unidades, se espera a que el porcentaje de nivel de cloro en liquido llegue a 50%, una vez está listo se activa una señal para descarga de este tanque.

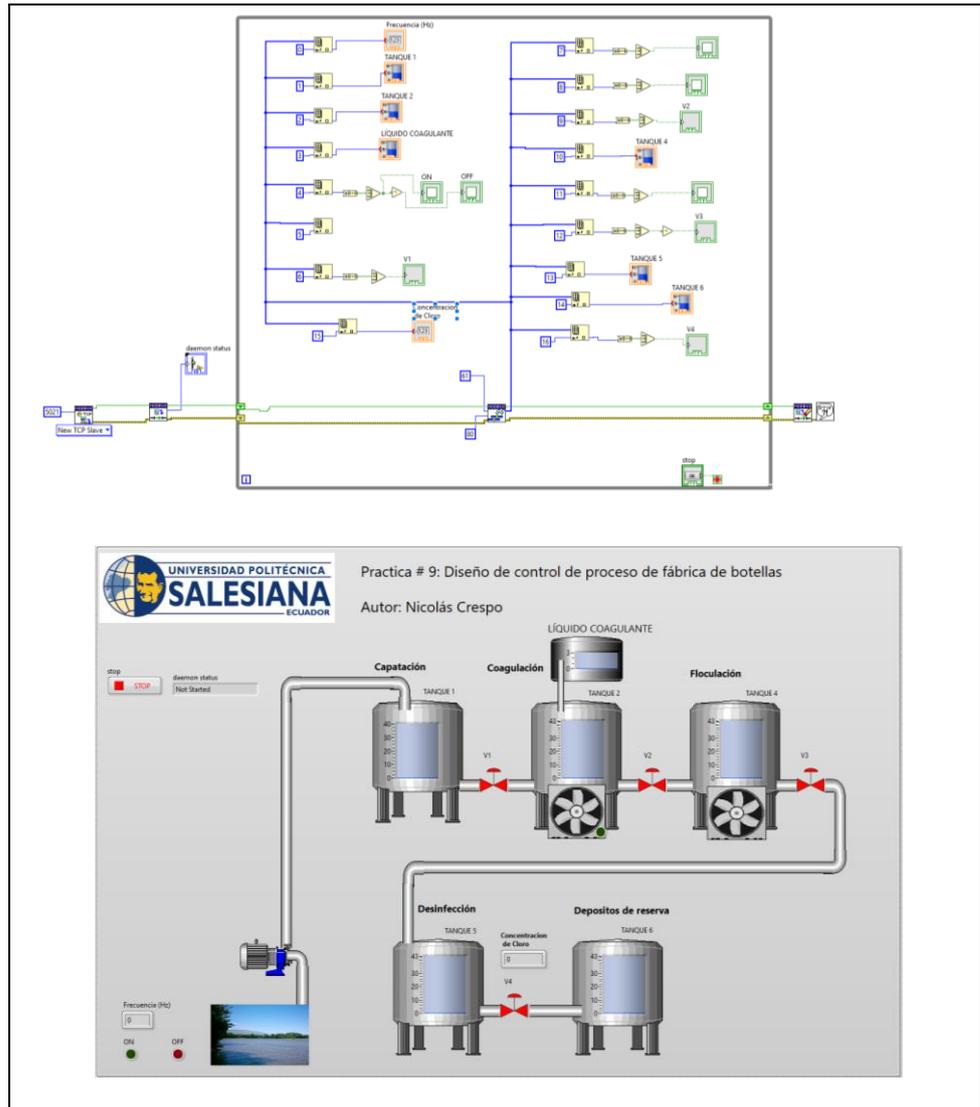


- Cuando en el tanque desinfección el nivel de cloro llegue a su nivel seteado, empezará la descarga del tanque desinfección y la carga del tanque de reserva de líquido hasta que llegue a las 40 Unidades y luego la válvula se cierra.



6. Diseñar una interfaz en Labview PC, donde se puedan visualizar todos los procesos del tratamiento de agua potable.

Los datos que vienen del PLC Unitronics Maestro, son leídos para así armar la interfaz de visualización del proceso completo de simulación de potabilización de agua.



ACTIVIDADES POR DESARROLLAR

- Considerar colocar sensores de nivel en interfaz en Labview para mayor comprensión del usuario.
- Considerar colocar un indicador de final de proceso para dar paso a un nuevo proceso.
- Realizar un informe detallado con los resultados obtenidos en la práctica.

RESULTADOS OBTENIDOS:

Se simuló un proceso de potabilización de agua potable haciendo interactuar los dispositivos conectados a la red Modbus TCP, cuando uno de los dispositivos terminaba su proceso daba la señal de apertura de válvula para que el siguiente continúe con el proceso y luego se visualizó todo el proceso desde una sola pantalla que es la de Labview, quien recibió todas las señales de nivel y actuadores como motores y válvulas.

Se cumplió el proceso de tanque captación, luego el paso a tanque coagulación que se mezcló con el líquido coagulante, luego se dio el paso para llenar tanque floculación en Arduino para finalmente pasar a tanques desinfección y reserva en PLC Siemens.

CONCLUSIONES:

En esta práctica se volvió a demostrar la comunicación entre todos los dispositivos Modbus TCP usando el PLC Unitronics como Maestro y los demás dispositivos como esclavos.

RECOMENDACIONES:

- Se recomienda realizar una tabla con los direccionamientos Modbus para una mayor comprensión.
- Verificar que la dirección IP de la de los dispositivos estén en el mismo rango de direcciones configuradas en las antenas previamente.
- Revisar la conexión a motor Trifásico antes de energizar el tablero para evitar choques eléctricos o un mal funcionamiento de este.

Docente/Técnico Docente: _____

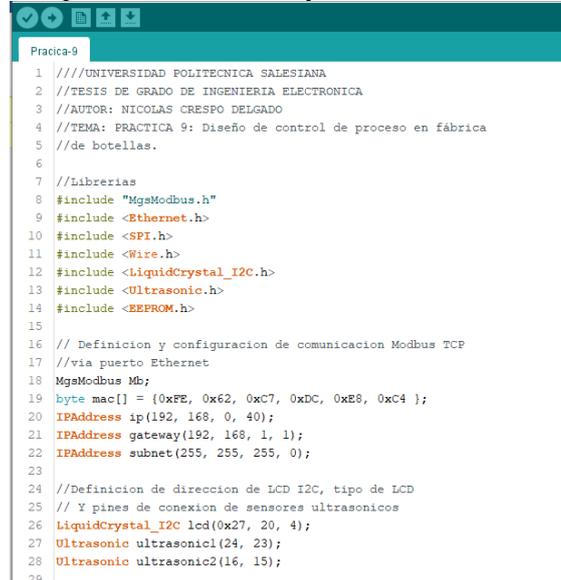
Firma: _____

4.9 Práctica 9

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados	
NRO. PRÁCTICA:	09	TÍTULO PRÁCTICA: Diseño de control de proceso en fábrica de botellas.	
OBJETIVOS: <ol style="list-style-type: none"> 1. Diseñar una red MODBUS TCP/IP entre PLC Unitronics como Master y esclavos: PC, Arduino y PLC Siemens S7-1200. 2. Realizar encendido de sistema en cualquiera de las estaciones de control. 3. Realizar un sistema SCADA – HMI en PLC Unitronics y PC. 4. Interactuar con los sensores conectados al controlador Arduino. 			
DEV:	1. El proceso lleva dos botones físicos en Arduino, Start, Stop desde tablero y un paro de emergencia en la banda transportadora.		
	2. El sistema arrancará desde cualquiera de las estaciones con botones de Start y Stop del sistema. El motor trifásico simula estar conectado a la banda transportadora		
	3. Al llegar la muestra de botella al sensor IR1 la banda se detiene y enciende el calentador que será controlado y simulado desde PLC Siemens S7-1200, hasta que llegue a temperatura dada en el sistema PC SCADA de Labview. Esta parte simula la fundición, formado y enfriado de resina PET (botella plástica de Teraflato de Polietileno).		
	4. Para esta práctica se programa dos presentaciones de botellas, de 20 mililitros y de 50 mililitros, si la botella tiene un tamaño pequeño, el sensor ultrasónico colocado en la banda transportadora detectará que tipo de presentación es para poder llenar la botella con el volumen correspondiente.		
	5. Al llegar al sensor IR2 se enciende el motor de llenado y se llenará la botella dependiendo de la presentación que corresponde, además este contará el número de botellas que pasan en ese instante hasta que cumpla con la condición de cantidad de botellas a llenar y el sistema se para y enciende un indicador de proceso terminado.		
	6. Cuando el tanque 1 de llenado de botellas se encuentre a un nivel bajo este se enciende hasta llenar el reservorio.		
	7: realizar la siguiente tabla de direccionamiento Modbus, o como sea el caso de parte del desarrollador.		
1. Programación en Arduino. <ul style="list-style-type: none"> • Se incluyen las librerías para esta práctica: MgsModbus, Ethernet, LiquidCrystal I2C, Ultrasonic, EEPROM 			

DEV:

- Se declara la variable MgsModbus Mb para ser usada en el direccionamiento y se configura la dirección IP del Arduino.
- Configuramos la dirección y el tipo de LCD I2C.
- Se declaran los pines de ultrasónico y el nombre de los mismos.



```
1 ///UNIVERSIDAD POLITECNICA SALESIANA
2 ///TESIS DE GRADO DE INGENIERIA ELECTRONICA
3 ///AUTOR: NICOLAS CRESPO DELGADO
4 ///TEMA: PRACTICA 9: Diseño de control de proceso en fábrica
5 //de botellas.
6
7 //Librerias
8 #include "MgsModbus.h"
9 #include <Ethernet.h>
10 #include <SPI.h>
11 #include <Wire.h>
12 #include <LiquidCrystal_I2C.h>
13 #include <Ultrasonic.h>
14 #include <EEPROM.h>
15
16 // Definicion y configuracion de comunicacion Modbus TCP
17 //via puerto Ethernet
18 MgsModbus Mb;
19 byte mac[] = {0xFE, 0x62, 0xC7, 0xDC, 0xE8, 0xC4 };
20 IPAddress ip(192, 168, 0, 40);
21 IPAddress gateway(192, 168, 1, 1);
22 IPAddress subnet(255, 255, 255, 0);
23
24 //Definicion de direccion de LCD I2C, tipo de LCD
25 // Y pines de conexion de sensores ultrasonicos
26 LiquidCrystal_I2C lcd(0x27, 20, 4);
27 Ultrasonic ultrasonic1(24, 23);
28 Ultrasonic ultrasonic2(16, 15);
29
```

- Se declaran los puertos de entrada y salida, la primera variable: sensor que está en el pin 3 corresponde al sensor de pulso, y las siguientes son las variables de configuración del sensor de pulso para hallar el valor real del sensor dado en litros por min.
- Se utiliza dos variables de: “litros_min” para obtener el valor de llenado en ese instante y los valores acumulados en el llenado.
- La variable pulsos son aquellos que se van dando cuando el líquido pasa por la turbina.
- El valor de pulsos acumulados, para luego mediante formula convertirlo a una unidad de pulsos vs litros.



```
30 //Definicion de variables y puertos IN OUT
31 const int sensor = 3;
32 double litros_min;
33 volatile int pulsos = 0; // Variable que almacena el número de pulsos
34 unsigned long tiempoAnterior = 0; // Variable para calcular el tiempo transcurrido
35 unsigned long pulsos_Acumulados = 0; // Variable que almacena el número de pulsos acumulados
36 float litros;
37
38 int litros_min1;
39 volatile int pulsos1 = 0; // Variable que almacena el número de pulsos
40 unsigned long tiempoAnterior1 = 0; // Variable para calcular el tiempo transcurrido
41 unsigned long pulsos_Acumulados1 = 0; // Variable que almacena el número de pulsos acumulados
42 float litros1;
43
```

- pul1 y pul2 corresponden a los pulsadores de marcha y paro.
- Piloto1-6 se deben declarar siempre en las prácticas y corresponden a las luces pilotos en el proyecto.
- Potenciometro1 corresponde a la velocidad de la banda transportadora.

- SetMotor corresponde a la memoria de encendido de la banda transportadora.
- Reset Motor corresponde a la memoria de apagado de la banda transportadora.
- Motor1 y Motor 2 son los motores de las bombas de llenado de líquido.
- motorBanda corresponde a el motor de banda Transportadora.
- SensorIR1 y SensorIR2 son los sensores que van al inicio y casi al final de la banda que detectaran el paso del objetivo en ese instante.
- Emergencia corresponde a el paro de emergencia.
- Las siguientes variables serán usadas internamente en la programación.

```

Practica-9$
44 int pull = 34;
45 int pul2 = 35;
46
47 int Piloto1 = 36;
48 int Piloto2 = 37;
49 int Piloto3 = 38;
50 int Piloto4 = 39;
51 int Piloto5 = 40;
52 int Piloto6 = 41;
53
54 int Potenciometro1;
55 int SetMotor;
56 int ResetMotor;
57
58 int motor1 = 6;
59 int motor2 = 5;
60 int motorBanda = 7;
61 int sensorIR1 = 22;
62 int sensorIR2 = 27;
63 int emergencia = 29;
64 int FaroEmergencia;
65 int IR1;
66 int IR2;
67 float llenar;
68 int Frecuencia;
69 int valMotorbanda;
70 int M_banda;
71 int M_start_stop;
72 int M_llenado;
73 int M_llenado2;
74
75 int distancia;
76 int cont;
77
78 int val; // variable para lectura del estado del pin
79 int val2; // variable para leer el estado delayed/debounced
80 int botonStado; // variable para mantener el estado del botón
81 int botonCounter = 0 ; // contador para el número de pulsos de botón

```

- Configuramos los pines de entrada y salida.
- El pin del sensor esta dado por una entrada de resistencia o funcionamiento de pull up.
- Inicializamos los protocolos a usar.
- interrupts() es la inicialización de la interrupción externa de Arduino.
- Flujo es la función de interrupción dada por el sensor en el flanco de subida del estado, se dará cada vez que exista un pulso en el sensor,

gracias a la interrupción el programa principal no se verá afectado por demoras o delays incrementando el conteo de pulsos.

```
Practica-9 Arduino 1.8.16 (Windows Store 1.8.51.0)
Archivo Editar Programa Herramientas Ayuda
Practica-9$
86 void setup() {
87   //Configuracion e inicializacion de protocolos
88   //Configuracion de puertos de entrada y salidas digitales
89   pinMode(Filoto1 , OUTPUT);
90   pinMode(Filoto2 , OUTPUT);
91   pinMode(Filoto4 , OUTPUT);
92   pinMode(Filoto3 , OUTPUT);
93   pinMode(Filoto5 , OUTPUT);
94   pinMode(Filoto6 , OUTPUT);
95   pinMode(sensor , INPUT_PULLUP);
96
97   Ethernet.begin(mac, ip, gateway, subnet); // start ethernet interface
98   Serial.println("Ethernet interface started");
99   Serial.begin(9600);
100
101   lcd.init();
102   lcd.backlight();
103   pinMode(sensorIR1 , INPUT);
104   pinMode(sensorIR2 , INPUT);
105   pinMode(Selector1 , INPUT);
106   pinMode(Selector2 , INPUT);
107   pinMode(motor1 , OUTPUT);
108   pinMode(motor2 , OUTPUT);
109   pinMode(motorBanda , OUTPUT);
110
111   interrupts(); // Habilito las interrupciones
112   // Interrupción INT0, llama a la ISR llamada "flujo" en cada flanco de subida en el
113   // pin digital 2
114   attachInterrupt(digitalPinToInterrupt(sensor), flujo, RISING);
115   tiempoAnterior = millis();
116   pinMode(6 , OUTPUT);
117   pinMode(5 , OUTPUT);
118   pinMode(30 , OUTPUT);
119   pinMode(31 , OUTPUT);
120
121   botonStado = digitalRead(sensorIR2);
122
123
123
124 // Rutina de servicio de la interrupción (ISR)
125 void flujo()
126 {
127   pulsos++;
128   pulsos1++; // Incrementa en una unidad el número de pulsos
129 }
130
```

- En el programa principal se ejecutará la función lectura de flujo, cuyo funcionamiento se detalla más adelante.
- Se lee la distancia del nivel de tanque dándole una ganancia negativa, recordemos que estos sensores tienen un margen de error, para lo cual se debe escalar su salida.
- La lectura del tamaño de la botella puede ser medida y dada según las mediciones que se hagan físicamente y que está en la variable Tam_botella.
- Se declara las variables de lecturas digitales.
- Para esta práctica, los valores que serán leídos desde el PLC master Unitronics van desde: mb.MbData[21], hasta: mb.MbData[40], permitirán dar marcha y paro desde el PLC Unitronics, Siemens o desde Labview además de leer variables que pueden ser seteadas o de lectura (como la temperatura del sensor ubicado en PLC siemens), desde los PLC's o PC.

```

Practica-9$
131 void loop()
132 {
133   //Ejecucion de funcion de lectura de sensor de flujo
134   lectura_flujo();
135
136   //Lectura sensor Ultrasonico tanques
137   distancia = ultrasonic2.read(CM) - 1;
138   Serial.println(distancia);
139
140   //Lectura sensor Ultrasonico de botellas
141   int Tam_botella = ultrasonic1.read(CM);
142   Serial.println(cont);
143
144   //Lecturas Digitales
145   IR1 = digitalRead(sensorIR1);
146   IR2 = digitalRead(sensorIR2);
147   SetMotor = digitalRead(pul1);
148   ResetMotor = digitalRead(pul2);
149   ParoEmergencia = digitalRead(emergencia);
150
151
152   //Lectura Modbus Unitronics - Labview
153   int lab_SetMotor = Mb.MbData[21];
154   int lab_ResetMotor = Mb.MbData[22];
155   int lab_indFrecuencia = Mb.MbData[23];
156   int lab_frecuencia = Mb.MbData[24];
157   int lab_numBotellas = Mb.MbData[25];
158   int lab_resetSistema = Mb.MbData[26];
159   int lab_VelLlenado = Mb.MbData[27];
160   int lab_SetTemperatura = Mb.MbData[28];
161   int Siemens_Temperatura = Mb.MbData[29];
162   int Unit_start = Mb.MbData[30];
163   int Unit_stop = Mb.MbData[31];
164   int Siemens_Start = Mb.MbData[32];
165   int Siemens_Stop = Mb.MbData[33];

```

- En la condición para velocidad remota y local, si el dato de la variable lab_indFrecuencia es igual a “0”, entonces se podrá dar control de la frecuencia desde Arduino con el potenciómetro1, caso contrario la frecuencia del motor y banda transportadora será seteada desde el programa Labview.
- Se crean condiciones de Start y Stop local y remoto.
- Si la distancia de tanque2 es 2 o 3, la variable m_llenado se activa y se llena el tanque1 que anteriormente se vacío, para poder seguir llenando con liquido las botellas de la simulación de llenado.

```

Practica-9$
167 //Condicion para velocidad de linea remota y local
168 if (lab_indFrecuencia == 0) {
169   Potenciometro1 = analogRead(A1);
170   valMotorbanda = map(Potenciometro1, 0, 1023, 0, 255);
171   Frecuencia = map(Potenciometro1, 0, 1023, 0, 60);
172 } else {
173   Frecuencia = lab_frecuencia;
174   valMotorbanda = map(lab_frecuencia, 0, 60, 0, 255);
175 }
176
177 //Start Stop desde cualquier estacion
178 if (SetMotor == 1 or lab_SetMotor == 1 or Unit_start == 1 or Siemens_Start == 1) {
179   M_start_stop = 1;
180   buttonCounter = 0;
181 }
182
183 if (ResetMotor == 1 or ParoEmergencia == 0 or lab_ResetMotor == 1 or Unit_stop == 1 Siemens_Stop == 1) {
184   M_start_stop = 0;
185   M_banda = 0;
186 }
187
188 // Llenado automatico de tanque 1
189 if (distancia == 2 or distancia == 3) {
190   M_llenado = 1;
191 }
192 if (distancia == 18 or distancia == 19) {
193   M_llenado = 0;
194 }
195
196 if (M_llenado == 1) {
197   analogWrite(motor1, 255);
198 }
199 if (M_llenado == 0) {
200   analogWrite(motor1, 0);
201 }
202

```

- La condición de inicio y parada de banda transportadora, corresponde cuando los sensores conectados se activan, si la botella pasa por el sensor IR1 la banda se detiene mientras la temperatura que es controlada por el potenciómetro en PLC Siemens sea menor al set de temperatura dado en Labview.
- Si IR2 que su señal esta invertida, es decir cuando hay objetivo envía un “0” lógico, y el sensor de litros indica que es menor a los litros a llenar, entonces se enciende el motor de llenado de botellas y se detiene la banda transportadora.

```

Pracica-9 Arduino 1.8.16 (Windows Store 1.8.51.0)
Archivo Editar Programa Herramientas Ayuda
Pracica-9 $
203 // Condicion de inicio y parada de banda transportadora
204 if (M_start_stop == 1) {
205     digitalWrite(Piloto1, HIGH);
206     digitalWrite(Piloto4, LOW);
207
208     if (IR1 == 1 and Siemens_Temperatura < lab_SetTemperatura ) {
209         M_llenado2 = 0;
210         M_banda = 0;
211     } else {
212         if (IR2 == 0 and litros < llenar) {
213             analogWrite(motor2, lab_VelLlenado);
214             M_llenado2 = 1;
215             M_banda = 0;
216             cont++;
217         } else {
218
219             analogWrite(motor2, 0);
220             M_banda = 1;
221             pulsos_Acumulados = 0;
222             M_llenado2 = 0;
223         }
224     }
225 }
226 if (M_start_stop == 0) {
227     digitalWrite(Piloto1, LOW);
228     digitalWrite(Piloto4, HIGH);
229     analogWrite(motor2, 0);
230 }
231
232 if (M_banda == 1) {
233     analogWrite(motorBanda, valMotorbanda);
234 } else {
235     analogWrite(motorBanda, 0);
236 }
237

```

- Si desde Labview llega un pulso de reset sistema, los pulsos acumulados1 que son los pulsos correspondientes a el conteo de litros llenados y el contador de botellas se reinician a 0.
- El sensor ultrasónico ubicado en la mitad de la banda transportadora es aquel que da el tamaño de la botella que pasa por ese instante y que dependiendo de ello llena la botella en 0.20 litros o en 0.50 litros.
- Cuando la botella se termine de llenar y siga el proceso, este aumenta el contador que funciona de la siguiente manera: guarda la variable en val, luego espera 10 milisegundos a espera del cambio de esta variable y la guarda en val2, esto es para obtener un pulso en el flanco de la señal de bajada es decir cuando el sensor indica un 1 positivo no cuenta, pero cuando de 1 pasa a cero buttonCounter suma un más uno hasta que la variable vuelva a ser 1. Esta condición se cumple mientras el sensor sea de lectura normalmente cerrada.

- Si el contador de botellas indica que es mayor al valor seteado en Labview, entonces la banda transportadora se detiene y enciende un indicador de proceso listo.

```

Pracica-9 Arduino 1.8.16 (Windows Store 1.8.51.0)
Archivo Editar Programa Herramientas Ayuda
Pracica-9 $
238 //Resetea los valores de litros totales y contador de botellas
239 if (lab_resetSistema == 1) {
240     pulsos_Acumulados1 = 0;
241     buttonCounter = 0;
242 }
243
244 //Condicion de llenado de botellas dadas por el sensor Ultrasonico
245 if ( Tam_botella == 14 ) {
246     llenar = 0.20;
247     digitalWrite(Piloto6, HIGH);
248 }
249 if ( Tam_botella == 8 or Tam_botella == 7 ) {
250     llenar = 0.50;
251     digitalWrite(Piloto6, LOW);
252 }
253
254 // Contador de botellas que pasan por el sensor IR2
255 val = digitalRead(sensorIR2); // lee el valor de entrada y almacénalo en val
256 delay(10); // 10 milisegundos son una cantidad de tiempo buena
257 val2 = digitalRead(sensorIR2); // lee la entrada otra vez para comprobar saltos
258 if (val == val2) { // asegurar que conseguimos 2 lecturas constantes
259     if (val != botonStado) { // el estado de botón ha cambiado!
260         if (val == LOW) { // compruebe si el botón es presionado
261             buttonCounter++;
262         }
263     }
264     if (buttonCounter > lab_numBotellas) {
265         digitalWrite(Piloto2, HIGH);
266         M_start_stop = 0;
267         M_banda = 0;
268     } else {
269         digitalWrite(Piloto2, LOW);
270     }
271     botonStado = val; // guardar el nuevo estado en la variable
272 }
273

```

- Las variables obtenidas en Arduino son enviadas al PLC Master Unitronics.

```

273
274 //Escritura Arduino - Master - Unitronics
275 Mb.MbData[0] = Frecuencia;
276 Mb.MbData[1] = litros_min;
277 Mb.MbData[2] = buttonCounter;
278 Mb.MbData[3] = llenar * 100; // Solo admite datos enteros;
279 Mb.MbData[4] = litros * 100;
280 Mb.MbData[5] = litros1 * 100;
281 Mb.MbData[6] = distancia - 1;
282 Mb.MbData[7] = M_start_stop;
283 Mb.MbData[8] = 1;
284 Mb.MbData[9] = ParoEmergencia;
285 Mb.MbData[10] = M_llenado;
286 Mb.MbData[11] = M_llenado2;
287 Mb.MbData[12] = M_banda;
288 Mb.MbData[13] = IRL;
289 Mb.MbData[14] = IR2;
290 Mb.MbData[15] = digitalRead(Piloto2);
291
292 Mb.MbsRun();
293 delay(200);
294 }
295

```

- Esta función permite obtener la lectura del flujo, el cronometro millis se activa mientras la resta con el tiempo anterior sea menor a 1000, es decir que el tiempo anterior que empieza en 0 este empieza a contar y da paso al proceso.
- El tiempo anterior toma la lectura del cronometro millis(), y el pulso acumulado aumenta.

- La fórmula es: $k = (\text{número de pulsos} / \text{Volumen})$, donde k es el factor de conversión el número de pulsos es el medido por el sensor y el volumen aquel que se mide.
- La cantidad de pulsos vienen a ser la cantidad de litros acumulados en un instante de tiempo, para escalar esta señal se deben hacer muestras tanto con el valor medido físicamente y el valor dado por el sensor, a esto se obtiene un promedio y se obtiene el valor de k con la formula antes mencionada, con este dato se puede tener el valor de litros_min usando la formula: $\text{litros_min} = (\text{pulsos acumulados} * 1000 / k)$, donde k hallado por fórmula o por método de tanteo, midiendo el volumen de tanque físico da como resultado 7.5, así obteniendo el valor de velocidad de llenado, sabiendo que el caudal de la bomba sumergible es de 240 L/H y que su conversión a Litros/min es de 4 L/min, y que la capacidad del tanque para esta práctica es de 2 Litros, también se puede obtener los litros que se han acumulado en el proceso.

```

300
301 void lectura_flujo() {
302   if (millis() - tiempoAnterior > 1000){
303     lcd.clear();
304     // Realiza los cálculos
305     tiempoAnterior = millis(); // Actualizacion el nuevo tiempo
306     pulsos_Acumulados += pulsos; // Número de pulsos acumulados
307     litros_min = (pulsos * 1000 / 7.5); // Q = frecuencia * 1050/ 7.5 (L/Hora)
308     litros = pulsos_Acumulados * 1.0 / 510; // Cada 510 pulsos son un litro
309     pulsos = 0; // Pongo nuevamente el número de pulsos a cero
310
311     tiempoAnterior = millis();
312     pulsos_Acumulados1 += pulsos1; // Número de pulsos acumulados
313     litros_min1 = (pulsos1 * 1000 / 7.5); // Q = frecuencia * 1050/ 7.5 (L/Hora)
314     litros1 = pulsos_Acumulados1 * 1.0 / 510; // Cada 510 pulsos son un litro
315     pulsos1 = 0; // Pongo nuevamente el número de pulsos a cero
316
317   }
318 }

```

- Finalmente, los datos obtenidos son mostrados en pantalla LCD de Arduino.



Practica-9 Arduino 1.8.16 (Windows Store 1.8.51.0)

Archivo Editor Programa Herramientas Ayuda

Practica-9 §

```

309 //Contador de botellas
310 lcd.setCursor(0, 0);
311 lcd.print("Bot: ");
312 lcd.setCursor(5, 0);
313 lcd.print(buttonCounter);
314 lcd.setCursor(7, 0);
315 lcd.print("de: ");
316 lcd.setCursor(11, 0);
317 lcd.print(lab_numBotellas);
318 lcd.setCursor(0, 2);
319
320 // Imprimo el caudal en L/m
321 lcd.print(litros_min / 1000);
322 lcd.setCursor(4, 2);
323 lcd.print(" L/m");
324 lcd.setCursor(0, 3);
325
326 // Imprimo el número de litros acumulados por llenado de botellas
327 lcd.print(litros);
328 lcd.print(" L");
329
330 // Imprimo el número de litros acumulados totales
331 lcd.setCursor(9, 3);
332 lcd.print(litros1);
333 lcd.print(" L tot");
334
335 // Imprimo el volumen del reservorio tanque l
336 lcd.setCursor(14, 0);
337 lcd.print("LT:");
338 lcd.setCursor(18, 0);
339 lcd.print(distancia);
340
341 //Imprimo la frecuencia de motor Trifasico y banda transportadora
342 lcd.setCursor(13, 1);
343 lcd.print("FR:");
344 lcd.setCursor(17, 1);
345 lcd.print(Frecuencia);
346

```

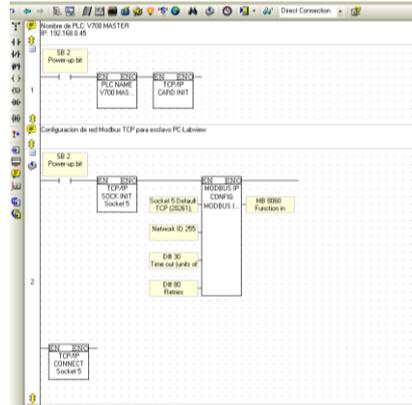
```

346
347 // Imprimo el valor a ser llenado en botella|
348 lod.setCursor(0, 1);
349 lod.print("Val L: ");
350 lod.setCursor(8, 1);
351 lod.print(llenar);
352
353 }
354 }

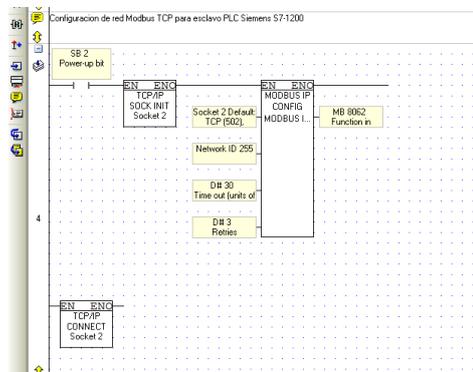
```

2. Desarrollo en Unitronics

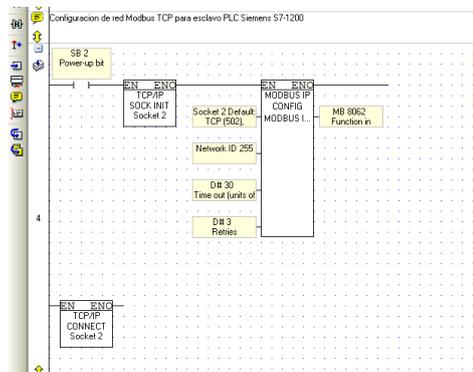
- Se configura la dirección IP del PLC en: 192.168.0.45
- Se configura la red Modbus para esclavo PC-Labview usando el Socket 5



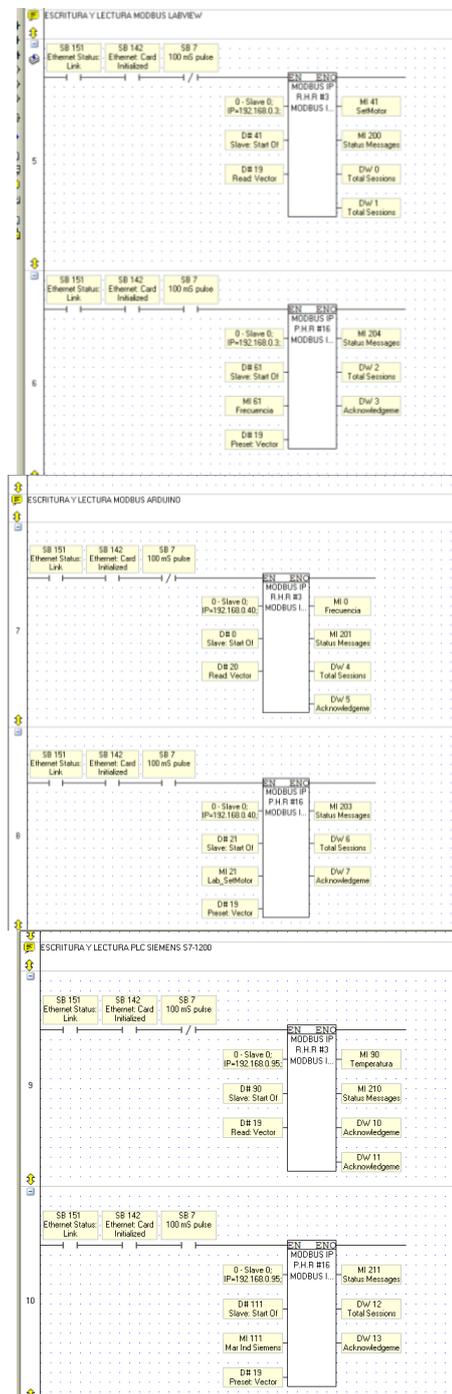
- Se configura la red Modbus para esclavo Arduino Ethernet usando el Socket 4.



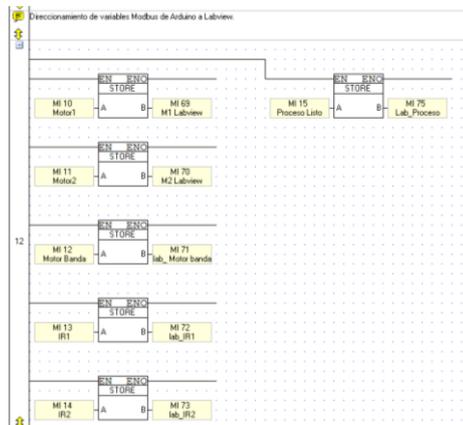
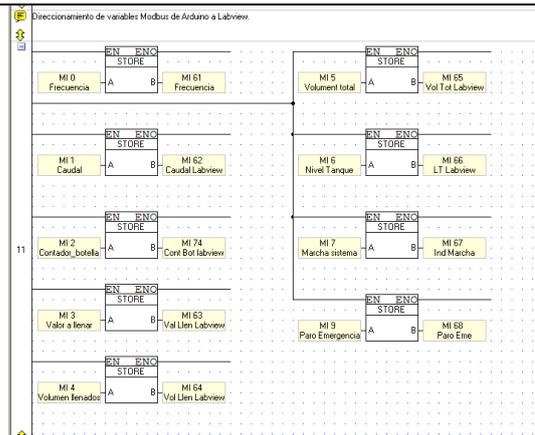
- Se configura la red Modbus para esclavo PLC Siemens S7-1200 usando el Socket 2.



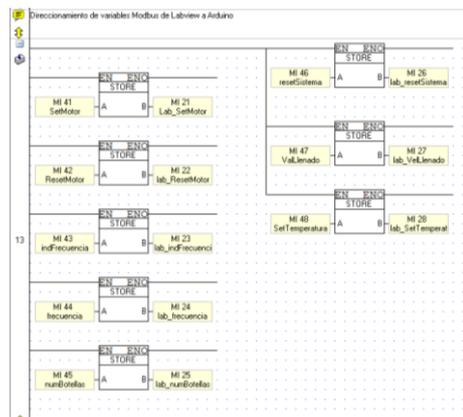
- Configuración de lectura: Read Holding Register y escritura: Preset Holding Register, para dispositivos esclavos, Arduino, Siemens PLC S7-1200 y PC – Labview.



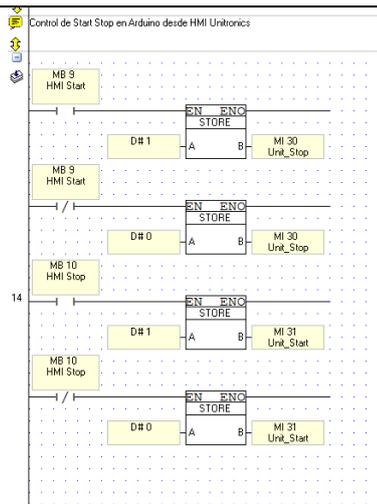
- Con el bloque Store se direccionan las variables para Labview, que vienen desde Arduino.



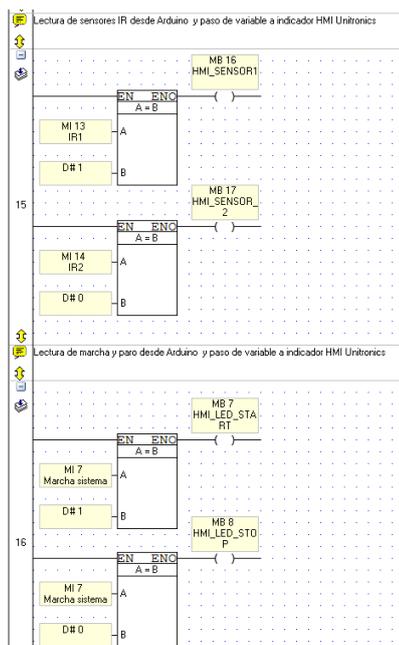
- Con el bloque Store se direccionan las variables de Modbus que vienen de sistema Labview para control en Arduino.



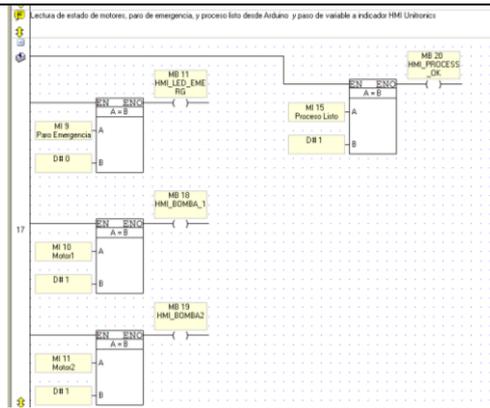
- Las acciones de botones en pantalla HMI Unitronics, son direccionadas con bloque store enviando bits hacia Arduino para control de Start y Stop de sistema.



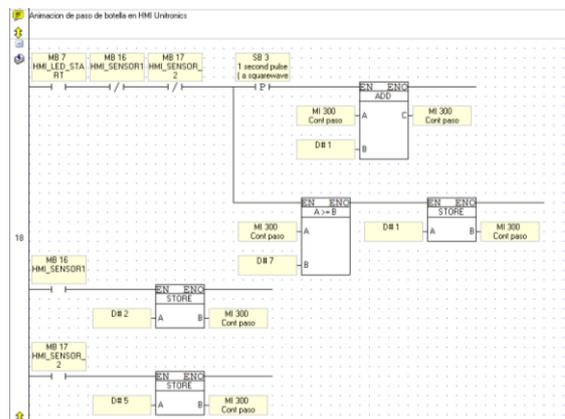
- Cuando uno de los sensores IR conectados a la banda transportadora se activan, se encienden los indicadores LED en la pantalla HMI de Unitronics y cuando Arduino da paso a la marcha del sistema este responde para activar indicadores de Start y Stop en HMI Unitronics.



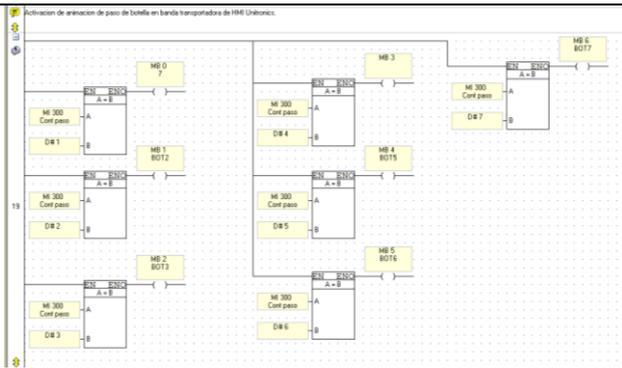
- Lectura Modbus desde Arduino de estado de encendido de motores, activación de paro de emergencia e indicador de proceso Listo para visualización en pantalla HMI Unitronics.



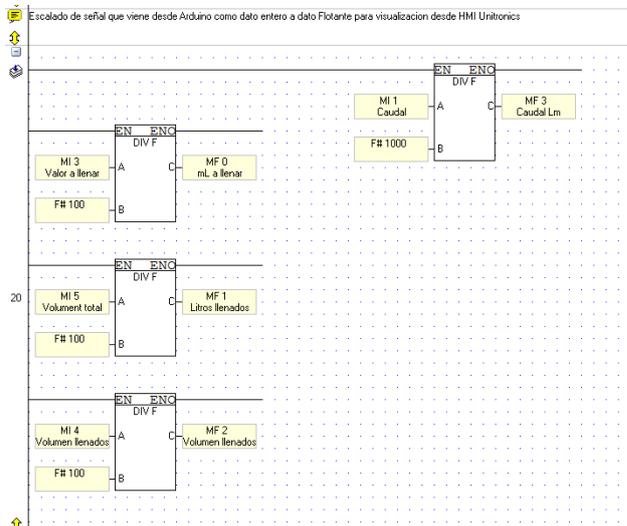
- Para animación de que botellas pasan por la banda transportadora en PLC Unitronics, el proceso se activa cuando el indicador de Start esta activado y mientras el sensor IR1 no este activado, entonces el pulso de 1 Segundo se activa para empezar a contar.
- En HMI Unitronics se agregaron seis botellas y cada que el contador pase por el número correspondiente a la botella colocada, esta imagen de tipo Bit se activa haciendo que la botella aparezca y desaparezca según el contador, este contador se reinicia cuando sea mayor o igual a 7.
- Si uno de los sensores IR se activa entonces el contador principal se desactiva y el contador se actualiza al número de botella que corresponde al sensor que se activó en ese momento.



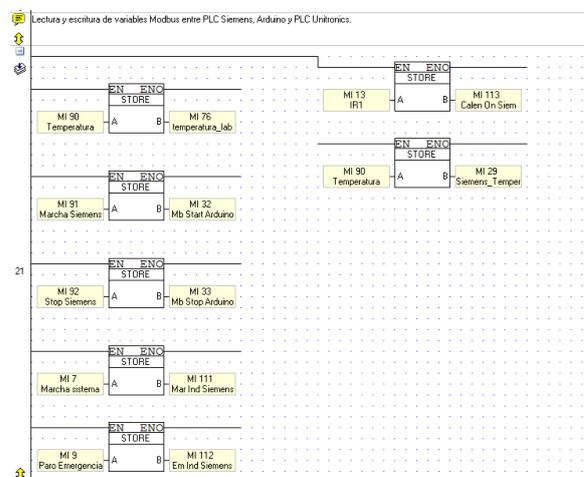
- Cuando el contador este en un número que corresponde a una de las botellas, entonces estas aparecerán, simulando la acción de paso de botella en la banda transportadora.



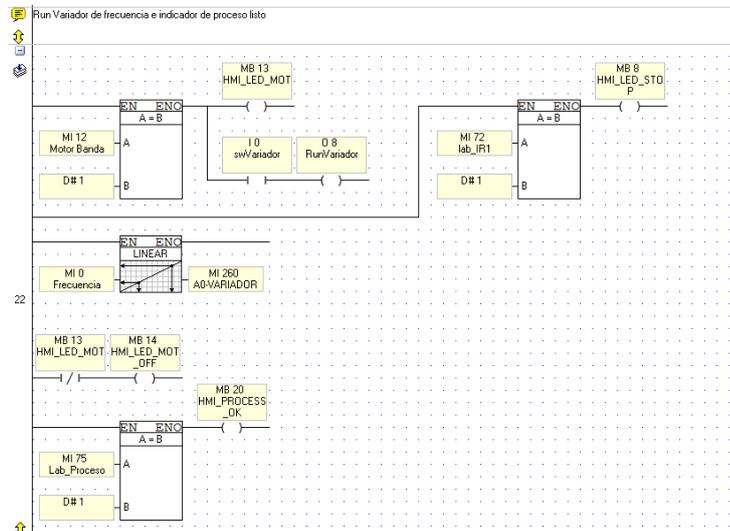
- Modbus TCP solo envía números enteros positivos y negativos, pero no decimales, para ello el valor se envía desde Arduino como un entero y este se divide con bloques matemáticos en Unitronics para obtener su valor correspondiente a su unidad en decimal.



- Lectura y escritura de variables que vienen desde Siemens y que son direccionadas a control en Arduino y también se comparten variables para indicadores desde estación Siemens.

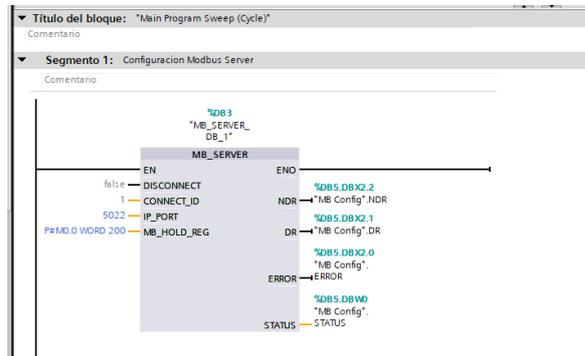


- Por último, la velocidad del motor trifásico conectado al variador de frecuencia es seteado con la velocidad dada en Hz que vienen desde control PC-Labview.

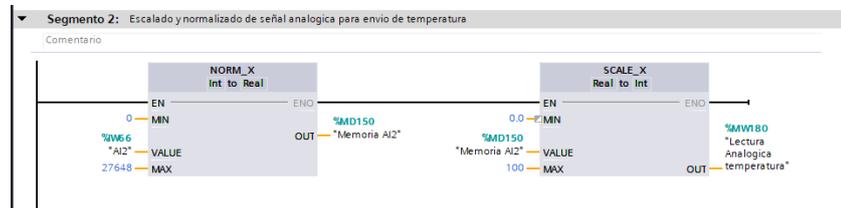


3. Desarrollo en PLC Siemens S7-1200

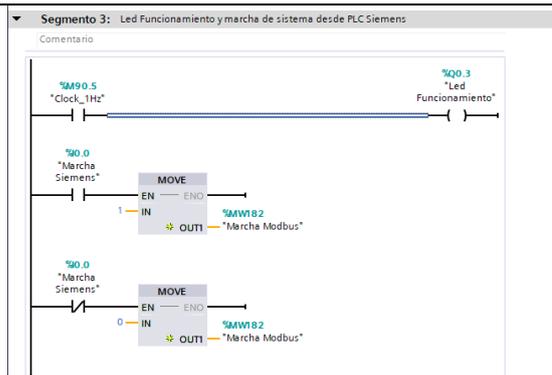
- Se configura el bloque se Modbus Server, la dirección IP del dispositivo debe estar en: 192.168.0.95



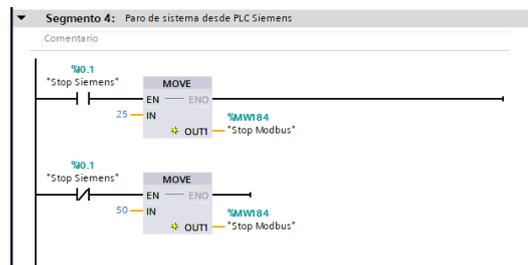
- Los bloques de Normalizado y Escalado obtienen la señal simulada de sensor de temperatura que va de 0 a 100 Grados Celcius.



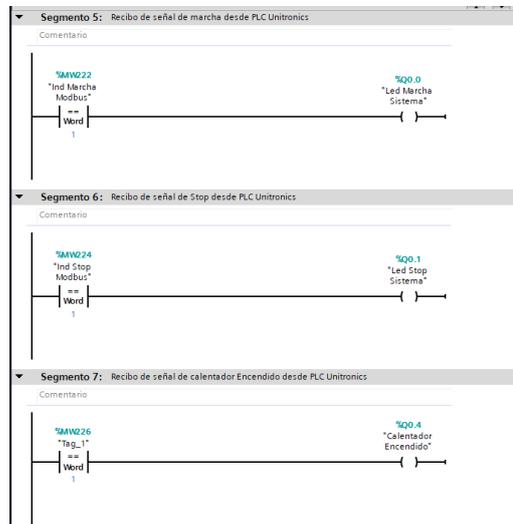
- Con el bloque Move se direcciona la variable de marcha Siemens a MW182 Modbus para Unitronics.



- Con el bloque Move direccionamos la variable de Stop Siemens a MW182 Modbus para Unitronics.



- Desde Unitronics se recibe la señal de indicadores de marcha y paro comparando los valores para encender los indicadores LED en el tablero del PLC Siemens.



4. Desarrollo en PC – Labview

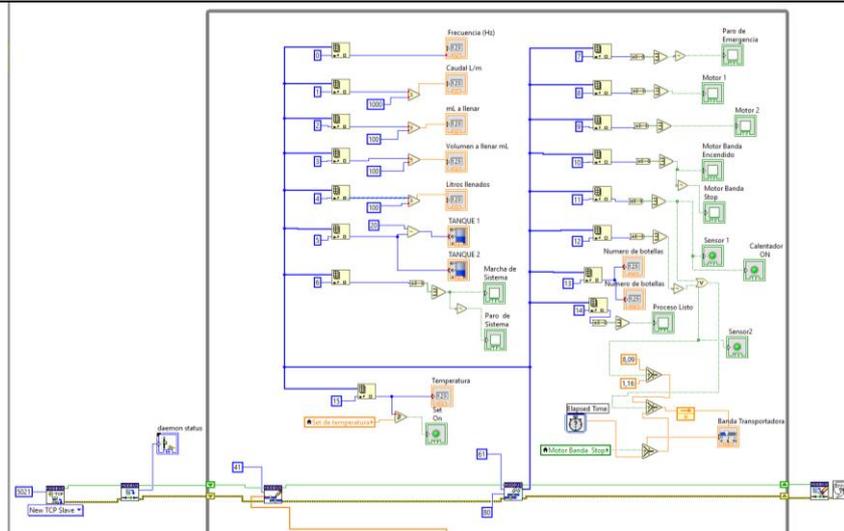
- La pantalla principal en Labview será la cual controle todo el sistema, el primer panel a la izquierda el usuario deberá ingresar el número de botellas a fabricar, podrá tener control manual o remoto de la velocidad de la banda transportadora y el motor trifásico, dar marcha y paro al sistema, visualizar indicadores de marcha, paro y paro de emergencia, y dar reset al sistema una vez cumplido el proceso de llenado del número de botellas a llenar.

- El segundo panel el usuario debe ingresar el set de temperatura, que simula el proceso de la fundición, formado y enfriado de resina PET (botella plástica de Teraflato de Polietileno). Este proceso es controlado por el PLC Siemens donde la temperatura será la lectura analógica y la manipulación de esta con potenciómetro desde el tablero.
- El tercer panel se visualiza la información siguiente: los mililitros a llenar en ese instante dependiendo del tamaño de la botella, los litros llenados acumulados, el caudal dado por el sensor al momento de llenar y el volumen a llenar en la botella.
- El cuarto panel será para control de llenado dada la velocidad de la bomba centrífuga de 0 a 100%.
- Por último, los indicadores tanto del número de botellas fabricadas, indicador de proceso listo, indicadores de motores encendidos y sensores activados.

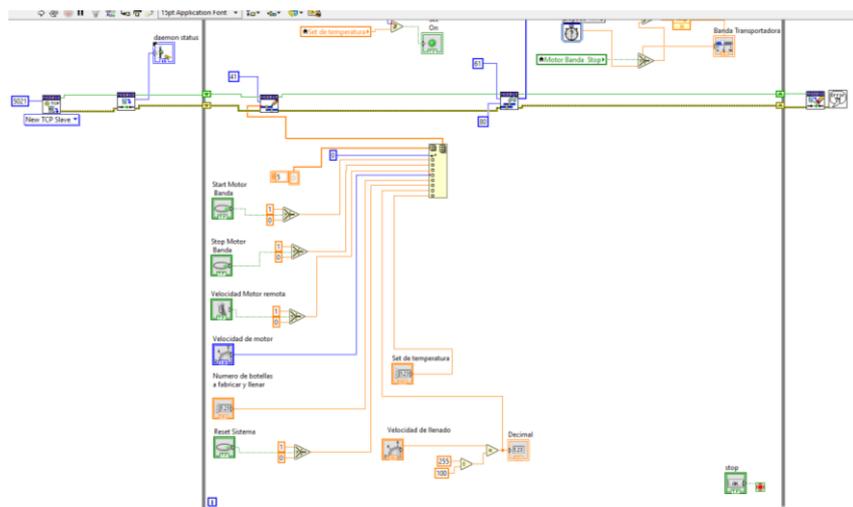


- El diagrama de bloques que se presenta está constituido por una estructura de While Loop que permitirá que los datos se muestren mientras el proceso está en ejecución en un bucle de repetición, se usan los bloques ya mencionados (Ver práctica número 3). Como el de comunicación Modbus, en el cual se establece el programa como Master de Modbus TCP/IP y se leen 20 direcciones desde M61 Start Address Modbus, se convierten los datos y se separa el array para ser leídos por indicadores programados en la pantalla principal o front panel, cuando se escribe desde Labview se usa un constructor de array que nos permitirá ingresar varios elementos y enviarlos por medio de un solo bloque de escritura y que se empieza a escribir desde la dirección M41 configurada como Start Address Modbus.

Recordar que los datos leídos vienen desde PLC Unitronics, todos los dispositivos apuntan a este PLC y las variables son direccionadas hacia el control del proceso, en este caso es Arduino.



- Los datos escritos hacia Unitronics y luego direccionados hacia control Arduino y control de temperatura en Siemens, son los de control de Motor de banda transportadora, frecuencia a motor trifásico, velocidad de llenado de bomba centrífuga en tanques, set de temperatura hacia PLC Siemens, número de botellas a fabricar, entre otras.



ACTIVIDADES POR DESARROLLAR

- Considerar colocar sensores de nivel en interfaz en Labview para mayor comprensión del usuario.
- Realizar un informe detallado con los resultados obtenidos en la práctica.
- Realizar un inicio y parada de proceso desde PLC Siemens.
- Realizar ingreso de datos de usuario desde PLC Unitronics.

RESULTADOS OBTENIDOS:

El proceso empieza cuando el usuario desde la interfaz gráfica Labview, ingresa los datos de numero de botellas a fabricar, la velocidad de motor de banda transportadora, el set de temperatura, y la velocidad de llenado de botellas.

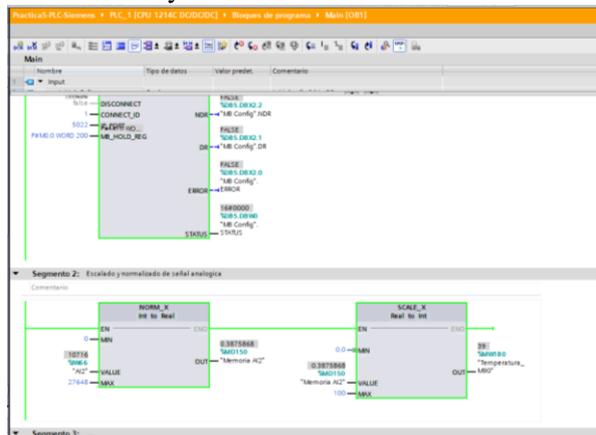
Cuando el sensor 1 detecta una botella la banda transportadora se detiene hasta que el sensor simulado en PLC Siemens de un valor igual o mayor al set de temperatura ingresado por el usuario en Labview, luego la botella avanza hasta sensor 2.

Si la botella que pasa por sensor ultrasónico es de un tamaño pequeño el programa llena hasta 0.20 Litros, si es grande lleno hasta 0.50 Litros.

Al culminar el contador se incrementa hasta que el proceso cumpla con la condición de fabricación de las botellas indicadas.



Desde PLC Siemens se obtienen los resultados de conexión de programa y la tabla de observación con las variables escritas y leídas desde PLC Unitronics.



Id	Nombre	Descripción	Formato visual	Valor de observación	Valor de forzaje	Comentario
1	"Temperatura_M90"	NAM180	DEC+1	39		
2	"M1_Start"	NAM182	DEC+1	0		
3	"M1_Stop"	NAM184	DEC+1	0		
4	"M1_Start"	NAM186	DEC+1	0		
5	"M1_Stop"	NAM188	DEC+1	0		
6	"M1_Start"	NAM190	DEC+1	0		
7	"M1_Stop"	NAM192	DEC+1	0		
8	"M1_Start"	NAM194	DEC+1	0		
9	"M1_Stop"	NAM196	DEC+1	0		
10	"M1_Start"	NAM198	DEC+1	0		
11	"M1_Stop"	NAM200	DEC+1	0		
12	"M2_Start"	NAM202	DEC+1	0		
13	"M2_Stop"	NAM204	DEC+1	0		
14	"M2_Start"	NAM206	DEC+1	0		
15	"M2_Stop"	NAM208	DEC+1	0		
16	"M2_Start"	NAM210	DEC+1	0		
17	"M2_Stop"	NAM212	DEC+1	0		
18	"M2_Start"	NAM214	DEC+1	0		
19	"M2_Stop"	NAM216	DEC+1	0		
20	"M2_Start"	NAM218	DEC+1	0		
21	"M2_Stop"	NAM220	DEC+1	0		
22	"Marcha Sistema"	NAM222	DEC+1	1		
23	"Paro Emergencia Sistema"	NAM224	DEC+1	1		
24	"Encender Calentador"	NAM226	DEC+1	0		

- La pantalla HMI de Unitronics queda de la siguiente manera:



CONCLUSIONES:

Se demostró la interacción entre los dispositivos esclavos, el proceso incluye el funcionamiento de timers en Arduino, procesos lógicos en PLC Unitronics, y manejo de los sensores conectados en la banda transportadora y tanques, el PLC Unitronics es quien lee las variables de los dispositivos y los redirecciona a PC Labview para un control completo de parte del Usuario.

RECOMENDACIONES:

- Revisar las conexiones y cableados desde Arduino a Banda transportadora.
- Verificar direcciones IP de todos los dispositivos antes de comenzar para evitar errores en la lectura de datos.
- Realizar una tabla con los direccionamientos Modbus TCP para una mejor comprensión en esta práctica.

Docente/Técnico Docente: _____

Firma: _____

4.10 Práctica 10

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Redes III / Sistemas Microprocesados	
NRO. PRÁCTICA:	10	TÍTULO PRÁCTICA: Diseño de controlador Fuzzy Logic para comparación y estudio de la práctica 7.	
OBJETIVOS: <ol style="list-style-type: none"> 1. Implementar una red MODBUS TCP/IP entre PLC Unitronics y Tablero Arduino. 2. Crear un control Fuzzy Logic para llenado de tanque controlado desde Tablero Arduino. 3. Interactuar con variables y graficas de control desde PLC Unitronics. 4. Controlar motor variando su velocidad dada la respuesta de control desde tablero Arduino. 			
Dev:	1. Introducción: El programa desarrolla un control Fuzzy Logic de llenado de tanque que es desarrollado en el tablero Arduino, el PLC Unitronics leerá las variables como: Setpoint, Input, Output, Kp, Kd, entre otras, y se ajustará la variable de salida para poder controlar el Motor mediante variador de velocidad simulando que este es quien funciona como bomba centrifuga, cabe recalcar que los dos motores conectados a Arduino son de 12VDC y serán controlados desde tablero Arduino. Considerar colocar un accionador de descarga de tanque, y un accionador de activación de perturbación para probar la estabilidad del control Fuzzy Logic. El tablero de PLC Unitronics debe estar cargado con el archivo de la práctica 7.		
	2. Desarrollo en Arduino: <ul style="list-style-type: none"> ● MgsModbus será la librería que contiene la configuración y puesta en marcha de Modbus TCP/IP para Arduino. ● LiquidCrystal_I2C permitirá la conexión con el LCD mediante protocolo I2C la cual nos permitirá usar solo dos cables de conexión y ahorrar espacio en entradas o salidas físicas de Arduino. ● Ethernet y SPI: Serán las que permitirán la conexión TCP/IP mediante el Shield Ethernet. ● NewPing: librería que contiene la configuración de lectura de los sensores ultrasónicos. ● Fuzzy: librería que contiene la configuración que permitirá el control cuando ocurran errores o perturbaciones en el sistema con Sistema de control Fuzzy Logic. ● TimerOne: Librería para interrupción interna de Arduino. ● tclab_lib: Librería para actualizar valores de error pasados en el control Fuzzy y la función de control PID Fuzzy. 		

```

Practica-10$ fuzzy.cpp fuzzy.h i2c0_i2c.cpp lcd0_lcd.h
1 ///UNIVERSIDAD POLITECNICA SALESIANA
2 ///TESIS DE GRADO DE INGENIERIA ELECTRONICA
3 ///AUTOR: NICOLAS CRESCO DELGADO
4 ///TEMA: PRACTICA 10: Diseño de controlador Fuzzy Logic para comparacion
5 /// estudio de la practica 7.
6
7 //Librerias
8 #include <TimerOne.h>
9 #include <NewPing.h>
10 #include "tclab_lib.h"
11 #include "fuzzy.h"
12
13 #include "MgsModbus.h"
14 #include <LiquidCrystal_I2C.h>
15 #include <Ethernet.h>
16 #include <SPI.h>
17

```

Definición de pines, configuraciones y asignación de variables:

- Se define el bus por el cual se van a conectar los sensores HC-SR04, el pin ECHO y TRIGGER del sensor ultrasónico, la máxima distancia que tiene el sensor ultrasónico como rango, y luego se configura el sonar con las variables ya declaradas.
- Se asigna la dirección HEX por defecto 0x27 del I2C para el LCD y el tipo de LCD para este caso con dimensiones de 20x4.
- Se realiza la configuración de direcciones Ethernet y la MAC del dispositivo, además de declarar la variable de Mb para Modbus.

```

Practica-10-Arduino Arduino 1.8.19 (Windows Store 1.8.57.0)
Archivo Editar Programa Herramientas Ayuda
Practica-10-Arduino
14 //Definicion de pines y configuracion de sensor Ultrasonico
15 #define TRIGGER_PIN 16
16 #define ECHO_PIN 15
17 #define MAX_DISTANCE 200
18 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
19
20 //Definicion y direccionamiento de de Pantalla LCD I2C
21 //como pantalla de 20x4
22 LiquidCrystal_I2C lcd(0x27, 20, 4);
23
24 // Definicion y configuracion de comunicacion Modbus TCP
25 //via puerto Ethernet
26 MgsModbus Mb;
27 byte mac[] = {0xFE, 0x62, 0xC7, 0xDC, 0xE9, 0xC4 };
28 IPAddress ip(192, 168, 1, 40);
29 IPAddress gateway(192, 168, 1, 1);
30 IPAddress subnet(255, 255, 255, 0);
31

```

- Las variables para el sistema Fuzzy vienen dadas por la necesidad del sistema, en este caso tenemos entrada, salida, un vector de error que indica el error presente y el error pasado para poder calcular la derivada del error, un vector de control que es el vector de salida del controlador, también de dos posiciones con la salida actual y la salida pasada para poder calcular la integral del control, la longitud del tamaño del vector de salida y error, los universos de discurso para las señales de error, derivada de error y la salida, el tiempo discreto y variables de representación.

```

=====
VARIABLES GLOBALES
=====
//Variables Fuzzy
float Input, Output = 0.0;
float e[2] = {0, 0}; // Vector de error - Pos 1,2 Error presente y Error Pasado
float u[2] = {0, 0}; //Vector de Ley de Control - Accion de control presente y pasada
int kU = sizeof(u) / sizeof(float) - 1; //Longitud de tamaño de vector de salida control
int kE = sizeof(e) / sizeof(float) - 1; // Longitud de tamaño de vector de error
//Universos de discursos error, derivada de error y ley de control
float discurso_e = 100, discurso_de = 1, discurso_u = 3;
float der; // Variable derivada del error
float lu; //Variable de calculo de control Fuzzy
float Ts = 3; //Periodo de Muestreo
int distancia; // Variable de sensor de entrada
double Setpoint; //SetPoint para fuzzy

```

- Finalmente se declaran las variables y se asignan los pines de entradas y salidas, para esto ver la tabla de entradas y salidas de Arduino en el Anexo: Arduino.

```

54
55 //Variables de control
56 int motor1 = 6;
57 int motor2 = 5;
58
59 int PinEmergencia = 29;
60 int ParoEmergencia;
61
62 int Selector1 = 30;
63 int Selector2 = 31;
64 int Sel1;
65 int Sel2;
66
67 int Piloto1 = 36;
68 int Piloto2 = 37;
69 int Piloto3 = 38;
70 int Piloto4 = 39;
71 int Piloto5 = 40;
72 int Piloto6 = 41;
73
74 double Potenciometro1;
75 double Potenciometro2;
76 double Potenciometro3;
77 double Potenciometro4;
78
79 int Frecuencia;
80 int paroUnitronics;
81 int paroHMI;
82
83 unsigned long TiempoAnterior1 = 0;
84 int HMI_Start = 0;

```

Configuración inicial del programa:

- Se inicializa el LCD, se enciende la luz de fondo y se escribe un mensaje colocando el cursor en la posición 1,0.
- Se inicializa el puerto serial y el Ethernet, colocando la dirección mac, IP, Gateway, subnet, anteriormente ya declaradas.
- Se establecen los pines de entrada y salida, Output para salidas digitales e Input para entradas digitales.
- Se inicializa las salidas Piloto 3 en '0' o falso.
- Se inicializa el timer en 3 segundos para ir actualizando el tiempo de muestreo de nuestro controlador Fuzzy.

```

141 void setup() {
142
143
144   lcd.init();
145   lcd.backlight();
146
147   Ethernet.begin(mac, ip, gateway, subnet);
148   Serial.println("Ethernet interface started");
149
150   //Configuración e inicialización de protocolos
151   //Configuración de pines de entrada y salidas digitales
152   pinMode(Selector1 , INPUT);
153   pinMode(Selector2 , INPUT);
154   pinMode(PinEmergencia , INPUT);
155   pinMode(Piloto1 , OUTPUT);
156   pinMode(Piloto2 , OUTPUT);
157   pinMode(Piloto4 , OUTPUT);
158   pinMode(Piloto3 , OUTPUT);
159   pinMode(Piloto5 , OUTPUT);
160   pinMode(Piloto6 , OUTPUT);
161   pinMode(motor1 , OUTPUT);
162   pinMode(motor2 , OUTPUT);
163
164   //Configuramos el puerto serial
165   Serial.begin(9600);
166   Piloto3 = 0;
167
168   //Valor máximo del Timer es 0.3 Segundos
169   Timer1.initialize(300000); //Configura el TIMER en 3 Segundos
170   Timer1.attachInterrupt(sampleTime); //Configura la interrupción del Timer 1
171
172 }
173

```

- **Variables lingüísticas de entrada:**

- **Entrada de Error:**

- **eNG** (Error negativo grande) Sucede cuando el valor de salida del error es mayor del valor que se tiene de referencia.
- **eNM** (Error negativo medio) Sucede cuando el valor de salida del error es medianamente mayor al valor que se tiene de referencia.
- **eNP** (Error negativo pequeño) Sucede cuando el valor de salida del error es ligeramente mayor pero no tiene mucha diferencia con relación a la señal de referencia.
- **eZ** (Error cero) La salida de error es igual a la referencia.
- **ePP** (Error positivo pequeño) La salida de error es menor sin mucha diferencia a la referencia.
- **ePM** (Error positivo medio) El error esta medianamente menor al valor de referencia.
- **ePG** (Error positivo grande) La salida de error es menor al valor de referencia.

- **Entrada de Derivada de Error:**

- **deNG** (Derivada error negativo grande) Cuando la salida tiene una pendiente positiva.
- **deNM** (Derivada error negativo mediano) Cuando la salida tiene una pendiente medianamente positiva
- **deNP** (Derivada error negativo pequeño) Cuando la salida tiene una salida positiva, pero esta también constante.
- **deZ** (Derivada error cero) Cuando la pendiente de salida es una constante.
- **dePP** (Derivada error positivo pequeño) Cuando la pendiente de salida es negativa que tiende a ser constante.
- **dePM** (Derivada error positivo mediano) Cuando la pendiente de salida es medianamente negativa.
- **dePG** (Derivada error positivo grande) Cuando la salida tiene una pendiente totalmente negativa.

- **Salida de control:**

- **UFNG** (Salida negativa grande) en este punto el valor de salida necesita ser disminuido de manera rápida, este puede ser la frecuencia de un motor.

- **UFNM** (Salida negativa mediana) en este punto el valor de salida necesita ser disminuido de forma rápida, pero con pequeñas limitaciones.
- **UFNP** (Salida negativa pequeña) en este punto el valor de salida baja constantemente.
- **UFZ** (Salida cero) aquí se deja de bajar la señal de salida y dejar constante.
- **UFPP** (Salida positiva pequeña) en este punto se sube de forma muy lenta el valor de salida.
- **UFPM** (Salida positiva mediana) en este punto el valor de salida debe incrementarse de manera rápida.
- **UFPG** (Salida positiva grande) el valor de subida sube hasta su punto máximo.

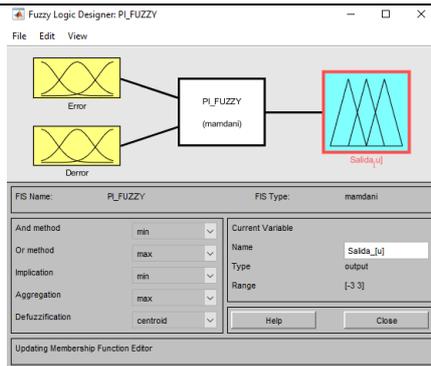
(se puede usar de referencia la salida de frecuencia de motor trifásico o PWM de salida de Arduino)

- Representación de funciones de control Fuzzy en Matlab:

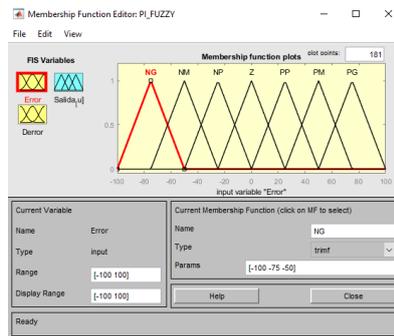
Para este controlador donde sus variables de entrada y salida son de 7 variables lingüísticas, se tendrán entonces 49 reglas de inferencia, para ello se realiza la siguiente tabla llamada FAM que involucra todas las combinaciones posibles que puede presentarse en el proceso, recordar que para el control PID lo que se quiere es tener una acción en la salida si el error está por encima o por debajo de la consigna, haciendo que esta acción de salida mantenga el error de sistema en cero.

e/de	deNG	deNM	deNP	deZ	dePP	dePM	dePG
eNG	UFNG	UFNG	UFNG	UFNG	UFNM	UFNP	UFZ
eNM	UFNG	UFNG	UFNG	UFNM	UFNP	UFZ	UFPP
eNP	UFNG	UFNG	UFNM	UFNP	UFZ	UFPP	UFPM
eZ	UFNG	UFNM	UFNP	UFZ	UFPP	UFPM	UFPG
ePP	UFNM	UFNP	UFZ	UFPP	UFPM	UFPG	UFPG
ePM	UFNP	UFZ	UFPP	UFPM	UFPG	UFPG	UFPG
ePP	UFZ	UFPP	UFPM	UFPG	UFPG	UFPG	UFPG

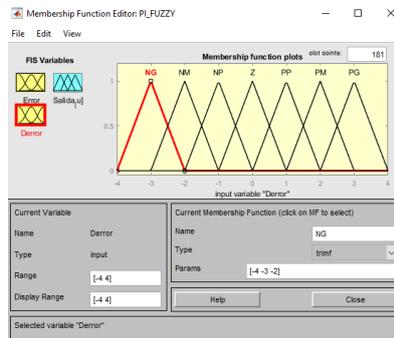
Desde Matlab entonces tenemos dos entradas: Error y Error derivativo, y su salida Salida [u]:



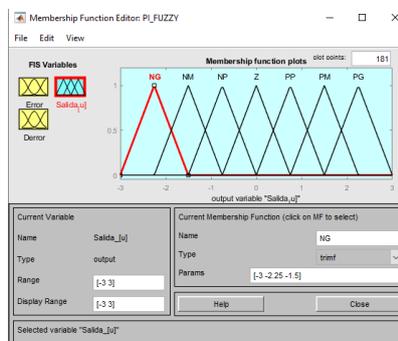
Recordar que el universo de discurso de error es desde -100 hasta 100:



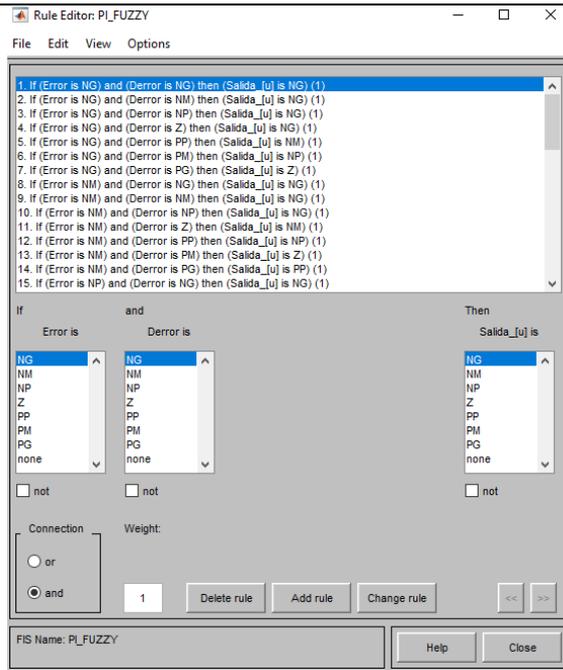
El universo de discurso de la derivada del error esta desde -4 hasta 4:



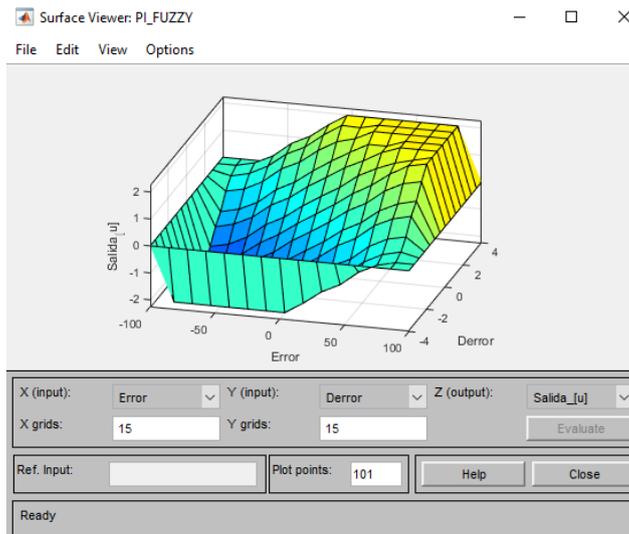
Y el universo de discurso de la salida es desde -3 hasta 3:



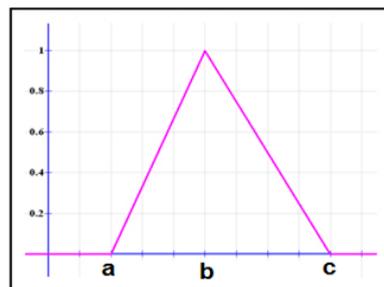
Según la tabla de FAM se transcribe las 49 reglas de la siguiente manera en el diseñador Fuzzy:



Según la salida del controlador se obtiene la siguiente gráfica de salida:



- Para la librería del sistema de control Difuso en Arduino, se tiene la siguiente función de pertenencia triangular representada en la siguiente figura:



Cuya ecuación viene dada por:

$$f(x, a, b, c) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{c-x}{c-b} & b \leq x \leq c \\ 0 & c \leq x \end{cases}$$

Entonces el grado de pertenencia del valor de entrada viene dado por:

$$f(x, a, b, c) = \max \left(\min \left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right)$$

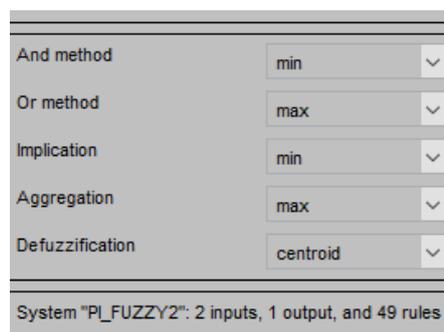
En la librería de Arduino Fuzzy.h esta ecuación se puede escribir en el código de la siguiente manera:

```

1 // Librería control difuso
2
3 //*****
4 #
5 CONTROL DE LÓGICA DIFUSA EN ARDUINO
6 by: Sergio Andrea Castaño Giraldo
7 https://controlautomaticodecacion.com
8
9 This code was used in the following research work. If you use this code or
10 modify this code, please cite the following research article in your work.
11 Remember to respect copyright:
12
13 CITE AS:
14 Giraldo, S. A. C., Gómez, D. S. W., & Blandón, J. N. S. (2022). Control y
15 monitoreo de temperatura para un horno de curado de prendas de algodón utilizando
16 lógica difusa y control de PI. REVISTA POLITECNICA, 9(17), 48 - 51.
17 Recuperado a partir de https://revistas.unpola.edu.co/index.php/pol/articulo/view/382
18
19 */
20 //*****
21
22 #undef max
23 #define max(a,b) ((a)>(b)?(a):(b))
24 #undef min
25 #define min(a,b) ((a)<(b)?(a):(b))
26
27 #ifndef __FUZZY_H
28 #define __FUZZY_H
29
30 //Creamos la firma de la función
31 float triangular(float x,float a,float b,float c);
32 float myfuzzy(float a,float de,float me,float mde,float ma);
33
34
35 #endif

```

- Las reglas serán escritas en el archivo fuzzy.cpp donde se declara la función de pertenencia triangular, las variables Fuzzy, las reglas del sistema usando el min que serán los consecuentes y la consecuencia por el método de agregación y la salida del control usando el método de cálculo de centro de gravedad, tal cual el diseñador de Fuzzy en Matlab lo realiza:



Y escrito en código en Arduino:

```
Practica-10 $ fuzzy.cpp $ fuzzy.h tclab_lib.cpp tclab_lib.h
19
20 #include "fuzzy.h"
21
22 float triangular(float x,float a,float b,float c)
23 {
24     float y;
25     y=max(min((x-a)/(b-a),(c-x)/(c-b)),0);
26     return(y);
27 }
28
29 float myfuzzy(float e, float de, float me, float mde, float mu)
30 {
31     // VARIABLES DE MYFUZZY
32     float eNG, eNM, eNP, eZ, ePP, ePM, ePG; //Error
33     float deNG, deNM, deNP, deZ, dePP, dePM, dePG; // Derivada del Error
34     float uNG=0, uNM=0, uNP=0, uZ=0, uPP=0, uPM=0, uPG=0; //Salida Parcial
35     float uFNG, uFNM, uFNP, uFZ, uFPP, uFPM, uFPG; //Salida Final
36     float Universe[3]; // Universo del discurso desde -1 hasta 1 con incrementos de 0.1
37     float r[49]; //Reglas
38     float maxU=0; //Maximos de todas las salidas
39     float cg; //Centro de Gravedad
40     float sumfx=0, sumUX=0;
41     int i; //Contadores
42     float k;
43
44     Universe[0]=-1*mu;
45     Universe[1]=(mu*2)/200;
46     Universe[2]=1*mu;
47
48
49     eNG=triangular(e,-1*me,-1*0.75*me,-1*0.5*me);
50     eNM=triangular(e,-1*0.75*me,-1*0.5*me,-1*0.25*me);
51     eNP=triangular(e,-1*0.5*me,-1*0.25*me,0);
52     eZ=triangular(e,-1*0.25*me,0,0.25*me);
53     ePP=triangular(e,0,0.25*me,0.5*me);
54     ePM=triangular(e,0.25*me,0.5*me,0.75*me);
55     ePG=triangular(e,0.5*me,0.75*me,me);
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586

```

```

134 //=====
135           ETAPA DE AGREGACION POR EL MAXIMO
136 //=====*/
137
138 //Reglas Negativo Grande
139 for (i=0;i<4;i++)
140 {
141     uNG = max(uNG,r[i]);
142 }
143 for (i=4;i<7;i++)
144 {
145     uNG = max(uNG,r[i+3]);
146 }
147 uNG = max(uNG,r[14]);
148 uNG = max(uNG,r[15]);
149 uNG = max(uNG,r[21]);
150
151 //Reglas Negativo Mediano
152 uNM = max(uNM,r[4]);
153 uNM = max(uNM,r[10]);
154 uNM = max(uNM,r[16]);
155 uNM = max(uNM,r[22]);
156 uNM = max(uNM,r[28]);
157
158 //Reglas Negativo Pequeno
159 uNP = max(uNP,r[5]);
160 uNP = max(uNP,r[11]);
161 uNP = max(uNP,r[17]);
162 uNP = max(uNP,r[23]);
163 uNP = max(uNP,r[29]);
164 uNP = max(uNP,r[35]);
165
166 //Reglas Zero
167 uZ = max(uZ,r[6]);
168 uZ = max(uZ,r[12]);
169 uZ = max(uZ,r[18]);
170 uZ = max(uZ,r[24]);
171 uZ = max(uZ,r[30]);
172 uZ = max(uZ,r[36]);

```

Salida del control Difuso:

```

203 //SALIDA DEL DIFUSO
204 k=Universe[0];
205 while (k<=Universe[2])
206 {
207     uFNG=triangular(k,-1*mu,-1*0.75*mu,-1*0.5*mu);
208     uFNM=triangular(k,-1*0.75*mu,-1*0.5*mu,-1*0.25*mu);
209     uFNP=triangular(k,-1*0.5*mu,-1*0.25*mu,0);
210     uFZ=triangular(k,-1*0.25*mu,0,0.25*mu);
211     uFPF=triangular(k,0,0.25*mu,0.5*mu);
212     uFPM=triangular(k,0.25*mu,0.5*mu,0.75*mu);
213     uFPG=triangular(k,0.5*mu,0.75*mu,mu);
214
215 //Saturo la salida con los puntos maximos de la agregacion
216 if (uFNG>=uNG)
217     uNG=uFNG;
218 if (uFNM>=uNM)
219     uNM=uFNM;
220 if (uFNP>=uNP)
221     uNP=uFNP;
222 if (uFZ>=uZ)
223     uZ=uFZ;
224 if (uFPF>=uFP)
225     uFP=uFPF;
226 if (uFPM>=uPM)
227     uPM=uFPM;
228 if (uFPG>=uPG)
229     uPG=uFPG;
230
231
232 //Agrupo los valores maximos hallados de las 7 funciones de membresia
233 maxU = max(maxU,uNG);
234 maxU = max(maxU,uNM);
235 maxU = max(maxU,uNP);
236 maxU = max(maxU,uZ);
237 maxU = max(maxU,uFP);
238 maxU = max(maxU,uPM);
239 maxU = max(maxU,uPG);
240
241

```

```

-      .....
:0
:1      //Guardo en el vector FX
:2      sumUX = sumUX + maxU*k;
:3      sumfx = sumfx + maxU;
:4
:5      k=k+Uuniverse[1];
:6      maxU = 0;
:7      }
:8
:9      //CALCULO DEL CENTRO DE GRAVEDAD
:10     cg=sumUX/sumfx;
:11
:12     return(cg);
:13
:14 }

```

- En el programa principal, el Control Fuzzy Logic a utilizarse es un sistema de tipo Proporcional, Integral y Derivativo (PID), la parte proporcional es el producto entre la señal de error y la constante proporcional, esto para lograr que el error en estado estacionario se aproxime a cero, la acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error obteniéndolo de la diferencia de la señal de error actual con la señal pasada y dividida para el tiempo de muestreo, y la acción integral es la suma de la señal de salida y la acción de control pasada, se puede multiplicar una ganancia para obtener una respuesta adecuada en la práctica, todo esto estará calculándose en el sampleTime, la función de interrupción que obtendrá los valores de control pasados, usando la librería tclab_lib.cpp, que actualizará la señal de error pasado y obtendrá el vector de salida de nuestro controlador.

```

Practica-10$ fuzzy.cpp fuzzy.h tclab_lib.cpp tclab_lib.h
88 void SampleTime(void)
89 {
90     //Actualiza los vectores u y e
91     update_past(u, KU);
92     update_past(e, KE);
93
94     //Calcula el Error
95     e[kE] = Setpoint - Input;
96
97     // Satura el error para estar dentro del universo del discurso
98     // para evitar valores fuera de universo de discurso
99     if (e[kE] < -1 * discurso_e + 0.1)
100         e[kE] = -1 * discurso_e + 0.1;
101     if (e[kE] > discurso_e - 0.1)
102         e[kE] = discurso_e - 0.1;
103
104     // Calcula la derivada del error discreta
105     der = (e[kE] - e[kE - 1]) / Te;
106
107     // Satura la derivada del error para estar dentro del universo del discurso
108     if (der < -1 * discurso_de + 0.01)
109         der = -1 * discurso_de + 0.01;
110     if (der > discurso_de - 0.01)
111         der = discurso_de - 0.01;
112
113     //Calculo del control Fuzzy
114     lu = myfuzzy(e[kE], der, discurso_e, discurso_de, discurso_u);
115
116     //Integral
117     //salida de control * constante de integral + accion de control pasada
118     u[kU] = lu * 20 + u[kU - 1];
119
120     // Anti - Windup
121     if (u[kU] >= 100.0)
122         u[kU] = 100.0;
123     if (u[kU] <= 0.0)
124         u[kU] = 0.0;
125
126     Output = u[kU];

```

```

126 Output = u[kU];
127 //Aplica la acción de control en el PWM
128 if (HMI_Start == 1) {
129   analogWrite(motor1, map(Output, 0, 100, 0, 255)); //max= 100, Min=0
130 }else{
131   analogWrite(motor1,0);
132 }
133
134
135 }
136

```

Calculo de error pasado y salida de control PID Fuzzy en librería tclab_lib.cpp

```

Practica-10$ fuzzy.cpp fuzzy.h tclab_lib.cpp tclab_lib.h
2 #include "tclab_lib.h"
3
4
5
6 /*===== FUNCION DE ACTUALIZACIÓN =====*/
7 /*=====*/
8 // Esta función desplaza todo el vector una unidad hacia atras para actualizar
9 // los valores pasados en la ecuaciones causales de los controladores
10 // float v: El vector que deseamos desplazar (Actualizar)
11 // int kT: Es la ultima posición del vector v
12 void update_past(float v[],int kT){
13   int i;
14   for(i=1;i<=kT;i++){
15     v[i-1]=v[i];
16   }
17 }
18
19 /*===== FUNCION DEL CONTROL PID =====*/
20 /*=====*/
21
22 float PID_Controller(float u[], float e[3], float q0, float q1, float q2)
23 {
24   float lu;
25   // Controle PID
26   // e[2] = e(k)
27   // e[1] = e(k-1)
28   // e[0] = e(k-2)
29   // u[0] = u(k-1)
30   lu = u[0] + q0*e[2] + q1*e[1] + q2*e[0]; //Ley del controlador PID discreto
31
32   // Anti - Windup
33   if (lu >= 100.0)
34     lu = 100.0;
35
36   if (lu <= 0.0)
37     lu = 0.0;
38
39   return (lu);
40 }

```

- El programa principal tal como la práctica 7, lee la señal de entrada en este caso el sensor ultrasónico para medir el nivel, adquiere los datos de la interfaz HMI para dar inicio y parada del control y presenta los datos enviándolos al mensaje LCD de Arduino y por medio de Modbus hacia PLC Unitronics:

```

Practica-10 Arduino 1.8.19 (Windows Store 1.8.57.0)
Archivo Editar Programa Herramientas Ayuda
Practica-10$ fuzzy.cpp fuzzy.h tclab_lib.cpp tclab_lib.h
176 void loop() {
177   unsigned int uS = sonar.ping();
178   distancia = uS / US_ROUNDTRIP_CM;
179
180   Sell = digitalRead(Selector1);
181   Sell2 = digitalRead(Selector2);
182   ParoEmergencia = digitalRead(PinEmergencia);
183   Potencimetro4 = analogRead(A4);
184
185   Setpoint = map(Potencimetro4, 0, 1020, 2, 20);
186   Input = distancia;
187
188   int Frec = map(Output, 0, 100, 0, 60);
189
190   // Adquisicion de dato de paro de emergencia desde pantalla
191   //HMI de Modulo de Unitronics en direccion Modbus M22
192   paroHMI = Mb.MbData[22];
193   paroUnitronics = Mb.MbData[23];
194
195   if (ParoEmergencia == HIGH) {
196     Mb.MbData[1] = 0;
197   } else {
198     Mb.MbData[1] = 1;
199   }
200   //Condicion de parada de motores en sistema
201   if (ParoEmergencia == HIGH and paroHMI == 0 and Sell == 0 and paroUnitronics == 0 ) {
202     //Envio de dato de parada de motor desde paro de Arduino a Unitronics
203     HMI_Start = 1;
204     digitalWrite(Piloto4, LOW);
205     digitalWrite(Piloto1, HIGH);
206   } else {
207
208     HMI_Start = 0;
209     analogWrite(motor1, 0);
210     digitalWrite(Piloto4, HIGH);
211     digitalWrite(Piloto1, LOW);
212
213   }

```

- Condiciones de Start – Stop – Paro de emergencia:

```

Practica-10$ fuzzy.cpp fuzzy.h tciab_lib.cpp tciab_lib.h
213 }
214 //Condicion de descarga de tanque, paro de emergencia y control de descarga
215
216 if (distancia - 1 > 3) {
217
218 //Activa Perturbacion
219 if ( Sel1 == 1 or distancia - 1 > Setpoint and paroHMI == 0 and Sel2 == 0 and paroUnitronics == 0 ) {
220 analogWrite(motor2, 70 );
221 } else {
222 analogWrite(motor2, 0);
223 }
224 } else {
225 analogWrite(motor2, 0);
226 }
227
228 if (distancia - 1 > 3) {
229 if ( Sel2 == 1 and paroHMI == 0) {
230 analogWrite(motor2, 15);
231 }
232 }
233

```

- Mensajes en LCD, envío de datos a Unitronics con Modbus y limpieza de mensaje en LCD:

```

Practica-10 Arduino 1.8.19 (Windows Store 1.8.37.0)
Archivo Editar Programa Herramientas Ayuda
Practica-10$ fuzzy.cpp fuzzy.h tciab_lib.cpp tciab_lib.h
233
234 // Mensajes en pantalla LCD
235 lcd.setCursor(1, 0);
236 lcd.print("Modulo 3 - Arduino");
237 lcd.setCursor(9, 1);
238 lcd.print("SP:");
239 lcd.setCursor(13, 1);
240 lcd.print(Setpoint);
241
242 lcd.setCursor(9, 3);
243 lcd.print("OUT:");
244 lcd.setCursor(13, 3);
245 lcd.print(map(Output, 0, 100, 0, 255));
246
247 lcd.setCursor(9, 2);
248 lcd.print("Dist:");
249 lcd.setCursor(14, 2);
250 lcd.print(distancia - 1);
251
252
253 // resta de nivel de tanque con distancia sensada
254 // para ajuste y visualizacion de tanque 2
255 int dist = 20 - distancia;
256
257 //Envio de datos a Unitronics
258 Mb.MbData[0] = distancia - 1;
259 Mb.MbData[2] = 0;
260 Mb.MbData[3] = 0;
261 Mb.MbData[4] = 0;
262 Mb.MbData[5] = Setpoint;
263 Mb.MbData[6] = dist;
264 Mb.MbData[7] = Frec;
265
266 if (millis() - TiempoAnterior1 > 400 ) {
267 TiempoAnterior1 = millis();
268 lcd.clear();
269 }
270
271 Mb.MbRun();

```

- **Pantalla en Unitronics**

La interfaz a usar es la misma de la Práctica número 7, se debe cargar el mismo programa en el PLC.



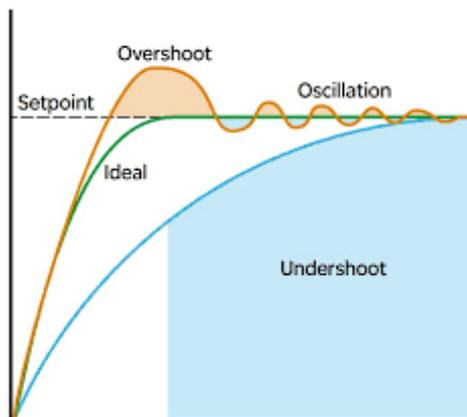
ACTIVIDADES POR DESARROLLAR:

1. Personalizar las reglas para poder tener una mejor respuesta del sistema
2. Cambiar el tiempo del Timer y observar cómo se comporta el sistema
3. En el código: $u[kU] = lu * 20 + u[kU - 1]$; cambiar el valor de 20 a gusto y observar cómo se comporta la salida del controlador.
4. Cambiar el tiempo de muestreo T_s .
5. Personalizar con datos reales del sistema los universos de discurso.

RESULTADOS OBTENIDOS:

Para esta práctica se pudo observar que la respuesta del sistema es mucho más eficaz de forma empírica, pero se debe ajustar los valores del tiempo de muestreo y el tiempo del Timer para mejorar la salida analógica del motor.

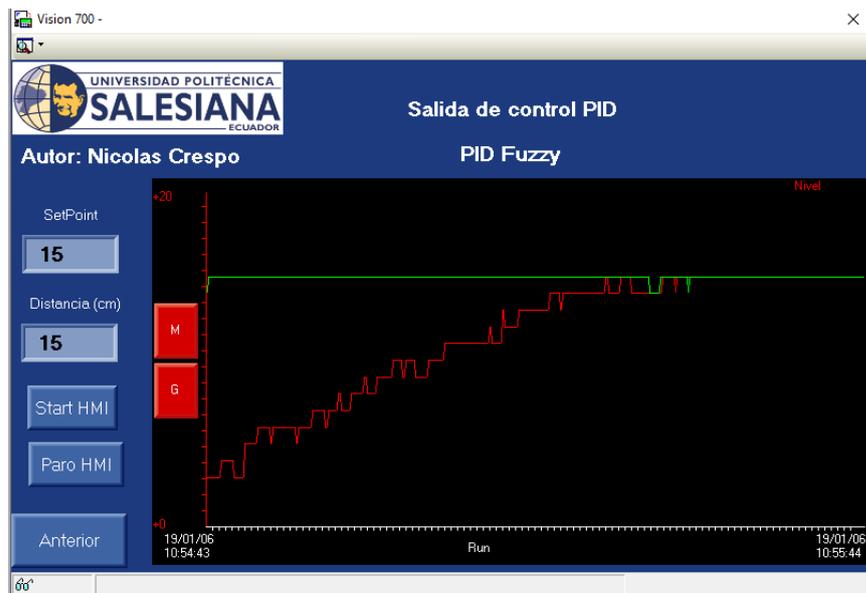
Con un tiempo de muestreo de 3 segundos, y un tiempo de timer también de 3 segundos se logró tener una respuesta que se acerca al obtenido con el controlador PID de la practica 7, aunque este sin tener un overshoot, sin tener Oscilaciones y más acercándose a un control Ideal.



Grafica obtenida de sistema PID de práctica 7:



Grafica obtenida de Sistema PID Fuzzy de esta práctica:



CONCLUSIONES:

Se diseñó el controlador Fuzzy mediante las reglas de correspondencia en Matlab programado en Arduino, se diseñó una interfaz interactiva con HMI Unitronics mostrando los parámetros relevantes, además se pudo contrastar las características de los controladores tipo PID con los controladores Fuzzy.

RECOMENDACIONES:

- Trabajar en un rango menor de distancia cuando se use algún líquido ya que puede ocurrir un rebosamiento y dañar el motor.
- Antes de encender el motor trifásico, asegurarse que ocurra la estabilización en output, evitando caídas de frecuencias bruscas.
- No trabajar mucho con el motor hasta encontrar las reglas de correspondencia para que no exista una perturbación en la señal de salida
- Verificar mediante los indicadores Led de las antenas que existe una conexión entre estas.
- Verificar que la dirección IP de la PC este en el mismo rango de direcciones configuradas en las antenas previamente.

Docente/Técnico Docente: _____

Firma: _____

5. Análisis y resultados

5.1 Resultados obtenidos

Para el desarrollo de este proyecto de titulación se obtuvo como resultado la implementación de dos módulos de comunicación inalámbrica usando el protocolo de comunicación Industrial Modbus TCP/IP, el cual permitió comunicarse e interactuar con procesos en los 2 Módulos más uno de los tableros que contiene un PLC Siemens S7-1200 que se encuentra en el laboratorio de Automatización industrial ubicado en el edificio F de la Universidad Politécnica Salesiana sede Guayaquil, como resumen se obtuvo:

- Diseño de tablero de control Arduino.

Con el tablero de control Arduino se obtuvo como resultado la comunicación Modbus TCP, el control de la planta didáctica de sensores, banda transportadora y llenado de tanques, además de control remoto de actuadores que se encuentran en los módulos de Siemens y Unitronics, también de implementar sistemas de control Fuzzy y PID con éxito en cada una de las pruebas.



Figura 64: Tablero de control Arduino.

Fuente: (El autor)

- Diseño de sistema de banda transportadora y llenado de tanques.
Como resultado se obtuvo el control de los procesos de prueba de los sensores y actuadores, control de llenado de tanques, control de banda transportadora, y acción con éxito en cada una de las prácticas donde se necesita su uso.

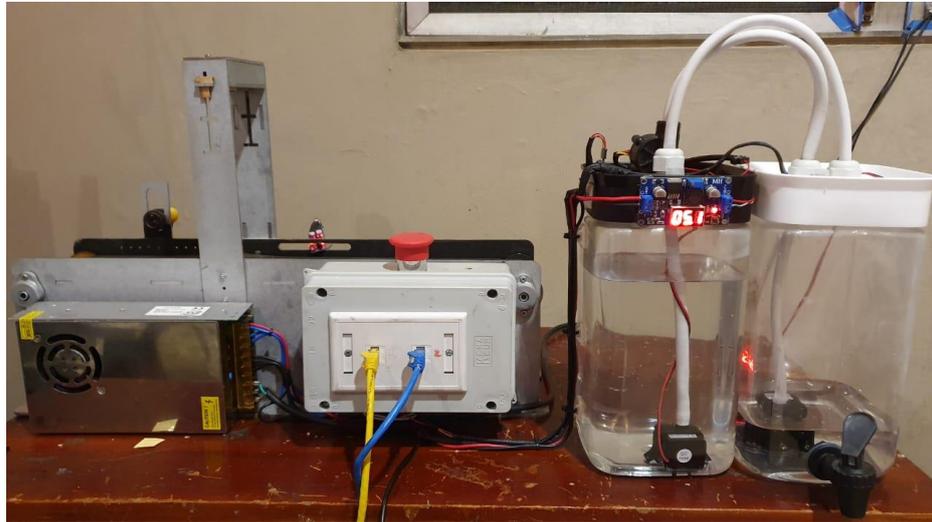


Figura 65: Planta didáctica, llenado de tanques y banda transportadora.

Fuente: (El autor)

- Diseño de tablero de control PLC Unitronics.
Con el tablero de control Unitronics se obtuvo como resultado la comunicación Modbus TCP usando este equipo como esclavo y maestro, el PLC Unitronics recibió y envió cada señal de los equipos conectados a él cuando la conexión se estableció como equipo maestro haciendo posible que los demás equipos (PC, Arduino, PLC Siemens), interactúen entre ellos y realizar controles remotos de cada uno de los actuadores conectados a los tableros. Cuando se utilizó el PLC Unitronics como equipo esclavo, este se conectó a un equipo maestro, que en este caso fue solo la PC y de manera básica y prueba se aprendió su uso conectándolo de este modo.
Así mismo se presentó el encendido, apagado y variación de velocidad del motor trifásico de manera local y remota (control remoto desde PC, Arduino o PLC Siemens).

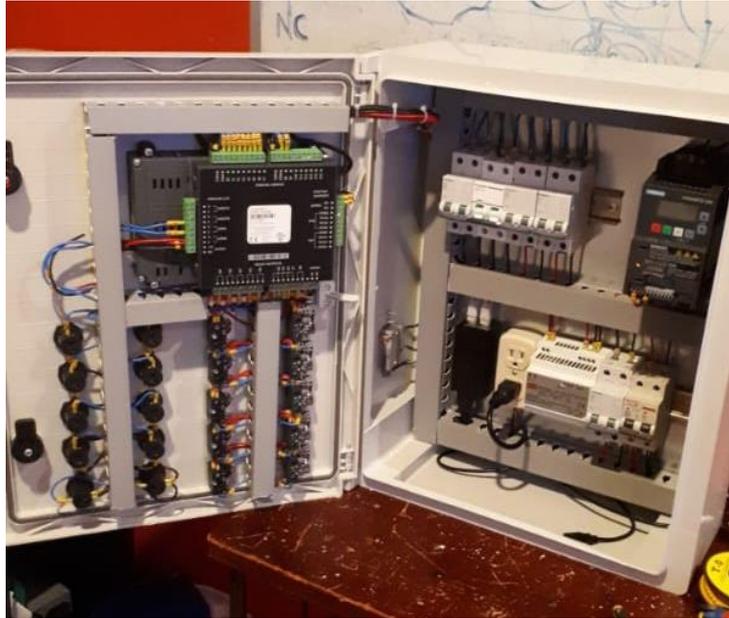


Figura 66: Tablero de control PLC Unitronics.

Fuente: (El autor)

- Diseño de comunicación analógica entre PLC Unitronics y variación de frecuencia.

Como resultado se pudo controlar el motor trifásico desde tablero de control de PLC Unitronics, también desde cualquiera de las estaciones implementadas, la comunicación se estableció de manera analógica desde una salida desde el PLC Unitronics hasta la entrada analógica del variador de frecuencia, también de la conexión digital para encender y apagar el motor.

El motor se uso en las prácticas para control PID, control Fuzzy, simulación de motor de banda transportadoras y pruebas básicas del mismo.



Figura 67: Motor trifásico en funcionamiento.

Fuente: (El autor)

- Pruebas de comunicación inalámbrica entre estaciones.
Las antenas realizaron enlaces exitosos a la antena de punto de acceso sin pérdida de señal en cuanto se establecía la comunicación, y se pudo proceder a comunicar los equipos.
- Diseño de red inalámbrica entre antenas Ubiquiti ubicadas en cada uno de los tableros.
Las antenas conectadas a los tableros con direcciones IP estáticas se configuraron en modo estación, cada estación se conecta a la antena acceso que fue configurada en modo punto de acceso y desde el punto de acceso se puede conectar de forma inalámbrica por wifi o por cable a un computador, donde se puede acceder a cada uno de los dispositivos conectados automáticamente a esta red, como resultado al configurar cada dispositivo con una dirección IP se pudo hacer ping a cada uno y comprobar su conexión.
- Elaboración de prácticas estudiantiles.
Las primeras cinco prácticas fueron la comunicación Modbus TCP básica entre las estaciones, estas prácticas se diseñaron así para generar entendimiento, y los resultados obtenidos fueron exitosos ya que en los siguientes desarrollos se implementaron procesos intermedios y avanzados aplicando lazos de control. La práctica número 9 con tema: diseño de control de fábrica de botellas demuestra la comunicación exitosa del protocolo Modbus TCP ya que los equipos esclavos se conectan al equipo maestro Unitronics, permitiendo que: desde PC se controle velocidad remota de motor trifásico, encendido y apagado de banda transportadora, control de velocidad de bomba centrífuga que simula llenado de botellas, visualización de proceso (SCADA), visualización de temperatura desde PLC Siemens, visualización y control de datos desde tablero Arduino, entre otros, demostrando que si existen procesos en una planta industrial donde cada uno se divide en varios procesos en distintas áreas, estos se pueden comunicar haciendo una integración y como resultado se obtiene un único control desde una estación.



Figura 68 Prueba de equipos conectados a la red.

Fuente: (El autor)



Figura 69 Prueba de conexión de Antenas Ubiquiti.

Fuente: (El autor)

Durante las pruebas realizadas y las prácticas desarrolladas se pudo interactuar entre todos los dispositivos conectados a la red, se aseguró de que existan los retardos

mínimos entre la comunicación tanto en programación como en la parte física, realizando las conexiones con cables RJ45 en buen estado.

En el PLC Siemens se conectó a un terminal hembra RJ45 la antena Ubiquiti para no intervenir en la red que ya se encuentra implementada en el módulo de PLC, la conexión entre antenas es automática, es decir las antenas de los dispositivos controladores se conectarán a la antena de Acceso cuando esta esté en línea de vista.

Las prácticas están dimensionadas con niveles de dificultad para una mejor comprensión de la comunicación entre equipos controladores.

5.2 Análisis de resultados.

Se aseguro que todas las conexiones estén correctamente dimensionadas, desde el circuito de fuerza hasta las conexiones con las antenas, para llevar a cabo se realizó los siguientes pasos:

- Revisión de circuitos eléctricos y electrónicos en los tableros Unitronics y Arduino.
- Conexión de antenas Ubiquiti a dispositivos de control.
- Pruebas de conexión y funcionamiento de red de Antenas:

Estas pruebas se llevaron a cabo con los equipos conectados y en funcionamiento, para aislar posibles problemas eléctricos y electrónicos que puedan realizar interferencias en la comunicación inalámbrica, para probar la comunicación inalámbrica no debe haber obstáculos intermedios ya que las antenas necesitan de una línea de vista directa, la distancia máxima de prueba fue de > 20 metros.

- Prueba de comunicación entre PLC Siemens, PLC Unitronics, Arduino, y PC.
- Prueba de dispositivos sensores y actuadores conectados a la banda transportadora y tanques conectados a Arduino.
- Elaboración de prácticas estudiantiles:

Para un entrenamiento de control y automatización óptimo, los resultados obtenidos van a ser directamente proporcionales a la manipulación correcta de

los equipos, asegurándose que los datos entre todas las estaciones sean enviados y recibidos correctamente.

Las prácticas implementadas se realizaron en base a los conocimientos graduales que se puedan obtener de ellas, es decir que los desarrollos van desde un nivel de implementación fácil hasta un nivel intermedio – avanzado.

CONCLUSIONES

- Se demostró de manera práctica y física el uso de protocolo de comunicación Industrial MODBUS TCP/IP que es muy requerido y utilizado por las industrias para sus aplicaciones.
- Se demostró que MODBUS TCP/IP es muy eficiente a la hora de establecer una comunicación industrial con muy poco tiempo de establecimiento y de conexión que permite al usuario tener datos sin retardos en momento real de sus procesos.
- Se estableció conexiones y se creó procesos físicos y simulados probando así la red MODBUS TCP/IP interactuando con los medios físicos como sensores.
- Se logró comandar diferentes actuadores como luces pilotos, relés, motores.
- Se estableció una conexión con un variador de velocidad para probar y estudiar un tipo de arranque remoto desde cualquier estación construida en esta implementación.
- Se realizó prácticas implementando los tipos de controladores fuzzy y PID para un estudio más profundo interactuando con sensores y actuadores en los Módulos didácticos.

RECOMENDACIONES

- Se recomienda hacer una tabla de direccionamiento entre Maestros y Servidores que se configuraran en la red MODBUS y así evitar errores a la hora de realizar las prácticas.
- Se recomienda revisar las fichas técnicas de cada uno de los equipos para poder entender mejor el funcionamiento de cada uno de los dispositivos conectados en los módulos didácticos.
- Se recomienda tomar en consideración la distancia entre la conexión de la antena y el router ya que si sobrepasa la normativa regular de cableado podría presentarse lentitud y pérdida de señal al transmitir la información.
- Se recomienda realizar un manual de usuario detallada para una futura implementación y un mejor desempeño en el usuario final.
- Se recomienda que el espacio destinado para ser el cuarto de los equipos tenga las condiciones climatizadas necesarias para evitar un sobrecalentamiento de los equipos.

BIBLIOGRAFÍA

- Arduino. (2018). arduino cc. Obtenido de <https://www.arduino.cc/en/Guide/Introduction>
- Arduino. (2020). Arduino. Obtenido de <https://store.arduino.cc/usa/mega-2560-r3>
- Bricogeek. (2022). Obtenido de [https://tienda.bricogeek.com/sensores-temperatura/510-sensor-ds18b20-estanco.html#:~:text=Caracter%C3%ADsticas%20del%20sensor%20DS18B20%3A%2C%20a%2012%20bits%20\(configurable\)&text=Multiples%20sensores%20puede%20compartir%20el,C%20a%20%2B85%C2%B0C](https://tienda.bricogeek.com/sensores-temperatura/510-sensor-ds18b20-estanco.html#:~:text=Caracter%C3%ADsticas%20del%20sensor%20DS18B20%3A%2C%20a%2012%20bits%20(configurable)&text=Multiples%20sensores%20puede%20compartir%20el,C%20a%20%2B85%C2%B0C)
- Colsein. (2022). Obtenido de <http://www.colsein.com.co/producto/170/vision-v700-t20bj>
- Estrada, R. (2017). ¿Que es Arduino? su historia e importancia. Universidad de Guadalajara, Guadalajara. Recuperado el 3 de Diciembre de 2018, de hetpro: <https://hetpro-store.com/TUTORIALES/que-es-arduino/>
- Factory, G. (30 de Junio de 2019). Geek Factory. Obtenido de <https://www.geekfactory.mx/tutoriales/tutoriales-arduino/ds18b20-con-arduino-tutorial-de-sensor-de-temperatura-digital/>
- Geekfactory. (29 de Mayo de 2017). Geekfactory. Obtenido de <https://www.geekfactory.mx/tutoriales/tutoriales-arduino/lcd-16x2-por-i2c-con-arduino/>
- Gines, E. (19 de Julio de 2019). Obtenido de <https://aprendiendoarduino.wordpress.com/2018/04/14/sensores-arduino-3/>
- helloauto. (2022). Obtenido de <https://helloauto.com/glosario/potenciometro#:~:text=Dentro%20del%20sector%20de%20la,de%20al%20menos%20tres%20terminales.&text=Es%20decir%2C%20podemos%20obtener%20entre,la%20diferencia%20de%20potencial%20total.>
- Keyence. (2022). Obtenido de <https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/>
- Martinez, I. (18 de Marzo de 2015). joautomation. Obtenido de <http://joautomation.com/arduino/la-introduccion-arduino/>
- MeanWell. (s.f.). meanwell-we.com. Obtenido de <https://www.meanwell-web.com/content/files/pdfs/productPdfs/MW/Dr-60/DR-60-spec.pdf>
- Mechatronics, N. (2021). Naylamp Mechatronics. Obtenido de <https://naylampmechatronics.com/sensores-liquido/108-sensor-de-flujo-de-agua-12-yf-s201.html>
- Modbus. (24 de Octubre de 2006). Modbus Organization. Obtenido de http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf
- Motorex. (4 de Agosto de 2020). Obtenido de <https://www.motorex.com.pe/blog/motores-trifasicos-ventajas/#:~:text=Los%20motores%20trif%C3%A1sicos%20son%20m%C3%A1quinas,utilizada%20en%20muchas%20aplicaciones%20industriales.>

MQTT. (s.f.). MQTT org. Obtenido de <http://mqtt.org/faq>

National Instruments. (17 de Septiembre de 2019). Obtenido de NI: <https://www.ni.com/es-cr/innovations/white-papers/14/the-modbus-protocol-in-depth.html>

ni.com. (5 de Febrero de 2021). Obtenido de <https://www.ni.com/es-cr/innovations/white-papers/14/application-development-with-modbus.html>

Portilla, L. (2015). Obtenido de https://www.unipamplona.edu.co/unipamplona/portallIG/home_74/recursos/visual-basic-para-excel/17052017/u5_fotoresistencia.jsp#:~:text=Una%20fotorresistencia%20es%20un%20componente,en%20ingl%C3%A9s%20light%2Ddependent%20resistor.

prometec. (2022). Obtenido de <https://www.prometec.net/siguelineas-ir/>

S&P. (9 de Diciembre de 2019). Obtenido de <https://www.solerpalau.com/es-es/blog/motor-trifasico/>

Sevillano, F. (2011). amelero.com. Obtenido de https://www.amelero.com/app/download/5858319664/Variadores_de_frecuencia_TIDA_13-14.pdf?t=1390409720

Sicma21. (Abril de 2021). Obtenido de <https://www.sicma21.com/que-son-las-redes-de-comunicacion-industrial/>

Siemens. (2014). S7 Controlador programable S7-1200 . Siemens.

Unitronics. (s.f.). Unitronicsplc. Obtenido de <https://unitronicsplc.com/>

VisiLogic. (s.f.). Elmark. Obtenido de http://support.elmark.com.pl/unitronics/PDF/VisiLogic_Software_Manual-Ladder.pdf

Web Robotica. (s.f.). Web Robotica. Obtenido de <https://www.web-robotica.com/arduino/como-funciona-el-modulo-arduino-ethernet-shield>

Anexos

Anexo 1: Timer WithOut Delays Crescer.h

Esta librería permite realizar un timer de retardo a un proceso, para evitar detención de procesos en Arduino con la función delay()

```
#include "Arduino.h"
#include "Crescer.h"

// Se crea una instancia
Tempora::Tempora(int indice)
{
    _indice = indice;
}
Tempora::Tempora(){}

//Este aqui sirve para definir el Setpoint
void Tempora::defiSP(unsigned long Setpoint)
{
    _Setpoint=Setpoint;
}

// esta función principal que cuando la entrada aguarda el tiempo de setpoint para
esperar la entrada y salida de este

// demuestra o tempo decorrido
boolean Tempora::Saida(boolean entrada)
{
    if (entrada==1)
    {
        if (setaIni==0)
        {
            tempoInicial=millis();
```

```

        setaIni=1;
    }

    if (millis() > tempoInicial + _Setpoint)
        {
            return 1;
        }
    else
        {
            CV = millis() - tempoInicial; //Tempo decorrido
            return 0;
        }
    }
else
{
    setaIni=0;
    CV=0;
    return 0;
}

}

int Tempora::Saida(int entrada)
{

    if (entrada==1)
    {
        if (setaIni==0)

```

```

{
    tempoInicial=millis();
    setaIni=1;
}

if (millis() > tempoInicial + _Setpoint)
    {
        return 1;
    }
else
    {
        CV = millis() - tempoInicial; //Tempo decorrido
        return 0;
    }
}
else
{
    setaIni=0;
    CV=0;
    return 0;
}
}

```

Anexo 2: Credenciales a antenas:

Antena Unitronics:

Usuario: ubnt

Contraseña: unitronics

Antena Arduino

Usuario: ubnt

Contraseña: arduino

Antena Siemens:

Usuario: ubnt

Contraseña: siemens

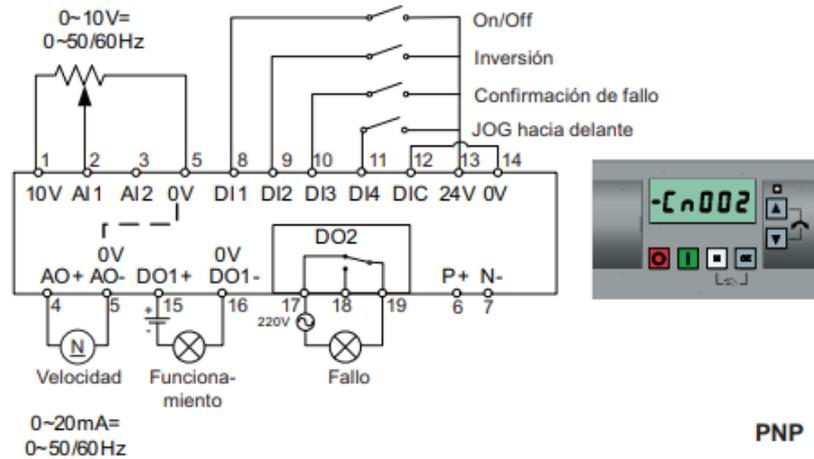
Antena Acceso

Usuario: ubnt

Contraseña: acceso

Anexo 3: Configuración de variador de frecuencia

Circuito:



Configuración de macros de conexión de variador de frecuencia

Parámetro	Descripción	Ajustes predeterminados de fábrica	Ajustes predeterminados de Cn002	Observaciones
P0700[0]	Selección de la fuente de señales de mando	1	2	Borne como fuente de señales de mando
P1000[0]	Selección de frecuencia	1	2	Análogica como consigna de velocidad
P0701[0]	Función de la entrada digital 1	0	1	ON/OFF
P0702[0]	Función de la entrada digital 2	0	12	Inversión
P0703[0]	Función de la entrada digital 3	9	9	Confirmación de fallo
P0704[0]	Función de la entrada digital 4	15	10	JOG hacia delante
P0771[0]	CI: Salida analógica	21	21	Frecuencia real
P0731[0]	BI: Función de la salida digital 1	52.3	52.2	Convertidor en funcionamiento
P0732[0]	BI: Función de la salida digital 2	52.7	52.3	Fallo del convertidor activo