ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN SISTEMA DE EXPANSIÓN DE ENTRADAS Y SALIDAS DIGITALES Y ANALÓGICAS PARA EL PLC S71200

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN ELECTROMECÁNICA

JEFFERSON STALIN YAR VÉLEZ
jefferson.yar@epn.edu.ec

MAYRA ELIZABETH ACURIO TIPANQUIZA
mayra.acurio@epn.edu.ec

DIRECTOR: ING. PABLO ANDRÉS PROAÑO CHAMORO, MSC.
pablo. proaño@epn.edu.ec

CODIRECTOR: ING. CARLOS ORLANDO ROMO HERRERA, MSC.
carlos.romo@epn.edu.ec

Quito, enero 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Yar Vélez Jefferson Stalin y Acurio Tipanquiza Mayra Elizabeth como requerimiento parcial a la obtención del título de TECNÓLOGO EN ELECTROMECÁNICA, bajo nuestra supervisión:

Ing. Pablo Andrés Proaño Chamorro, Msc.	**Ing. Carlos Orlando Romo Herrera, Msc.**
DIRECTOR DEL PROYECTO	CODIRECTOR DEL PROYECTO

DECLARACIÓN

Nosotros Jefferson Stalin Yar Vélez con CI: 172311993-7 y Mayra Elizabeth Acurio Tipanquiza con CI: 172047409-5 declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, somos titulares de la obra en mención y otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entregamos toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Jefferson Stalin Yar Vélez

Mayra Elizabeth Acurio Tipanquiza

DEDICATORIA

Este proyecto va dedicado a toda mi familia en general, sobre todo a mi padre Luis y a mi madre Patricia que siempre confiaron en mi en todo este proceso para mi formación profesional ya que sin ellos este camino hubiese sido difícil.

A mi hermana Jennifer Yar por su paciencia y por su apoyo incondicional en todos los aspectos de mi vida, ya que ella es una de las personas más importantes que Dios me pudo dar.

Y una mención especial a mi primo hermano Miguel Ángel Vélez quien ha sido mi motivación y mi inspiración para superarme en la vida, ya que gracias a toda su confianza depositada en mí y sus consejos me ayudaron a crecer día tras día en todo este ciclo como estudiante universitario y en mi vida personal.

Jefferson S. Yar Vélez

AGRADECIMIENTO

Le agradezco primeramente a Dios por darme la sabiduría, paciencia y las fuerzas necesarias para culminar este ciclo que fue el más importantes y complicados de mi vida.

En segundo lugar, a mi madre Patricia Vélez, por todo su amor incondicional y sincero que solo ella me pudo brindar, por enseñarme que todo en la vida se obtiene con esfuerzo y sacrificio, a ser persistente y jamás rendirme pese a las adversidades que se puedan presentar en la vida.

A mi padre Luis Yar por apoyarme siempre en las alegrías y en las adversidades que se pudieron presentar a lo largo de todo este camino, por transmitirme su humildad y sencillez ya que es son los valores más importantes que debe tener una persona.

Al Ing. Pablo Proaño por haber confiado en nosotros desde un principio y por el apoyo brindado a lo largo de toda la ejecución de este proyecto, y de igual forma al Ing. Alan Cuenca por todos los conocimientos que me transmitió a lo largo de sus clases y por ser una gran persona dentro y fuera del aula.

Finalmente, la Escuela Politécnica Nacional y a todos los docentes que a lo largo de todo este camino supieron tener la paciencia y la predisposición de compartir sus conocimientos que serán de mucha utilidad a lo largo de mi vida profesional y a todos mis amigos que supieron estar a mi lado en las buenas y en las malas.

Jefferson S. Yar Vélez

AGRADECIMIENTO

Mi sincero agradecimiento

A Dios, por permitirme culminar esta etapa de vida y por guiarme en cada paso de mi carrera universitaria.

A mis padres, por enseñarme con el ejemplo, que la vida siempre es grata con la persona que se esfuerza.

A mis profesores, por compartir su conocimiento y experiencia y que con su vocación incentivaron en mí la necesidad de aprender todos los días de la vida.

Al Ing. Pablo Proaño, por orientarnos en el desarrollo de esta tesis enfocándonos en realizar un trabajo de calidad, práctico y útil.

Y a cada una de las personas que, con su apoyo, con su tiempo, con su motivación hicieron posible completar mi educación universitaria con éxito.

Mayra E. Acurio Tipanquiza

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el presente trabajo se detalla el proceso para la implementación de un módulo de expansión con el propósito de aumentar el número de señales digitales y analógicas que un controlador lógico programable (PLC) puede manejar, fue construido en base a una plataforma de hardware libre como es Arduino.

De igual forma a través de los programas Proteus y Ares se implementó un circuito para el procesamiento de las señales discretas o analógicas con la finalidad de proteger los pines del microcontrolador, también se implementó un circuito de control de voltaje que comanda las salidas analógicas. Para definir la forma y el tamaño de las piezas del módulo se hizo el modelado de la estructura en el programa SolidWorks.

Se desarrolló un algoritmo en el programa TIA PORTAL V15 para comunicar un controlador lógico programable Siemens S7-1200 tipo cliente servidor vía modbus TCP/IP, de igual manera se hizo un programa que fue cargado a un Arduino mega 2560 para el procesamiento de señales y establecer comunicación con el autómata para el intercambio de información entre ambos dispositivos.

Finalmente, se grabaron dos videos explicativos, en el primero se hace referencia a las generalidades del módulo de expansión como también los pasos a seguir para la programación tanto del autómata como del microcontrolador. En el segundo video se explica el mantenimiento que se sugiere para el módulo de expansión.

PALABRAS CLAVE: PLC, Modbus, Arduino, Expansión, Control, TIA PORTAL

ABSTRACT

In the present work the process for the implementation of an expansion module is detailed in order to increase the number of digital and analog signals that a programmable logic controller (PLC) can handle, it was built based on a free hardware platform such as its Arduino.

Similarly, through the Proteus and Ares programs, a circuit was implemented for the processing of discrete or analog signals in order to protect the microcontroller pins; a voltage control circuit that commands the analog outputs was also implemented. To define the shape and size of the module parts, the structure was modeled in the SolidWorks program.

An algorithm was developed in the TIA PORTAL V15 program to communicate a Siemens S7-1200 programmable logic controller type client server via Modbus TCP / IP, in the same way a program was made that was loaded to an Arduino mega 2560 for signal processing and establish communication with the automaton for the exchange of information between both devices.

Finally, two explanatory videos were recorded, the first one refers to the generalities of the expansion module as well as the steps to follow for programming both the automaton and the microcontroller. The second video explains the suggested maintenance for the expansion module.

KEYWORDS: *PLC, Modbus, Arduino, Expansion, Control, TIA PORTAL.*

1 INTRODUCCIÓN

Es común que en el ámbito industrial en ciertos tipos de procesos se maneja gran variedad de variables físicas y lógicas con el fin de tomar diferentes tipos de acciones de control dependiendo el propósito, para esto se emplea un controlador lógico programable (PLC) el cual por medio de un algoritmo de control procesa la información de entrada ya sea de pulsadores, interruptores o sensores y acciona diferentes tipos de salidas como contactores, luces indicadoras, motores, etc. [1]

Sin embargo, en procesos grandes este número de periféricos de entradas y salidas son numerosos provocando que las entradas y salidas que posee el controlador no sean suficientes, frecuentemente es necesario adquirir módulos de expansión que permiten aumentar la capacidad para conectar más dispositivos al proceso, pero una de sus desventajas es su elevado costo incluso en casos superando al mismo controlador.

Es por todo esto que se ve la necesidad de buscar alternativas más económicas que permitan al controlador lógico programable abarcar más señales de control manteniendo la confiabilidad de un módulo de fábrica, por lo tanto, se plantea construir un módulo de expansión en base a una plataforma electrónica de código abierto, que a través de comunicación Modbus TCP permita la interacción entre ambos dispositivos. [2]

En el siguiente trabajo de titulación: “IMPLEMENTACIÓN DE UN SISTEMA DE EXPANSIÓN DE ENTRADAS Y SALIDAS DIGITALES Y ANALÓGICAS PARA PLC S71200”, se basa en la comunicación de una tarjeta Arduino Mega mediante el protocolo de comunicación Modbus TCP/IP a través de un conmutador el cual permite conectar diferentes equipos a una misma red, esto con el afán de monitorear mayor número de variables físicas y compartir datos entre ambos dispositivos para comandar el accionamiento de actuadores o adquirir la información proporcionadas por sensores.

En el primer capítulo se analiza los requerimientos técnicos del módulo, posteriormente se implementa una tarjeta de adaptación y se define la estructura donde se montará el sistema. A continuación, se crea un algoritmo de comunicación para el autómata y el microcontrolador y finalmente, se elaboran las pruebas correspondientes a cada uno de las funciones del módulo.

El proyecto se desarrolla para el Laboratorio de Tecnología Industrial de la Escuela de Formación de Tecnólogos de la EPN, con el fin de ayudar a los estudiantes de la carrera

de electromecánica a reforzar los conocimientos con respecto a la intercomunicación de dispositivos y el uso de protocolos de comunicación.

1.1 Objetivo general

Implementar un sistema de expansión de entradas y salidas digitales y analógicas para PLC S71200

1.2 Objetivos específicos

- Realizar estudios de requerimientos técnicos y condición del sistema.
- Implementar el módulo de expansión de entradas y salidas.
- Implementar el sistema de comunicación TCP/IP entre el PLC y el módulo
- Realizar pruebas y el análisis de resultados
- Elaborar un manual de uso y mantenimiento del sistema.

1.3 Fundamentos

Protocolo de comunicación Modbus

De la gran variedad de protocolos de comunicación uno que más destaca es el modbus el cual permite conectarse en red a equipos industriales como microcontroladores, controladores lógicos programables (PLC), drivers, ordenadores y otros periféricos de entradas y salidas, sin importar el modelo o fabricante, siempre y cuando el dispositivo permita establecer este tipo de comunicación. Una de las variantes del protocolo es el modbus TCP/IP el cual fue implementado por “Schneider Automation”, donde su campo de aplicación es amplio para el control y supervisión de equipos en procesos industriales por su facilidad de conexión por medio de una red de área local (LAN). [3]

Trama de Modbus TCP/IP

La trama modbus TCP/IP está compuesta por un encapsulado donde destacan la unidad de procesamiento de datos (PDU) la cual permite manipular los datos asociados, su la longitud va a depender del tipo de función modbus y los datos en sí, como se muestra en la Figura 1.1.

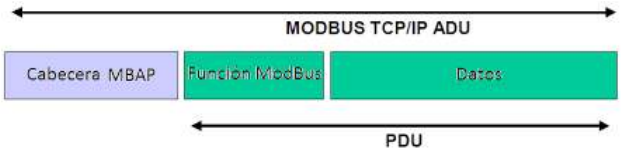


Figura 1.1 Trama Modbus TCP/IP. [3]

Modelo de Datos de Modbus

Este protocolo se define por cuatro rangos de dirección o bancos de datos, los cuales se muestran en la Tabla 1.1 dependiendo la configuración del dispositivo ya sea como maestro o esclavo, este solo podrá leer o escribir datos.

Tabla 1.1 Bloques de Modelo de Datos de Modbus. [4]

Bancos de datos	Tipo de dato	Acceso como maestro	Acceso como esclavo
Bobinas	Booleano	Escritura/Lectura	Escritura/Lectura
Entradas digitales	Booleano	Solo Lectura	Escritura/Lectura
Registros de entrada	Word	Solo Lectura	Escritura/Lectura
Registros de Retención	Word	Escritura/Lectura	Escritura/Lectura

Rangos de dirección de datos.

Los datos modbus son identificados con una serie de prefijos que varían dependiendo el tipo a estos se le asigna un rango esto con la finalidad de comprender los bloques de memoria modbus disponibles en un dispositivo especifico, en la Tabla 1.2 se muestran los prefijos usados para los diferentes tipos de datos.

Tabla 1.2 Prefijos de rangos de datos Modbus. [4]

Bloques de datos	Prefijo
Bobinas	0
Entradas discretas	1
Registros de entrada	3
Registros de retención	4

Por ejemplo, si a un elemento se le asigna la dirección 30101, el prefijo 3 hace referencia a un registro de entrada y el 100 a la dirección especificada, se aumenta un 1 porque que el autómata asigna las direcciones desde 30.001 hasta la 39.999, correspondientes a registros de entrada, ver Tabla 3.4.

2 METODOLOGÍA

2.1 Descripción de la metodología usada

Inicialmente se determinaron los requerimientos necesarios para el sistema, el número de entradas y salidas digitales se definió en función de la cantidad de bits que se pueden intercambiar entre cada dispositivo, la comunicación se logró mediante el uso del protocolo de comunicación modbus TCP/IP, ya que Arduino posee librerías que permiten establecer una comunicación a través de este medio.

Después, se adquirieron de los materiales para la implementación del módulo, como pilar fundamental esta un microcontrolador Arduino mega 2560 que al acoplarle una placa ethernet le va a permitir conectarse vía ethernet por medio de un conmutador y de esta manera crear una red con el controlador siemens S7-1200 y un ordenador, esto con la finalidad de intercambiar información entre en autómata y el microcontrolador.

Debido a que el sistema funciona a 24 (V_{DC}), se diseñaron circuitos encargados para reducir este voltaje a 5 (V_{DC}) para el control de las 8 entradas digitales (4 pulsadores y 4 interruptores) y no averiar lo pines digitales del Arduino. De igual manera se implementó un circuito encargado de acondicionar las salidas analógicas que funcionan en un rango de 0 a 10 (V_{DC}) manejando cargas hasta de 1.5 (A) y para las entradas analógicas se empleó un circuito de protección con seguidores de tensión para proteger los pines del microcontrolador.

Mediante el programa Ares se estableció las dimensiones de la placa de circuito impreso que se empleó para el procesamiento de las entradas digitales y para el acondicionamiento de las salidas analógicas, por medio del software SolidWorks se modeló una estructura que cumpla con el espacio necesario para garantizar la facilidad de montaje de los componentes.

Se creó bloques de datos, uno por cada función del módulo de expansión, es aquí donde se configura el número de variables a manejar y la dirección IP del otro equipo con el que se pretende establecer una comunicación y se configuró el bloque “MB_CLIENT”, uno por cada una de las variables que se va a manejar. Este bloque posee todos los parámetros necesarios para establecer una comunicación modbus con una topología cliente-servidor. Se hizo una Interfaz Humano Máquina (HMI) para verificar el estado del sistema.

Se desarrolló un programa en IDE Arduino para establecer el direccionamiento Modbus de las variables que se intercambió con el PLC, así como también la dirección IP única, para posteriormente configurar los parámetros que permiten intercambiar datos entre el microcontrolador y el PLC.

3 RESULTADOS Y DISCUSIÓN

3.1 Estudio de requerimientos del sistema

El módulo es alimentado a una tensión de 110/220 (V_{AC}) proporcionada por las tomas que poseen los tableros de control del Laboratorio de Tecnología Industrial, pero debido a que el microcontrolador encargado de establecer la comunicación y los demás dispositivos funcionan a tensiones menores de 8 (V_{DC}), fue necesario adquirir una fuente reguladora de voltaje con el fin de que los componentes funcionen correctamente y evitar averías. En la Figura 3.1 se muestra el esquema de conexión del módulo de expansión.

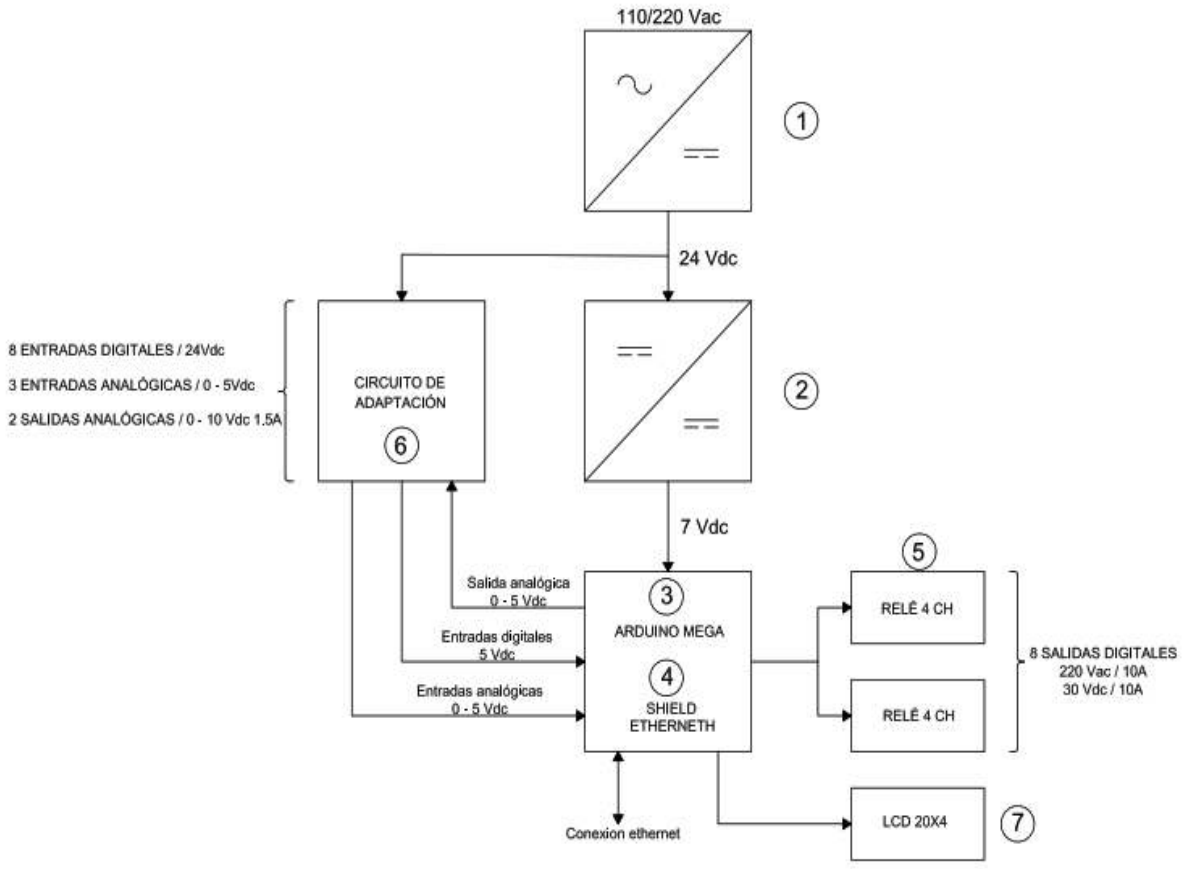


Figura 3.1 Esquema de conexión del módulo de expansión.

1. **Fuente rectificadora WXDC - 2412:** esta fuente rectifica la señal de corriente alterna de 110/220 (V_{AC}) a una en continua de 24 (V_{DC}) y es la encargada de alimentar todos los componentes del módulo, esta fuente es capaz de suministrar 6 (A) de corriente y se la puede apreciar en la Figura 3.2.

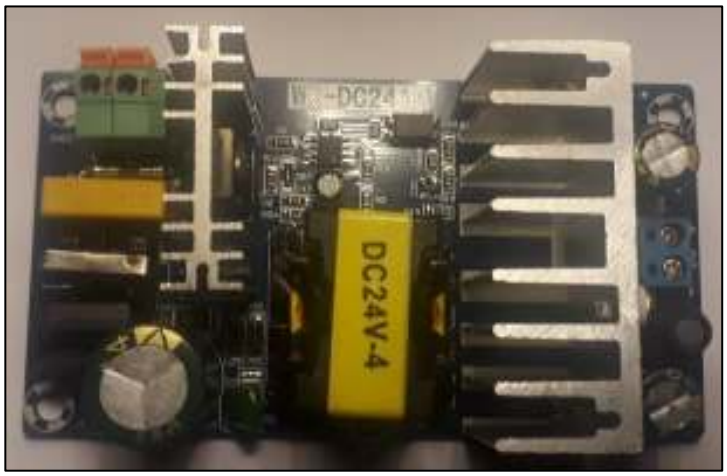


Figura 3.2 Fuente de 24 (V_{DC}).

2. **Fuente reductora de voltaje LM2596S DC-DC**: su principal característica es reducir el voltaje de salida con respecto al que tiene en su entrada entre un rango de 1.5 a 35 (V_{DC}) mediante un potenciómetro incorporado, esta fuente se encarga de alimentar el Arduino mega 2560, está calibrada en una tensión de 7 a 8 (V_{DC}) y suministra una corriente máxima de 3(A), puede ser observada en la Figura 3.3.

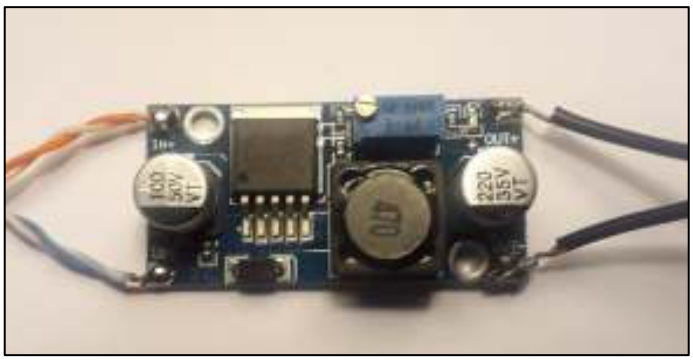


Figura 3.3 Fuente Step Down.

3. **Arduino mega 2560:** este es el componente principal del módulo de expansión ya que este se encarga de procesar la información de entradas y salidas, este modelo fue seleccionado por el número de pines que posee, ver Figura 3.4.

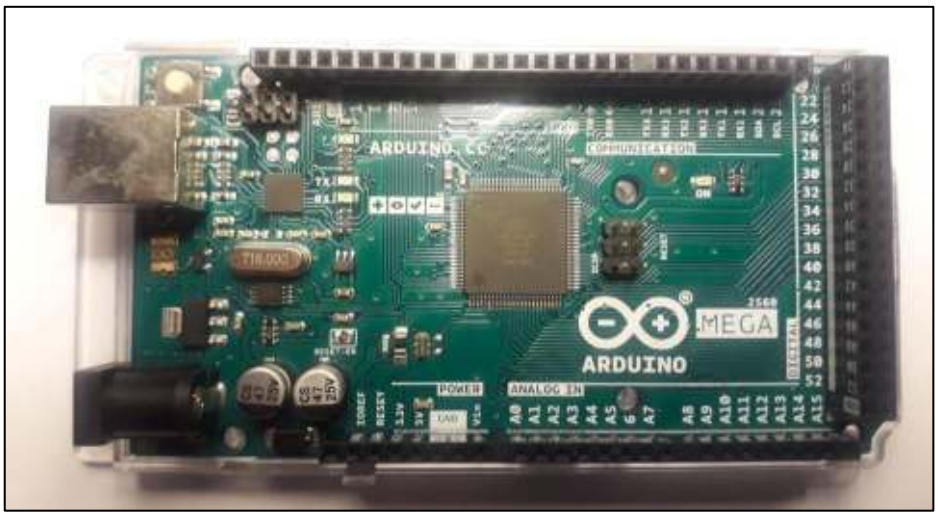


Figura 3.4 Arduino Mega 2560.

4. **Placa ethernet:** es una tarjeta que permite conectar a Arduino a una red ya sea ethernet o como en el caso del módulo una red Modbus. Esto por medio de un cable RJ45, ver Figura 3.5.

Figura 3.5 Placa ethernet.

5. **Módulo relé de cuatro canales:** es un circuito que cuenta con un relé y componentes de amplificación necesarios para permitirle a Arduino encender o apagar cargas de tensiones mayores, ya sea 110/220 (V_{AC}) como 30 (V_{DC}) y con un consumo no mayor a 10 (A), ver Figura 3.6. Por esta razón son empleados para controlar las salidas digitales del módulo.

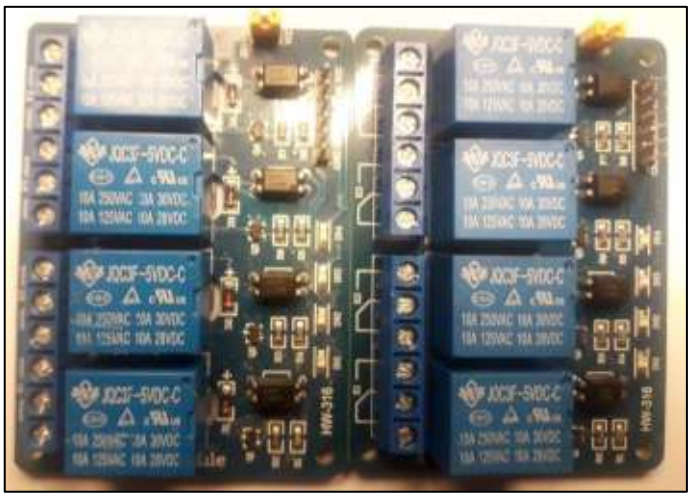


Figura 3.6 Módulo relé de 4 canales

6. **Circuito de adaptación:** es una tarjeta electrónica que cuenta con los circuitos de alimentación, aislamiento y adaptación de señales, ver Figura 3.7. Estas señales son procesadas por Arduino mega 2560 y enviadas hacia el PLC, funciona a 24 (V_{DC}) y se encuentran los circuitos para entradas y salidas analógicas, y de entradas digitales. Cuenta con un Arduino Nano encargado de hacer el control del voltaje de las salidas analógicas.

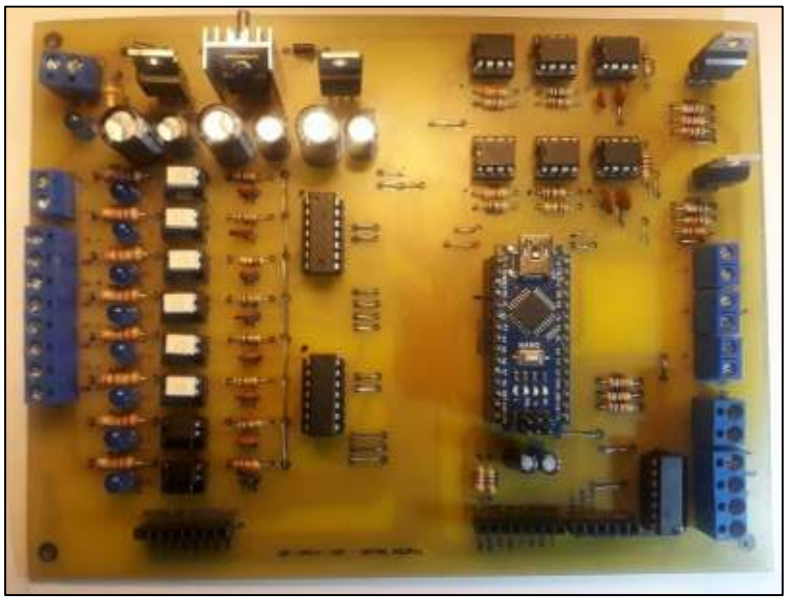


Figura 3.7 Circuito de adaptación

7. **Pantalla LCD 20X4:** permite visualizar los datos de entradas y salidas analógicas con el fin de corroborar que los valores enviados y recibidos son los correctos, ver Figura 3.8.

Figura 3.8 Pantalla LCD 20X4.

3.2 Implementación del módulo de expansión de entradas y salidas

Simulación de circuitos en el programa Proteus 8.9

Entradas digitales

Este circuito consta de opto acopladores para aislar eléctricamente cada una de las tensiones que se involucran en el proceso como se indica en la Figura 3.9, el pulso inicial que proveniente del exterior por medio de pulsadores o interruptores está alimentado a 24 (V_{DC}) el fotodiodo activa el fototransistor del opto acoplador dejando conducir la segunda sección que es alimentada a por medio de un regulador 7805, que suministra 5 (V_{DC}), se emplea un circuito integrado 74HC14 que corresponde a un gatillo Schmitt que ayuda a prevenir el ruido que podría generarse en los opto acopladores con el afán de prevenir falsos cambios de estado e invertir la señal que ingresa hacia el Arduino, en total se manejan 8 entradas digitales.

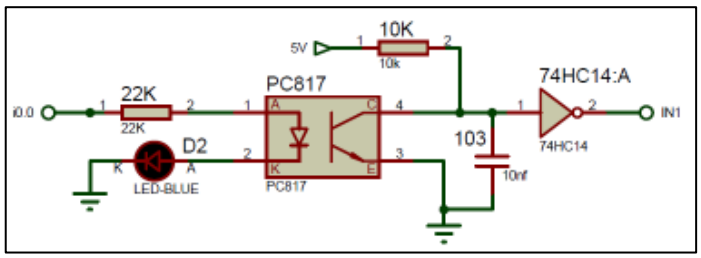


Figura 3.9 Circuito de aislamiento para entradas digitales.

Debido a que el módulo funciona a una tensión de 24 (V_{DC}), fue necesario hacer un circuito de regulación para reducir el voltaje a 5 (V_{DC}) por medio del circuito integrado

7805, ya que esta es la tensión máxima que admiten los pines digitales del microcontrolador, como se observa en la Figura 3.10.

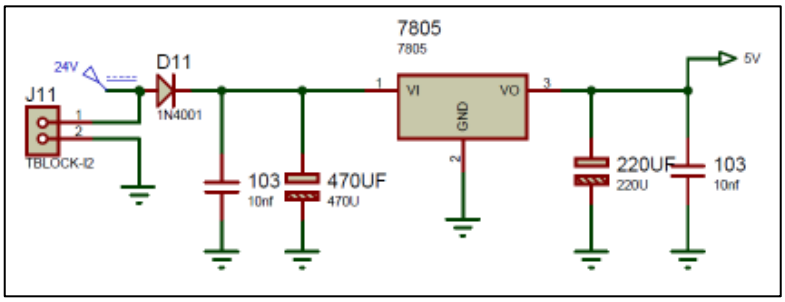


Figura 3.10 Fuente de 5 (V_{DC}).

Salidas digitales

Para este circuito no es necesario realizar simulación debido a que se empleó 2 módulos relé de 4 canales compatibles con Arduino para el manejo de cargas de 230 (V_{AC}) y 30 (V_{DC}). En la Figura 3.11 se muestra el módulo empleado para la implementación.

Figura 3.11 Módulo relé de 4 canales.

Entradas analógicas

El circuito consta de un potenciómetro que permite simular una señal analógica la cual varia de 0 a 5 (V_{DC}), también, posee una entrada que puede ser conmutada por medio de un interruptor de dos vías, como se visualiza en la Figura 3.12, esto con la finalidad de conectar señales de sensores acondicionadas en el rango de voltaje previamente mencionado.

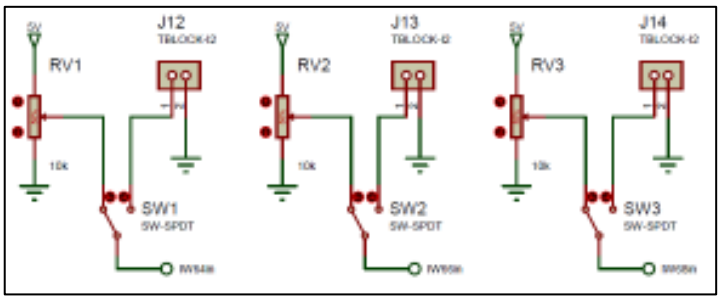


Figura 3.12 Circuito de salidas analógicas.

Como medio de aislamiento se empleó seguidores de tensión para proteger las entradas del microcontrolador, ya que estos operan en un rango de 0 a 5 (V_{DC}). El circuito integrado empleado fue el LM324, que tiene 4 amplificadores operacionales en el mismo encapsulado, ver Figura 3.13.

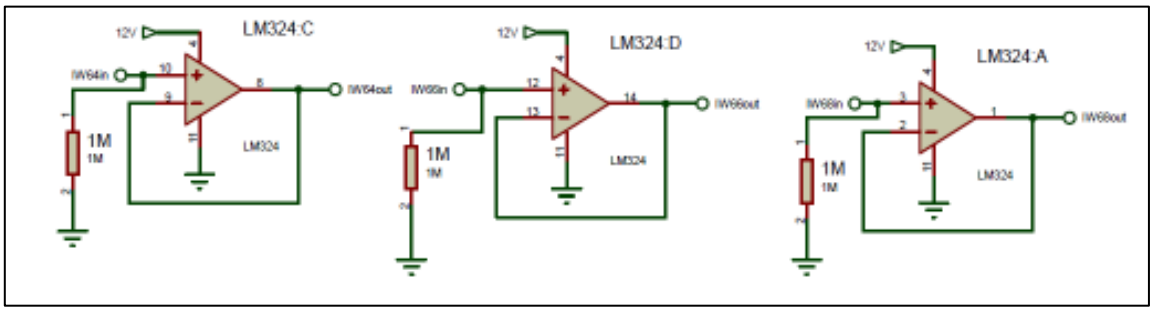


Figura 3.13 Aislamiento de entradas analógicas.

Salidas analógicas

Se creó un circuito para el control de las salidas analógicas debido a que la señal PWM generada por el Arduino tiene una corriente muy baja, inapropiada para el manejo de motores de corriente continua.

Ya que el sistema usa una fuente de 24 (V_{DC}) para su funcionamiento se emplean dos reguladores de voltaje, un 7812 para polarización de los amplificadores operacionales LM741 y un regulador 7808 que entrega 8 (V_{DC}) los cuales se encargan de alimentar el Arduino nano, como se parecía en la Figura 3.14.

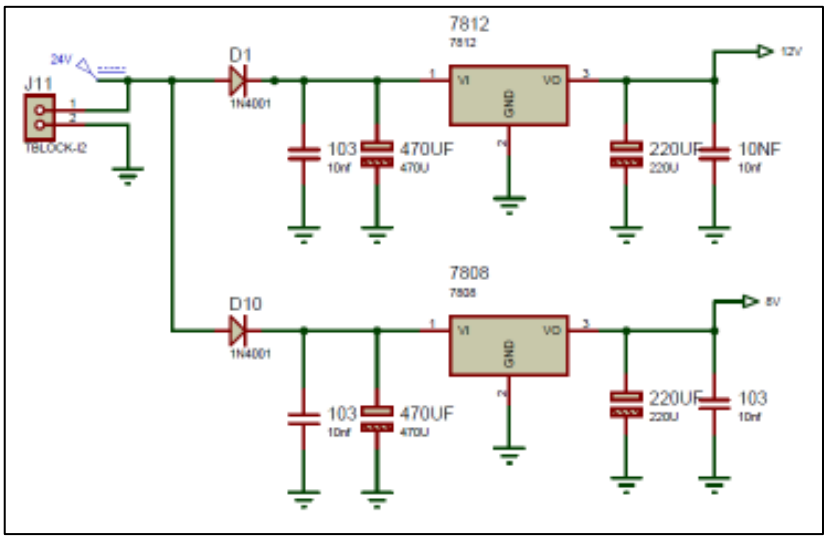


Figura 3.14 Fuentes de alimentación y polarización

La señal recibida a través de Arduino mega delegado a la comunicación con el PLC, es enviada hacia un Arduino nano, esto con la finalidad de tomar las acciones de control necesarias para mantener la señal de voltaje de salida en un rango estable de 0 a 10 (V_{DC}).

Esta información es transmitida conectando las salidas analógicas del Arduino Mega directamente con las entradas analógicas del Arduino nano por medio de un filtro RC para convertir la señal PWM en una continua, como se observa en la Figura 3.15.

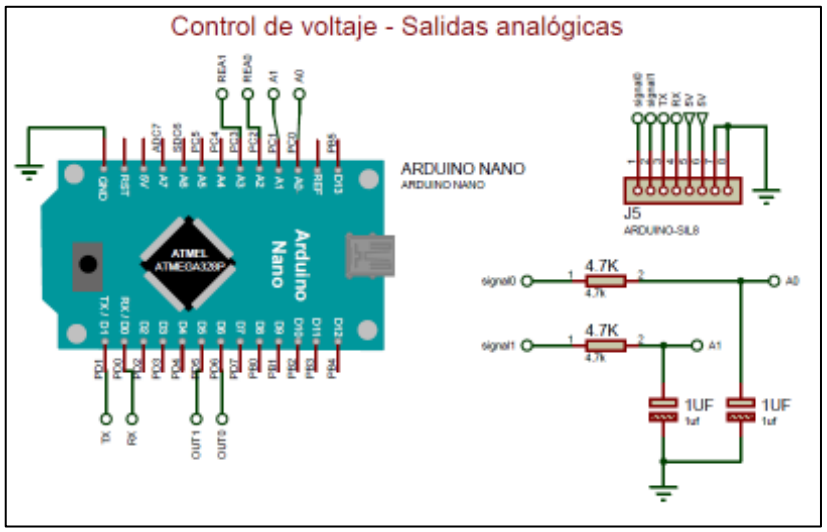


Figura 3.15 Arduino nano - pines de conexión.

La señal de entrada es reflejada en los pines correspondientes a las salidas PWM del Arduino nano que pasan por una etapa etapas de filtrado usando un comparador y seguidores de tensión, esto por medio de amplificadores operacionales LM741 como se

aprecia en la Figura 3.16, esta señal polariza la base de un transistor TIP 122 el cual amplifica la corriente pudiendo así conectar cargas que consumen hasta 1.5 amperios en corriente continua.

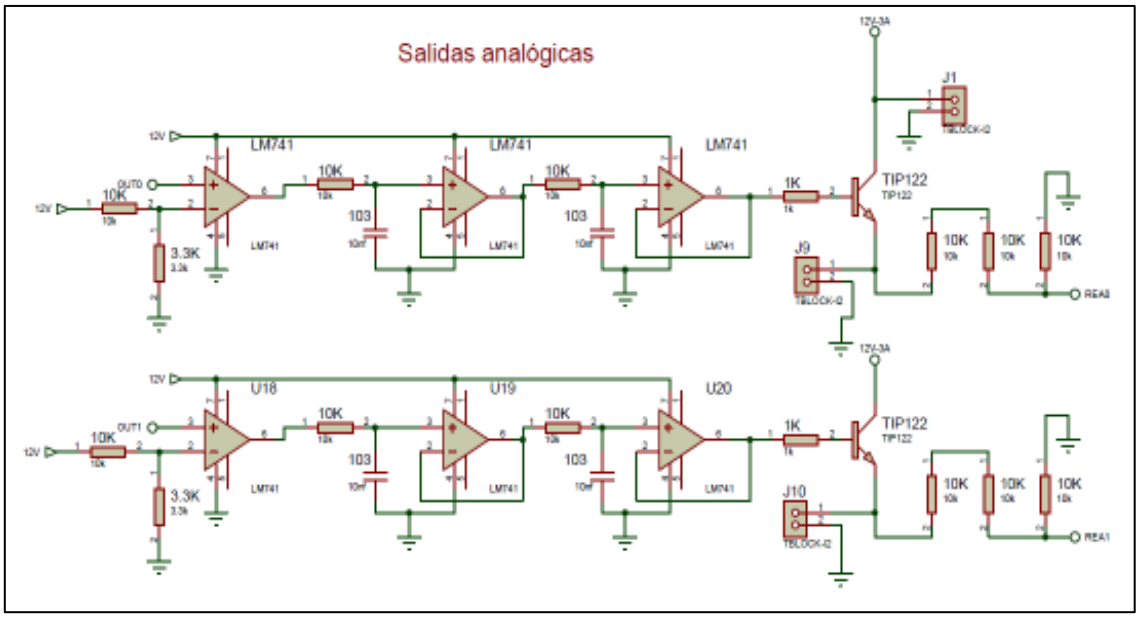


Figura 3.16 Circuito de amplificación y filtrado.

El voltaje de realimentación es tomado desde un divisor de tensión conectado al emisor del transistor, esta señal ingresa por un pin analógico de Arduino nano con el objetivo de calcular el error que existe entre la entrada y la salida, con esto el microcontrolador mantiene los valores de voltaje en el rango establecido de 0 a 10 (V_{DC}).

Creación de PCB por medio del programa Ares.

Se ubican los elementos elegidos para la implementación a la herramienta Ares que forma parte del Proteus y aquí se define el orden de todos los elementos en la placa, el ruteado de las pistas como se muestra en la Figura 3.17, así como también las dimensiones del circuito impreso el cual tiene las medidas de 18x13.5 (cm).

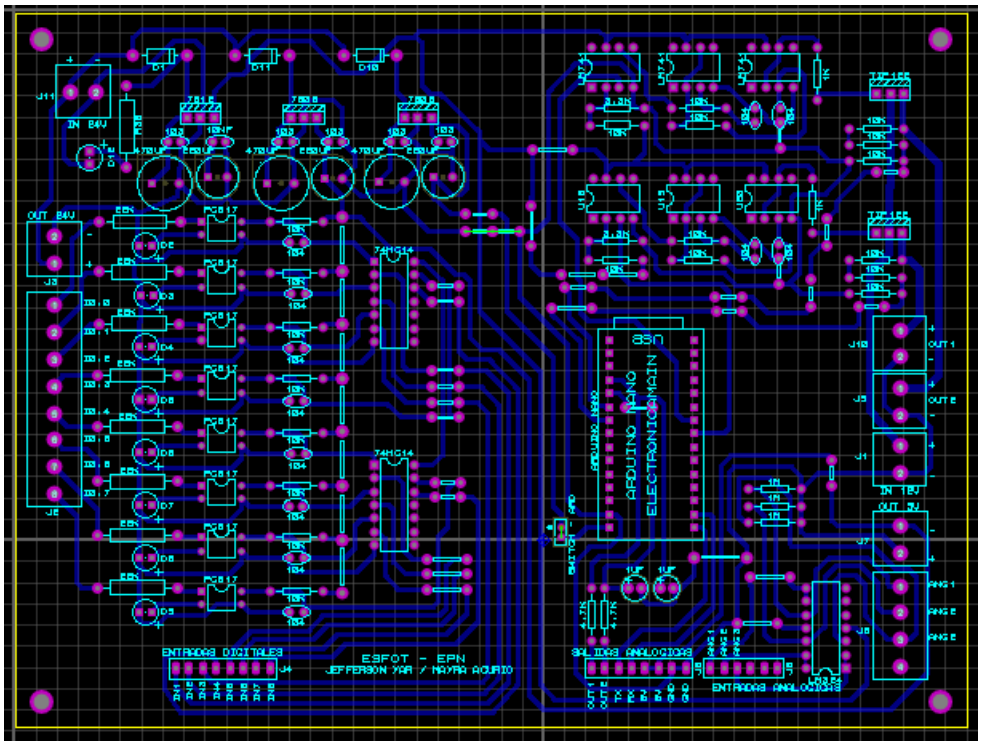


Figura 3.17 Montaje de elementos en el programa Ares.

Ares también ayuda a tener una visualización previa en 3 dimensiones de cómo se vería el circuito ya terminado con todos sus componentes montados, como se observa en la Figura 3.18. Posteriormente se exporta el archivo en formato pdf con las pistas y la máscara de componentes, para imprimirlo con una impresora láser sobre un papel brillante ya sea papel fotográfico o papel couche.

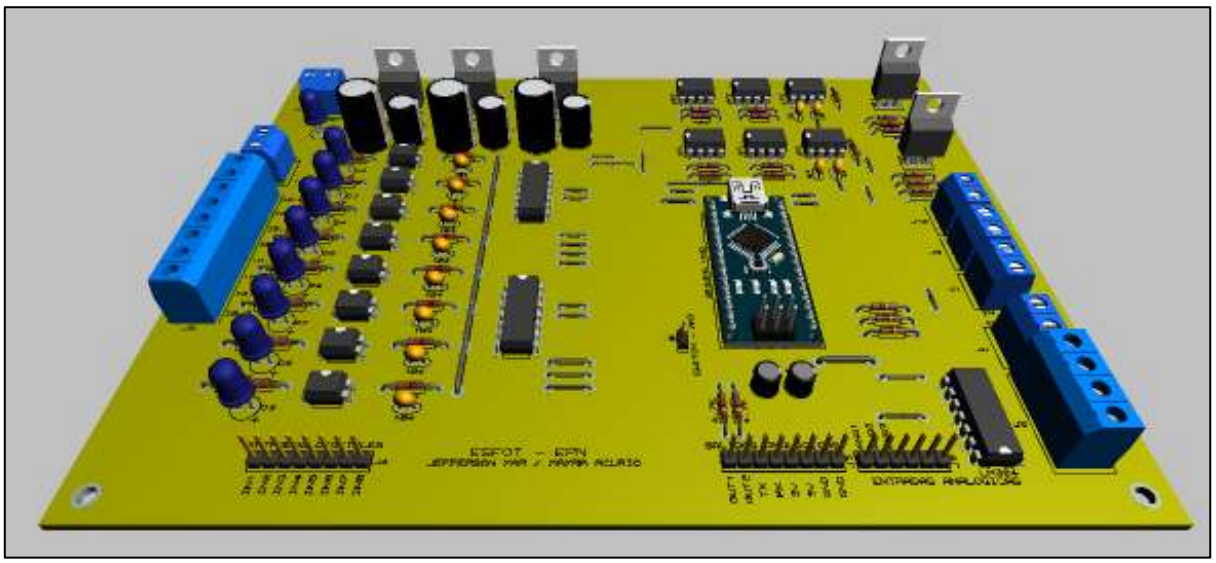


Figura 3.18 Visualización 3D de la PCB.

Modelado de estructura en SolidWorks 2018.

Se empleo el programa SolidWorks 2018 para modelar la estructura que contendrá todos los elementos, así como sus puntos de conexión. El modelo más acorde para este propósito se muestra en la Figura 3.19.

Figura 3.19 Modelado de la estructura

1. **Base para placa PCB:** en esta placa, ver Figura 3.20, es donde se coloca el circuito de acondicionamiento y adaptación del módulo, tiene las dimensiones de 19x15 (cm).

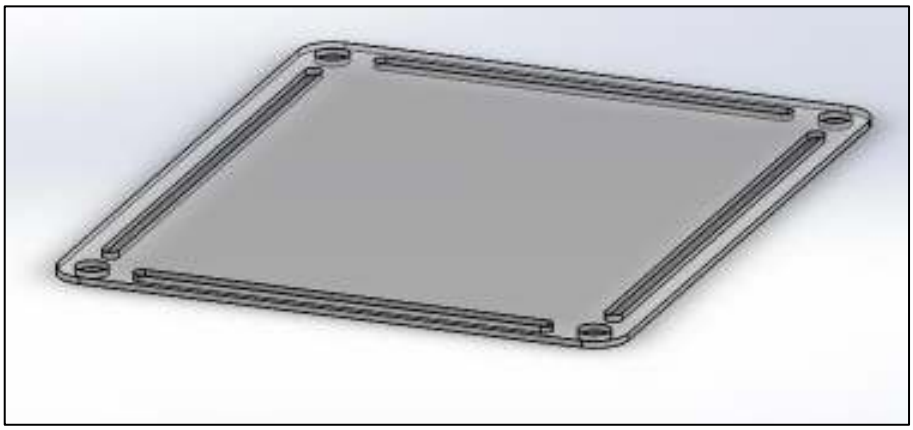


Figura 3.20 Placa para PCB

2. **Base para Arduino y fuente:** en esta placa, ver Figura 3.21, es donde se ubica la fuente que alimenta todo el sistema, como también el Arduino mega

2560 y los módulos relé encargados de las salidas digitales, tiene las dimensiones de 22 x 27.7 (cm).

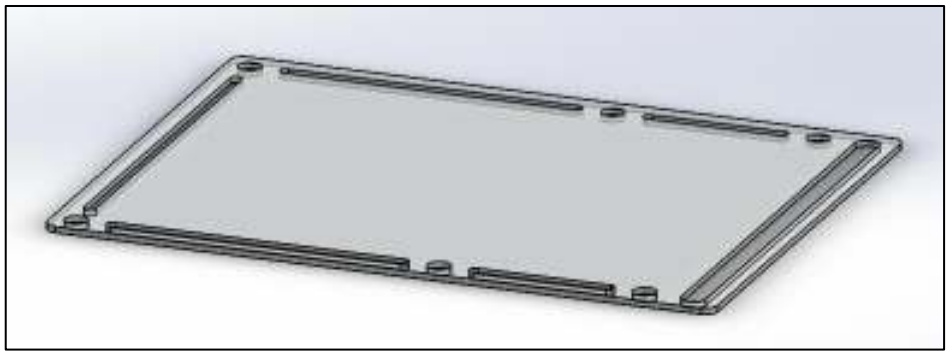


Figura 3.21 Placa para Arduino y fuente.

3. **Panel frontal:** en este panel, ver Figura 3.22, se ubican las borneras, pulsadores e interruptores del módulo de expansión. Esto con la finalidad de facilitar la conexión de los componentes que se desea controlar con el módulo. También posee el LCD de 20x4, en el cual se visualiza los datos analógicos transferidos entre Arduino y PLC**.**

Figura 3.22 Panel frontal del módulo de expansión.

4. **Placa base del módulo:** esta es la placa base donde se apoyará todo el módulo de expansión, ver Figura 3.23, está hecha en acrílico tiene un espesor de 3 (mm) y las dimensiones son 30 x 24 (cm).

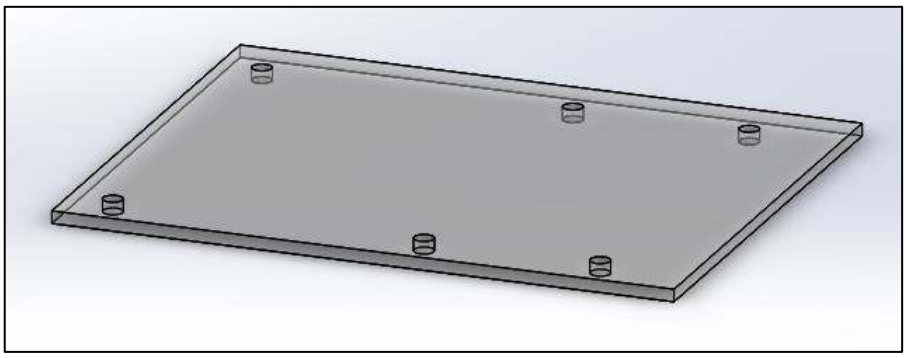


Figura 3.23 Placa base.

5. **Tornillos sin fin y tuercas:** son usados como medio de sujeción de todos los componentes empleando dos niveles para acomodar los componentes de una forma óptima y así ahorrar espacio, ver Figura 3.24.

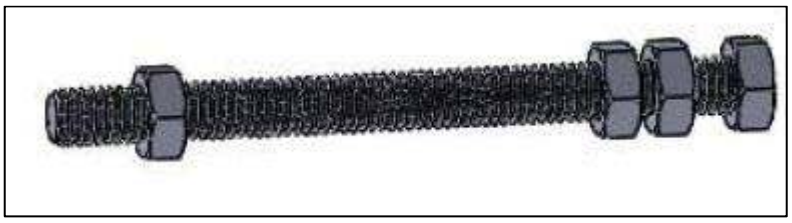


Figura 3.24 Tornillo sin fin y tuercas.

Se hizo el diseño de cada una de las piezas por separado y a través de la opción de ensamble unirlas para formar una estructura, esto es de gran ayuda para tener una idea previa de cómo se quiere que se vea el módulo y de igual manera definir el tamaño de cada una de las piezas que conforman el módulo.

Montaje que los componentes

En primer lugar, se colocó un vinil adhesivo con la información de cada uno de los puntos de conexión del módulo, para posteriormente ubicar cada uno de los elementos en el lugar correspondiente.

En la parte superior se ubica la pantalla que permite la visualización de los datos analógicos de entradas y salidas, cuenta con 4 pulsadores y 4 interruptores que permiten crear señales digitales, como se observa en la Figura 3.25.

Figura 3.25 Pantalla de visualización y entradas digitales.

También se ubicaron borneras para la conexión con los relés correspondientes a las salidas digitales, como se aprecia en la Figura 3.26. Los dos pares borneros ubicadas en la parte inferior permiten la alimentación de las cargas.

Figura 3.26 Borneras correspondientes a las salidas digitales.

En la Figura 3.27 se muestran los componentes que permiten la simulación de entradas analógicas se ubicaron los conmutadores que permiten habilitar las señales internas proporcionadas por el módulo y controladas por los potenciómetros, como también señales externas generadas por sensores acondicionadas a los valores indicados.

Figura 3.27 Potenciómetros y borneras correspondientes a entradas analógicas.

En la Figura 3.28, se indican las borneras correspondientes a las dos salidas analógicas que proporcionan voltaje variable de 0 a 10 (V_{DC}), es importante conectar una fuente externa de 12 (V_{DC}) para el funcionamiento del circuito.

Figura 3.28 Borneras correspondientes a las salidas digitales.

Finalmente se hizo el montaje y cableado de los componentes electrónicos como se muestra en la Figura 3.29, el esquema de conexión se adjunta en el Anexo 5.

Figura 3.29 Montaje, cableado de componentes electrónicos y vista frontal.

3.3 Comunicación Modbus TCP/IP entre el PLC y el módulo

Asignación de dirección IP al PLC

A continuación, se presentan los pasos a seguir para establecer la comunicación entre un PLC Siemens S7-1200 y un Arduino Mega 2560, mediante el protocolo de comunicación Modbus TCP/IP, esto con el fin de enviar y recibir datos entre los dos dispositivos.

En el programa TIA portal V15 se crea un nuevo proyecto y se escoge un dispositivo nuevo el cual debe corresponder al dispositivo fisico, en este caso el PLC que se dispone en el laboratorio es un 1212C AC/DC/RLY, Se le asigna una direccion IP unica la cual puede ser configurada haciendo click derecho sobre el dispositivo y seleccionando “Propiedades”, como se muestra en la Figura 3.30.

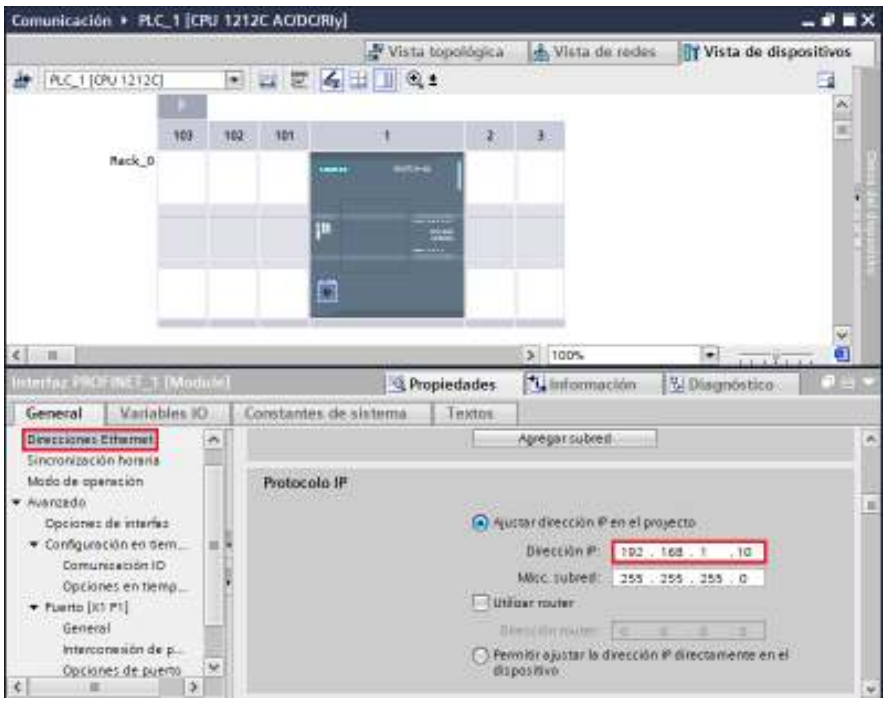


Figura 3.30 Asignación de dirección IP.

Creación de los bloques de datos

En la pestaña bloque de programa en el árbol del proyecto, se agregan cuatro bloques de datos uno para cada una de las funciones que tendrá el módulo, tal como se indica en la Figura 3.31.

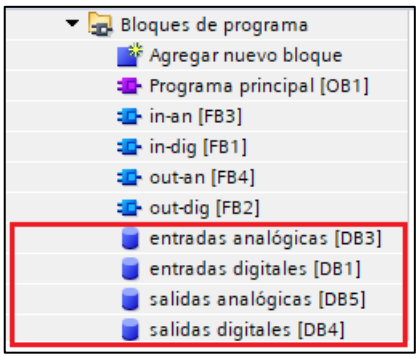


Figura 3.31 Bloques de datos creados.

Una vez creado el bloque de datos se debe crear una variable de tipo estructura "*Struct*" donde se va a guardar la configuración general del bloque de datos y la variable donde se almacena el dato proveniente desde el Arduino o desde el PLC. La configuración genérica de este bloque se puede observar en la Figura 3.32.

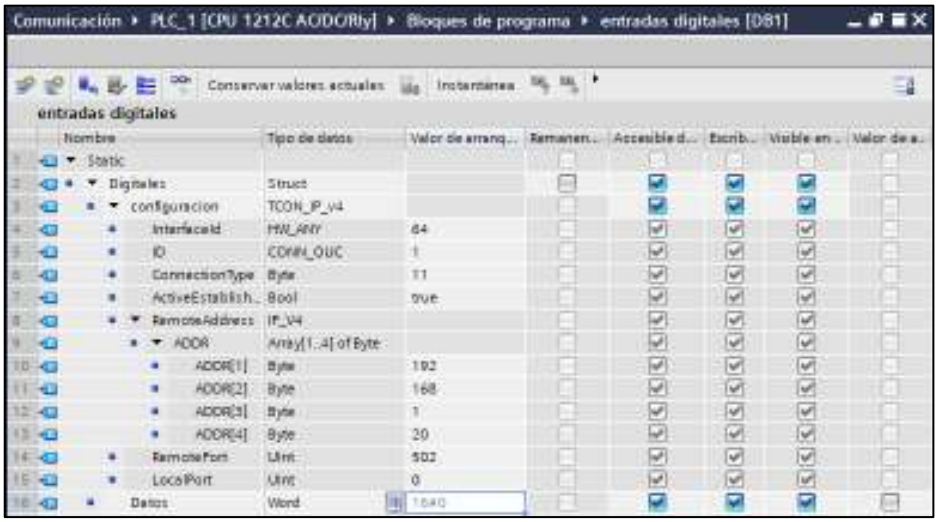


Figura 3.32 Estructura general de configuración del bloque de datos.

Descripción de los parámetros del bloque de datos

Tabla 3.1 Parámetros de configuración TCON_IP_V4.

Parámetro	Tipo de datos	Valor de Arranque	Descripción
Interface Id	HW_ANY	64	Identificador de hardware de la interfaz local.
Id	CONN_OUC	1	Referencia a esta conexión (rango de valores: de 1 a 4095).
Connection_Type	BYTE	11	Tipo de conexión: • 11: TCP

Parámetro	Tipo de datos	Valor de Arranque	Descripción
			• 19: UDP
Active_Established	BOOL	TRUE	FALSE: establecimiento pasivo de la conexión TRUE: establecimiento activo de la conexión
Remote_Address	ARRAY [1.4] of BYTE		Dirección IP del punto final del interlocutor • addr [1] = 192 • addr [2] = 168 • addr [3] = 1 • addr [4] = 20
Remote_Port	UINT	502	Dirección de puerto del interlocutor remoto
Local_Port	UINT	0	Dirección de puerto del interlocutor local

Para la configuración de cada uno de los bloques de datos los parámetros se mantienen como se indica en la Tabla 3.1, únicamente se cambia la ID de cada uno de los bloques, como se muestra en la Tabla 3.2, este dato tiene la función de identificar cada uno de los lo que datos. En el parámetro de "*Remote Address*" se ingresa la dirección IP del dispositivo con el que se va a establecer comunicación para este caso la que fue asignada al Arduino la cual es 192.168.1.20.

Tabla 3.2 Identificaciones asignadas a los datos

Dato	ID
Entradas digitales	1
Salidas digitales	2
Entradas analógicas	3
Salidas analógicas	4

Bloque MB_CLIENT

Por medio de esta instrucción es posible establecer comunicación Modbus TCP como cliente por medio de enlace ethernet del autómata S7-1200, una de las ventajas es que no es necesario adicionar un *hardware* extra para su funcionamiento. Este bloque

permite una vinculo tipo cliente servidor, en donde el autómata envía y recibe peticiones, como también controla la conexión y desconexión del servidor modbus, en la Figura 3.33 se muestra el bloque y sus pines de cada uno de los pines que lo conforman. [5]

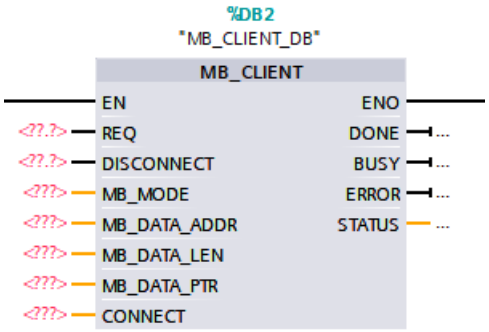


Figura 3.33 Bloque MB_CLIENT.

Descripción de los parámetros del bloque "MB_CLIENT"

En la Tabla 3.3, se observa cada uno de los parámetros que conforman el bloque de comunicación "MB_CLIENT", de igual forma se muestra el tipo de datos que emplea como la descripción de cada uno.

Tabla 3.3 Parámetros del bloque MB_CLIENT. [5]

Parámetro	Tipo de datos	Descripción
REQ	BOOL	El parámetro REQ se controla por nivel. Así, mientras la entrada esté activada (REQ=true), la instrucción enviará peticiones de comunicación.
DISCONNECT	BOOL	Mediante este parámetro se controla el establecimiento de la conexión y la desconexión con el servidor Modbus: • 0: Establecer conexión de comunicación con el interlocutor configurado en el parámetro CONNECT • 1: Deshacer la conexión. Durante la desconexión no se ejecuta ninguna otra función.
MB_MODE	USINT	Selección del modo de petición Modbus (lectura, escritura o diagnóstico) o selección directa de una función Modbus.
MB_DATA_ADDR	UDINT	Dirección del dato a enviar o recibir

Parámetro	Tipo de datos	Descripción
MB_DATA_LEN	UINT	Longitud de datos: Número de bits o palabras para el acceso a los datos.
MB_DATA_PTR	VARIANT	Puntero hacia un búfer de datos para los datos que se van a recibir desde el servidor Modbus o que se van a enviar al servidor Modbus.
CONNECT	VARIANT	Puntero hacia la estructura de la descripción de la conexión TCON_IP_v4: contiene todos los parámetros de direccionamiento necesarios para establecer una conexión programada y la conexión se establece al llamar la instrucción "MB_CLIENT".
DONE	BOOL	El bit del parámetro de salida DONE se ajusta a "1" en cuanto se ejecuta sin errores la última petición Modbus.
BUSY	BOOL	• 0: Ninguna petición Modbus en proceso • 1: La petición Modbus se está ejecutando
ERROR	BOOL	• 0: Ningún error • 1: Con errores. La causa del error se indica mediante el parámetro STATUS.
STATUS	WORD	• Información de estado detallada de la instrucción.

Parámetros MB_MODE, MB_DATA_ADDR y MB_DATA_LEN

En la Tabla 3.4, se muestra los parámetros de configuración de los pines MB_MODE, MB_DATA_ADDR y MB_DATA_LEN del bloque de función MB_CLIENT, lo cual permite establecer el modo de funcionamiento del bloque ya sea como lectura o escritura, el direccionamiento de cada uno de los datos y su longitud, es decir el número de bits por recibir o enviar.

Tabla 3.4 Parámetros de comunicación Modbus. [6]

MB_MODE	MB_DATA_ADDR	DATA_LEN	Función Modbus	Función y tipo de datos
0	de 1 a 9.999	de 1 a 2.000	01	Leer de 1 a 2.000 bits de salida en la dirección remota de 0 a 9.998
0	de 10.001 a 19.999	de 1 a 2.000	02	Leer de 1 a 2.000 bits de entrada en la dirección remota de 0 a 9.998
0	de 40.001 a 49.999	de 1 a 125	03	Leer de 1 a 125 registros de parada en la dirección remota de 0 a 9.998
0	de 30.001 a 39.999	de 1 a 125	04	Leer de 1 a 125 palabras de entrada en la dirección remota de 0 a 9.998
1	de 1 a 9.999	1	05	Escribir 1 bit de salida en la dirección remota de 0 a 9.998
1	de 40.001 a 49.999	1	06	Escribir 1 registro de parada en la dirección remota de 0 a 9.998
1	de 1 a 9.999	de 2 a 1.968	15	Escribir de 2 a 1.968 bits de salida en la dirección remota de 0 a 9.998
1	de 40.001 a 49.999	de 2 a 123	16	Escribir de 2 a 123 registros de parada en la dirección remota de 0 a 9.998
2	de 1 a 9.999	de 1 a 1.968	15	Escribir de 1 a 1.968 bits de salida en la dirección remota de 0 a 9.998
2	de 40.001 a 49.999	de 1 a 123	16	Escribir de 1 a 123 registros de parada en la dirección remota de 0 a 9.998

Creación de bloques de función

Toda la programación está contenida dentro de un bloque de función con el propósito de llamar esta función dentro del programa principal, esto con fines de practicidad. Para crear un bloque se función, en la opción “bloques de programa”, se selecciona la opción agregar bloque, se desplegará una ventana donde se selecciona “bloque de función” y se configura el nombre del bloque y el lenguaje de programación, en este caso se la realizó en lenguaje Ladder o KOP. Cabe recalcar se debe crear un bloque por cada una de las funciones que va a cumplir el módulo, como se indica en la Figura 3.34.

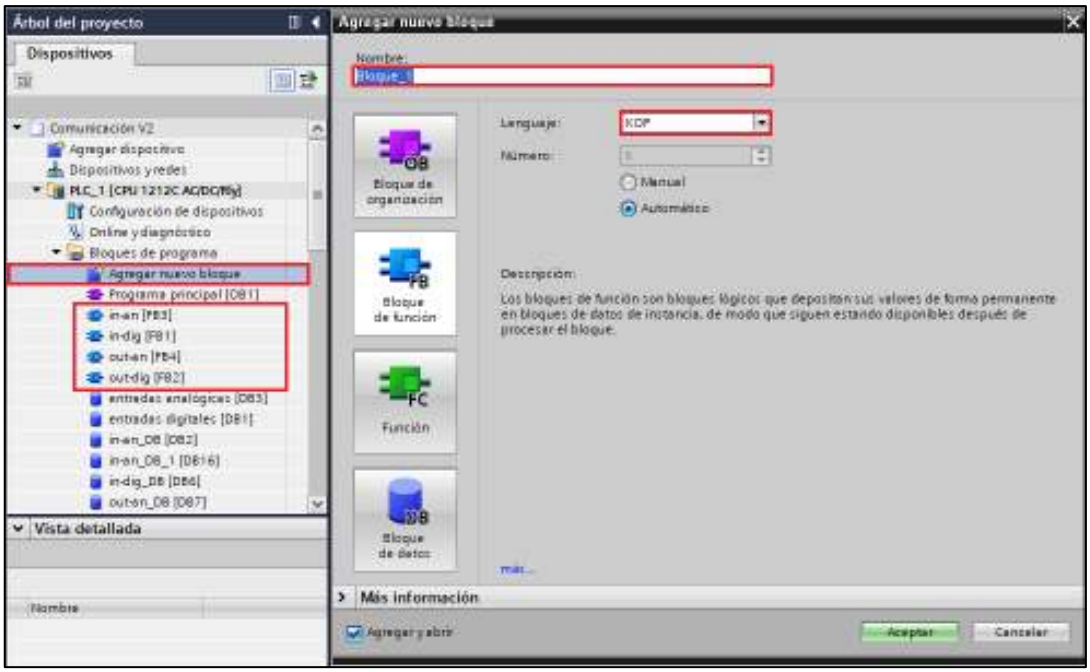


Figura 3.34 Creación del bloque de función.

- **in-an:** bloque correspondiente a entradas analógicas.
- **in-dig:** bloque correspondiente a entradas digitales.
- **out-an:** bloque correspondiente a salidas analógicas.
- **out-dig:** bloque correspondiente a salidas digitales.

Configuración del bloque de función “in-an”

En la Figura 3.35, se indica el bloque de datos correspondiente a las entradas analógicas, aquí se crean tres variables del tipo “Word” donde se guardarán los datos enviados desde Arduino una vez que se establezca la comunicación Modbus entre ambos dispositivos. Se usa este tipo de variable debido a que los datos se almacenan en registros de entrada y este tiene una longitud de 16 bits.

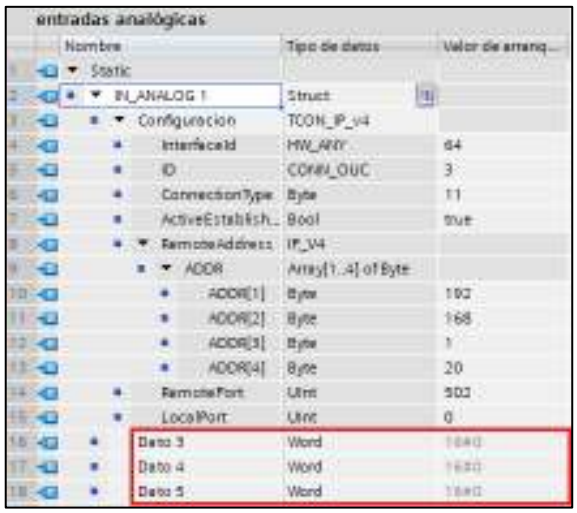


Figura 3.35 Configuración del bloque de datos (entradas analógicas).

En bloque de función “in-an”, se insertan tres bloques MB_CLIENT para establecer comunicación entre el Arduino y el PLC, en la Figura 3.36 se aprecia la configuración de cada uno de los bloques, el pin MB_MODE se le asigna el valor de 0 que le permite al autómata recibir datos desde el Arduino, las direcciones Modbus para las variables son 500, 700 y 900 las cuales se configuran en la programación de Arduino, por lo tanto, si el dato a recibir es un registro de entradas al bloque se le asignan los valores que se observan en la Tabla 3.5 y la longitud del dato es 1 correspondiente al número de datos que se pretende recibir.

Tabla 3.5 Direcciones Modbus de entradas analógicas (MB_CLIENT).

Dato	Direcciones Modbus
Entrada analógica 1	30501
Entrada analógica 2	30701
Entrada analógica 3	30901

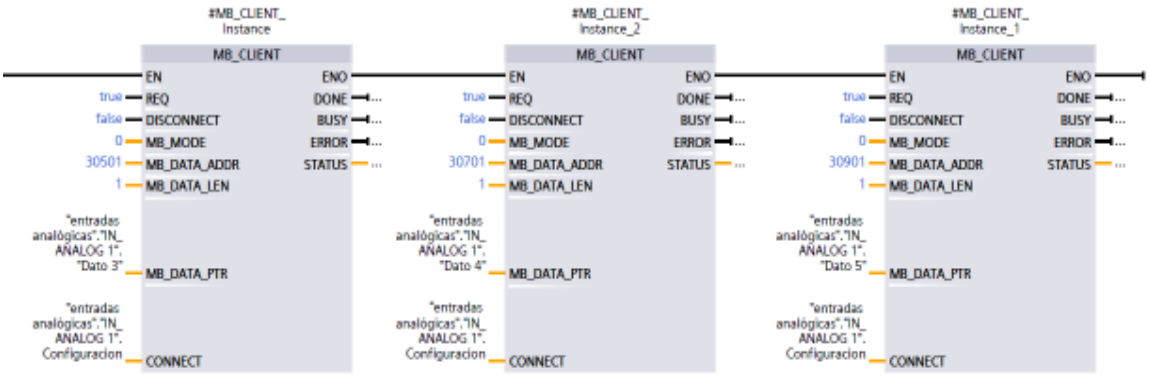


Figura 3.36 Bloques MB_CLIENT (recepción de señales analógicas).

En la opción “interfaz de bloque”, ver Figura 3.37, en el bloque de función “in-an” se configuran los pines, las etiquetas que tendrán cada uno de los mismos y así como también el tipo de variable que le va a asociar a dicho pin, en la sección output se crean tres pines donde se conectarán las tres variables que contendrá la información proveniente de los tres sensores que está leyendo el microcontrolador, cabe recalcar que las variables deben ser del mismo tipo de las que fueron creadas en el bloque de datos.

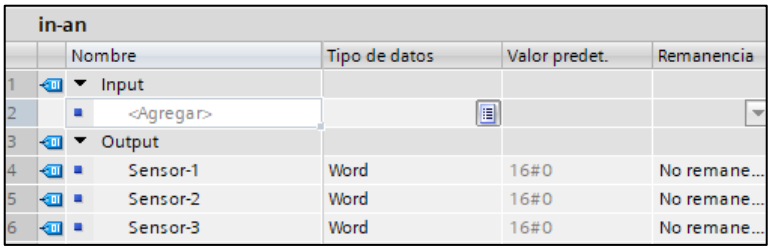


Figura 3.37 Interfaz de bloque (in-an).

Se configuran tres bloques “MOVE”, ver Figura 3.38, que se encargan de mover la información recibida en el bloque de datos hacia los pines del bloque de función creados, después de esto la información almacenada en dicha variable ya puede ser procesada por el autómata.

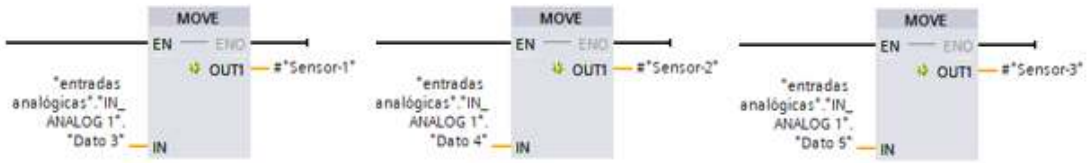


Figura 3.38 Bloques MOVE (entradas analógicas).

En el programa principal se arrastra el bloque de función “in-an” y como se aprecia en la Figura 3.39 se aprecia que posee los tres pines que se crearon en la opción “interfaz de bloque”, es a estos pines a los que se les puede asociar variables globales tipo “Word” y de esta manera poder procesar dicha información, ya sea para monitorear en una HMI como también para tomar acciones de control.

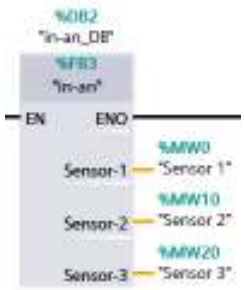


Figura 3.39 Bloque de función para entradas analógicas (in-an).

Configuración del bloque de función “in-dig”

La variable que contienen los bits provenientes de Arduino se almacena en un bloque de datos al que se le ha asignado la dirección IP del servidor, el bloque de comunicación “MB_CLIENT” apunta mediante el pin “MB_DATA_PTR” hacia el dato que se está recibiendo y este debe de estar creado en el bloque de datos, de igual manera, el pin “CONNECT” apunta a la configuración de todos los parámetros de direccionamiento necesarios para establecer una conexión cliente – servidor y toda está contenida en el bloque de datos, como se observa en la Figura 3.40.

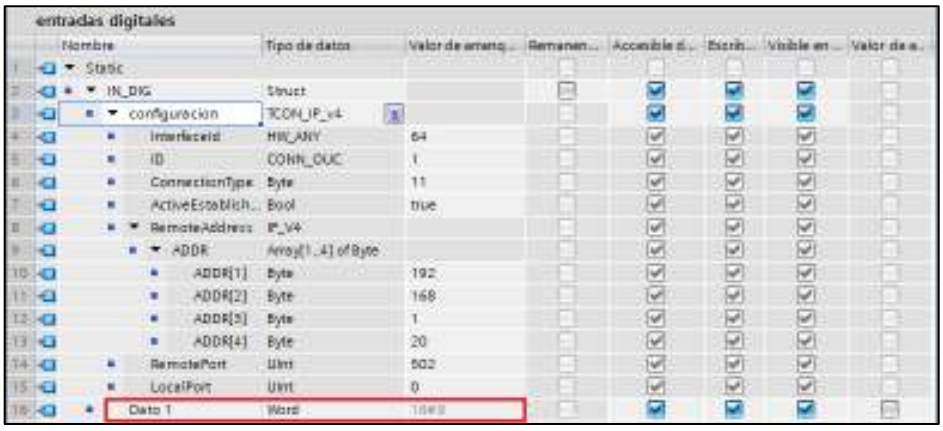


Figura 3.40 Estructura del bloque de datos (entradas digitales).

Se crea una variable de tipo Word que se compone de 16 bits asignándole la dirección MW100 y nombrándola “Pulsadores”, se toman los 8 bits menos significativos asociados a las 8 entradas digitales que se pretende recibir, como se aprecia en la Figura 3.41. Es

importante mencionar que se puede utilizar cualquier dirección, siempre y cuando se emplee los bits menos significativos.

Pulsadores - MW100															
Bits mas significativos								Bits menos significadtivos							
M100.7	M100.6	M100.5	M100.4	M100.3	M100.2	M100.1	M100.0	M101.7	M101.6	M101.5	M101.4	M101.3	M101.2	M101.1	M101.0

Figura 3.41 Bits que conforman una variable “Pulsadores”.

Teniendo esto como consideración, en la tabla de variables globales se crea una variable tipo booleano por cada uno de los 8 bits menos significativos de la variable “Pulsadores”, ver Figura 3.42, ya que es esta variable donde se moverán los bits que se encuentren en el bloque de datos. Este tipo de variables son globales esto indica que pueden ser llamadas en cualquier parte del programa.

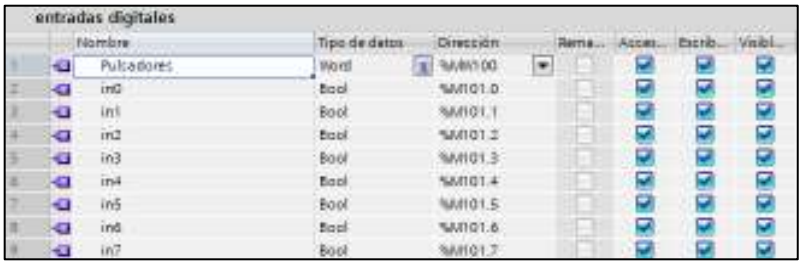


Figura 3.42 Variables donde se almacenan los datos provenientes de Arduino.

En el bloque de función “in-dig”, en la opción “interfaz de bloque” se configuran los pines y el etiquetado que tendrá el bloque de función, ver Figura 3.43, en la sección “Input” se ingresan los 8 pines correspondientes a los bits transferidos desde Arduino almacenados en la variable tipo Word creada en el bloque de datos. En la sección “Output” se ingresan los pines a los cuales se les pueden designar marcas que permitan activar o desactivar cargas.

in-dig

	Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible d...	Escrib...	Visible en ...
1	Input						
2	in-1	Bool	false	No rem...			
3	in-2	Bool	false	No remane...			
4	in-3	Bool	false	No remane...			
5	in-4	Bool	false	No remane...			
6	in-5	Bool	false	No remane...			
7	in-6	Bool	false	No remane...			
8	in-7	Bool	false	No remane...			
9	in-8	Bool	false	No remane...			
10	Output						
11	out-1	Bool	false	No remane...			
12	out-2	Bool	false	No remane...			
13	out-3	Bool	false	No remane...			
14	out-4	Bool	false	No remane...			
15	out-5	Bool	false	No remane...			
16	out-6	Bool	false	No remane...			
17	out-7	Bool	false	No remane...			
18	out-8	Bool	false	No remane...			
19	Variable	Word	16#0	No remane...			

Figura 3.43 Interfaz de bloque (in-dig).

Se inserta un bloque MB_CLIENT para establecer comunicación entre el autómata y Arduino, el pin MB_MODE se le asigna el valor de 0 que le permite recibir el dato proveniente de Arduino, la dirección Modbus para la variable es 100 la cual se configura en la programación del microcontrolador, por lo tanto, si el dato a recibir es un registro de entrada al bloque se le asigna el valor 30101 y la longitud del dato es 8 correspondientes al número de entradas, ver Figura 3.44.

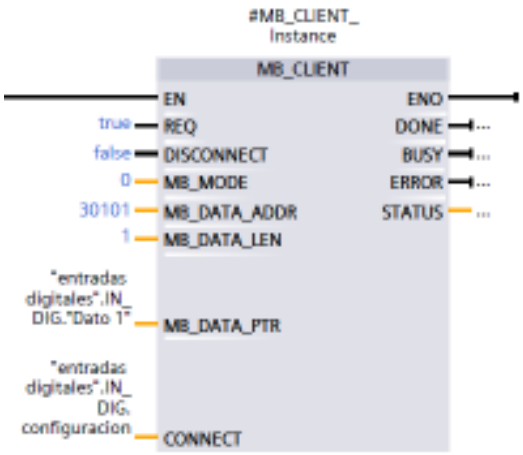


Figura 3.44 Bloque MB_CLIENT (recepción de datos digitales).

Cabe mencionar que este tipo de variables son locales, esto quiere decir que únicamente son programadas en la interfaz del bloque y actuaran únicamente dentro de este.

Debido a que el objetivo es encender o apagar marcas en la interfaz de programación del bloque de función “in-dig” se realiza el arreglo que se aprecia en la Figura 3.45, esto con el fin de crear un punto de enlace entre el bloque de datos y el programa principal

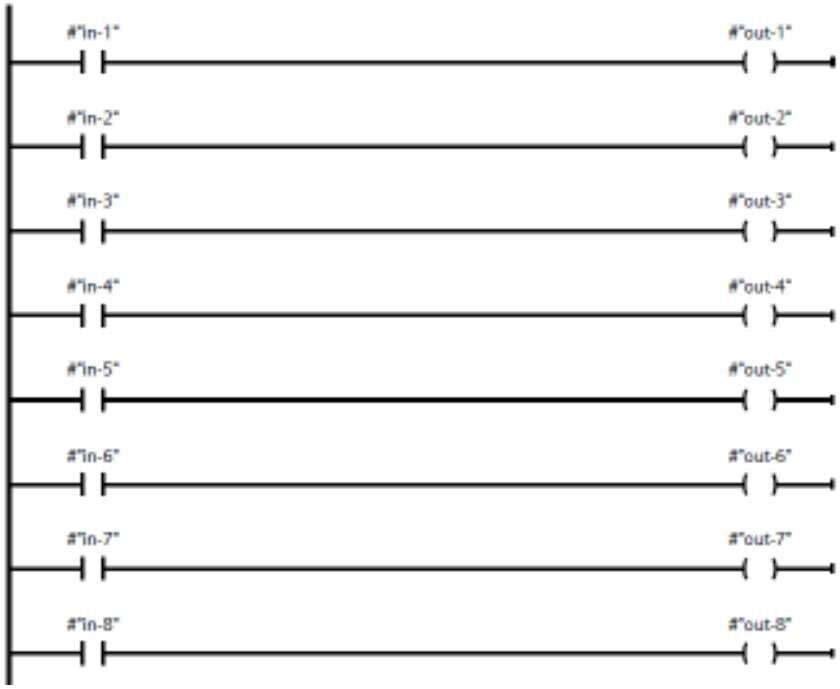


Figura 3.45 Arreglo de contactos y bobinas para entradas digitales.

Ahora se debe mover la información que se encuentra en el bloque de datos a una variable global que permita procesar dicha información para tomar acciones de control, como se aprecia en la Figura 3.46, mediante un bloque MOVE cada bit recibido desde Arduino que esta guardado en el bloque de datos se mueve a una variable local creada en el bloque de función.

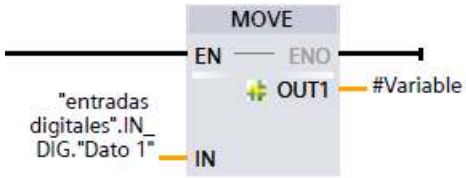


Figura 3.46 Bloque MOVE (entradas digitales).

Como este pin está asociado a una salida del bloque de función a este se le designa la variable global que contendrá los datos provenientes de Arduino, en este caso corresponde a la variable “Pulsadores”, mencionada en la Figura 3.41.

Se arrastra desde el árbol de proyectos, el bloque de función correspondiente a entradas digitales hacia el programa principal y ahora es posible designarle a los pines creados

las variables que permiten activar o desactivar marcas, ver Figura 3.47. Como variables de entrada se tiene a los bits menos significativos correspondientes a la variable “Pulsadores” que es donde se movió la información del bloque de datos.

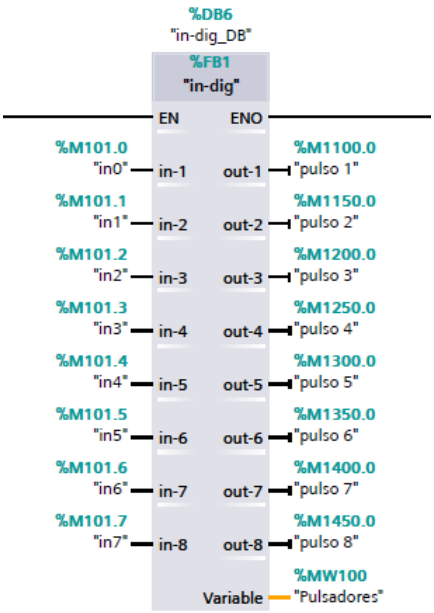


Figura 3.47 Bloque de función para entradas digitales

De igual manera las marcas designadas para la activación pueden estar en cualquier dirección, para este caso se tomó las que se indican en la Figura 3.48. El bloque de función permite activar o desactivar estas marcas a las cuales se les puede asociar una salida física del autómata o como también una luz indicadora en una HMI.

10		pulso 1	Bool	%M110.0	☐	☑	☑	☑
11		pulso 2	Bool	%M115.0	☐	☑	☑	☑
12		pulso 3	Bool	%M120.0	☐	☑	☑	☑
13		pulso 4	Bool	%M125.0	☐	☑	☑	☑
14		pulso 5	Bool	%M130.0	☐	☑	☑	☑
15		pulso 6	Bool	%M135.0	☐	☑	☑	☑
16		pulso 7	Bool	%M140.0	☐	☑	☑	☑
17		pulso 8	Bool	%M145.0	☐	☑	☑	☑

Figura 3.48 Tabla de variables de entradas digitales.

Configuración del bloque de función “out-an”

En el bloque de datos correspondiente a las salidas analógicas se crean dos variables locales del tipo “Word”, ver **Figura 3.49**, es aquí donde se guardará la información que se enviara hacia Arduino para posteriormente procesarla en una señal de voltaje que varía entre 0 y 10 (V_{DC}). Se usa este tipo de variable debido a que los datos se almacenan en Registros de retención y este tiene una longitud de 16 bits.

Figura 3.49 Bloque de datos (Salidas analógicas)

En bloque de función “out-an”, se insertan dos bloques MB_CLIENT para establecer comunicación entre el autómata y el microcontrolador, en la Figura 3.50 se aprecia la configuración de cada uno de los bloques, el pin MB_MODE se le asigna el valor de 1 que le permite escribir datos desde el autómata hacia el microcontrolador.

Las direcciones Modbus para las variables son 1100 y 1300 las cuales se configuran en la programación de Arduino, por lo tanto, si el dato a enviar es un registro de retención al bloque se le asignan los valores que se observan en la Tabla 3.6 y la longitud del dato es 1 correspondiente al número de registros que se pretende enviar.

Tabla 3.6 Direcciones Modbus de salidas analógicas (MB_CLIENT).

Dato	Direcciones Modbus
Salida analógica 1	41101
Salida analógica 2	41301

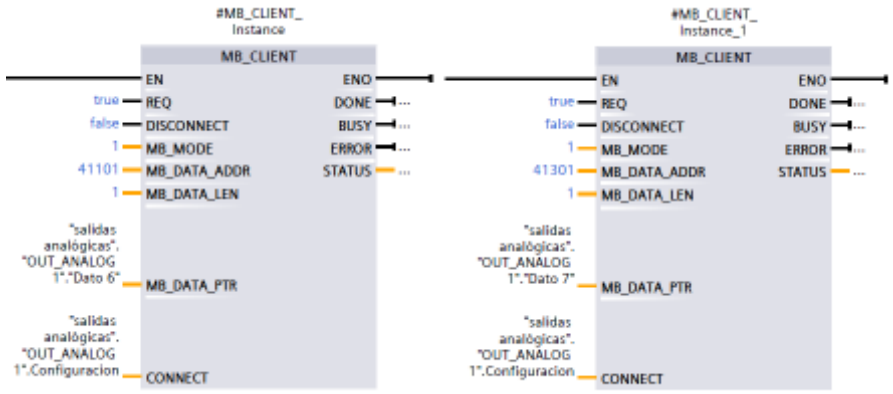


Figura 3.50 Bloques MB_CLIENT (envío señales analógicas).

Es posible simular datos analógicos de dos maneras, se puede configurar un deslizador en un interfaz humano máquina para asignarle el rango máximo y mínimo, esto se consigue por medio del bloque LIMIT como se aprecia en la Figura 3.51.

Se asigna un rango a una variable del tipo “Int” y después asociarla a algún medio de control en este caso un deslizador, para finalmente mover dicho valor a un bloque de datos que permita con un bloque de comunicación MB_CLIENT enviarlo hacia el microcontrolador.

Figura 3.51 Bloque LIMIT.

El otro método es procesar la información proveniente de la entrada analógica del PLC, transformar los datos en valores enteros que puedan ser almacenados en una variable global y posteriormente cargarlos en la variable local creada en el bloque de datos correspondiente a salidas analógicas.

En la Figura 3.52, se observa los bloques de normalización y escalamiento que permiten convertir las unidades crudas del conversor análogo digital (ADC) del autómata a valores enteros comprendidos en un rango que se puede establecer según el propósito, para este estudio se estableció el rango de 0 a 100, que son los valores que se va a enviar por comunicación Modbus hacia el microcontrolador para posteriormente ser convertidos en señales de voltaje.

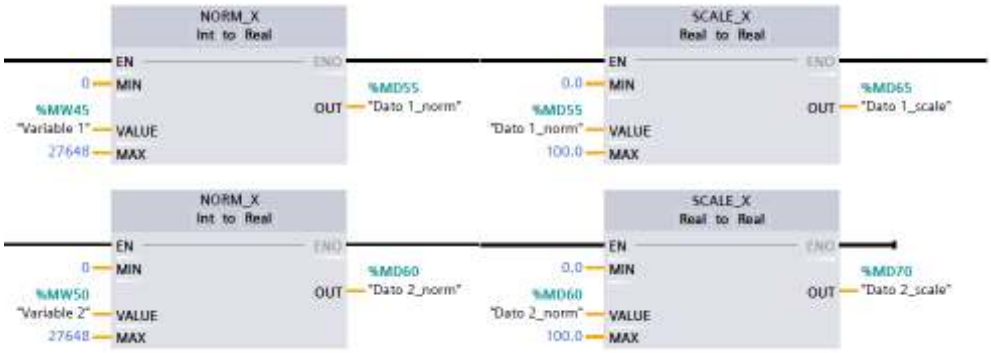


Figura 3.52 Bloques normalización y escalamiento.

En el bloque de función “out-an”, se ingresa a la opción “interfaz del bloque” donde se asignan los pines de entrada y salida que tendrá el bloque de datos, ver Figura 3.53, en la sección de entradas se crean pines que permitan ingresar los rangos máximos y mínimos de los valores que se va a enviar vía Modbus hacia el microcontrolador, esto con la finalidad de ingresar los rangos de una forma más cómoda y fácil.

Cabe mencionar que, si se cambia el rango este también debe ser cambiado en la programación de Arduino por lo que es recomendable trabajar en el rango de 0 a 100. En la sección de “Output” se establecen dos pines para verificar el valor que se está enviando (Salida 1 y Salida 2) y dos pines para verificar si se está trabajando con datos del HMI (int) o provenientes de la entrada analógica del autómata (ext).

out-an

	Nombre	Tipo de datos	Valor predet.	Remanencia
1	Input			
2	Range 1_min	Int	0	No rem...
3	Range 1_max	Int	0	No remane...
4	Range 2_min	Int	0	No remane...
5	Range 2_max	Int	0	No remane...
6	Output			
7	int	Bool	false	No remane...
8	ext	Bool	false	No remane...
9	Salida 1	Int	0	No remane...
10	Salida 2	Int	0	No remane...

Figura 3.53 Interfaz de bloque (out-an).

Debido a que se está simulando señales analógicas por medio de dos maneras, la primera procesando la información que vienen del exterior por medio de las entradas analógicas del autómata y la segunda simulando por medio de un deslizador configurado en un interfaz humano máquina. Es necesario configurar en el interfaz humano maquina un botón que permita conmutar entre estas dos formas de leer señales y de esta manera emplear solo 2 bloques de comunicación.

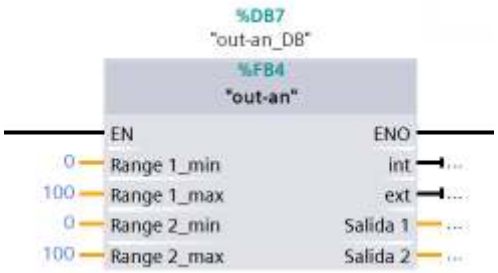


Figura 3.54 Bloque de función para salidas analógicas (out-an).

En la Figura 3.54, se observa que el bloque de función “out-an” consta con los pines para configurar los rangos mínimos y máximos de los campos de entrada del interfaz humano maquina por donde se ingresara la información que enviara.

También posee 2 pines que indican en qué modo se encuentra operando el bloque, ya sea leyendo valores del exterior o los valores internos ingresados por los campos de entrada, los pines de Salida 1 y 2 indican el valor que se está enviando a cada una de las salidas.

Configuración del bloque de función “out-dig”

La variable que contienen los bits que se va a enviar hacia Arduino se almacena en un bloque de datos al que se le ha asignado la dirección IP del servidor, el bloque de comunicación “MB_CLIENT” apunta mediante el pin “MB_DATA_PTR” hacia la variable que contiene el dato que se está enviando y que debe de estar creado en el bloque de datos, de igual manera, el pin “CONNECT” apunta a la configuración de todos los parámetros de direccionamiento necesarios para establecer una conexión cliente – servidor y toda está contenida en el bloque de datos, como se observa en la Figura 3.55.

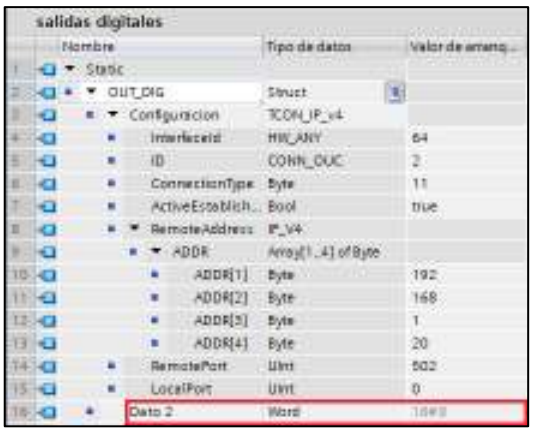


Figura 3.55 Estructura del bloque de datos (salidas digitales).

Se crea una variable global de tipo “Word” que se compone de 16 bits asignándole la dirección MW300 y se consigna el nombre de “Actuadores”, se toman los 8 bits menos significativos asociados a las 8 salidas digitales que se pretende enviar, como se aprecia en la Figura 3.56. Es importante mencionar que se puede utilizar cualquier dirección, siempre y cuando se emplee los bits menos significativos.

Actuadores - MW300															
Bits mas significativos								Bits menos significadtivos							
M300.7	M300.6	M300.5	M300.4	M300.3	M300.2	M300.1	M300.0	M301.7	M301.6	M301.5	M301.4	M301.3	M301.2	M301.1	M301.0

Figura 3.56 Bits que conforman una variable “Actuadores”.

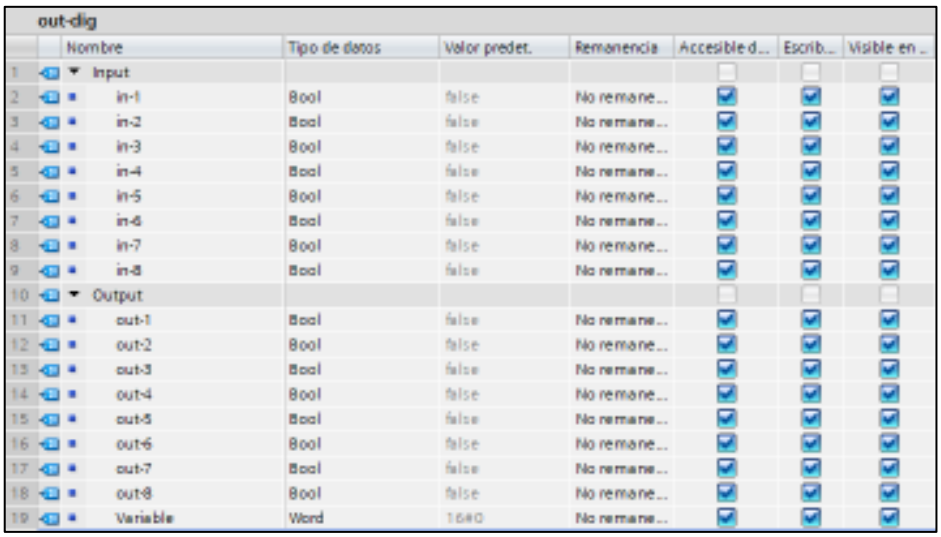
Teniendo esto como consideración, en la tabla de variables se crea una variable tipo booleano por cada uno de los 8 bits menos significativos de la variable “Actuadores” ya que es esta variable donde se moverán los bits hacia el bloque de datos, ver Figura 3.57. Este tipo de variables son globales esto indica que pueden ser llamadas en cualquier parte del programa.

		Nombre	Tipo de datos	Dirección	Rema...	Acces...	Escrib...	Visibl...
1		actuadores	Word	%MW300	☐	☑	☑	☑
2		out0	Bool	%M301.0	☐	☑	☑	☑
3		out1	Bool	%M301.1	☐	☑	☑	☑
4		out2	Bool	%M301.2	☐	☑	☑	☑
5		out3	Bool	%M301.3	☐	☑	☑	☑
6		out4	Bool	%M301.4	☐	☑	☑	☑
7		out5	Bool	%M301.5	☐	☑	☑	☑
8		out6	Bool	%M301.6	☐	☑	☑	☑
9		out7	Bool	%M301.7	☐	☑	☑	☑

Figura 3.57 Variables donde se almacenan los datos que se enviara a Arduino.

En el bloque de función “out-dig”, en la opción “interfaz de bloque” se configuran los pines y el etiquetado que tendrá el bloque de función. En la sección “Input” se ingresan los pines a los cuales se les pueden designar marcas las cuales son activadas desde un interfaz humano máquina.

En la sección “Output” se ingresan los 8 pines correspondientes a los bits transferidos desde el autómata hacia el microcontrolador almacenados en la variable tipo “Word” creada en el bloque de datos. Esta configuración puede observarse en la Figura 3.58.



out-dig

	Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible d...	Escrib...	Visible en ...
1	Input						
2	in-1	Bool	false	No remane...	✓	✓	✓
3	in-2	Bool	false	No remane...	✓	✓	✓
4	in-3	Bool	false	No remane...	✓	✓	✓
5	in-4	Bool	false	No remane...	✓	✓	✓
6	in-5	Bool	false	No remane...	✓	✓	✓
7	in-6	Bool	false	No remane...	✓	✓	✓
8	in-7	Bool	false	No remane...	✓	✓	✓
9	in-8	Bool	false	No remane...	✓	✓	✓
10	Output						
11	out-1	Bool	false	No remane...	✓	✓	✓
12	out-2	Bool	false	No remane...	✓	✓	✓
13	out-3	Bool	false	No remane...	✓	✓	✓
14	out-4	Bool	false	No remane...	✓	✓	✓
15	out-5	Bool	false	No remane...	✓	✓	✓
16	out-6	Bool	false	No remane...	✓	✓	✓
17	out-7	Bool	false	No remane...	✓	✓	✓
18	out-8	Bool	false	No remane...	✓	✓	✓
19	Variable	Word	16#0	No remane...	✓	✓	✓

Figura 3.58 Interfaz de bloque (out-dig).

Se inserta un bloque MB_CLIENT para establecer comunicación entre el Arduino y el PLC, el pin MB_MODE se le asigna el valor de 1 que le permite enviar datos desde el autómata hacia Arduino, la dirección Modbus para la variable es 300 la cual se configura en la programación de Arduino, por lo tanto, si el dato a enviar es un registro al bloque se le asigna el valor 40301 y la longitud del dato es 1 correspondientes al número de salidas, ver Figura 3.59.

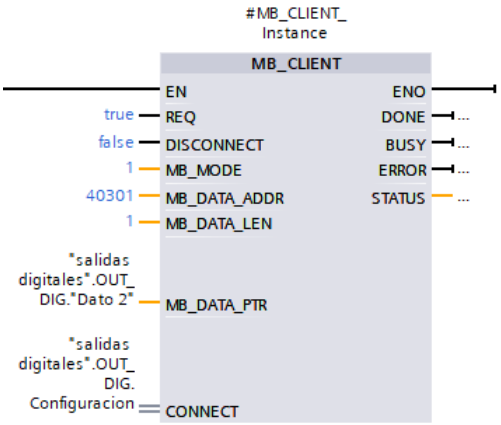


Figura 3.59 Bloque MB_CLIENT (envío datos digitales).

Cabe mencionar que este tipo de variables son locales, esto quiere decir que únicamente son programadas en la interfaz del bloque y actuaran únicamente dentro de este.

Debido a que el objetivo es encender o apagar marcas, en la interfaz de programación del bloque de función “out-dig” se realiza el arreglo que se aprecia en la Figura 3.60 esto con el fin de crear un punto de enlace entre el bloque de datos y el programa principal.

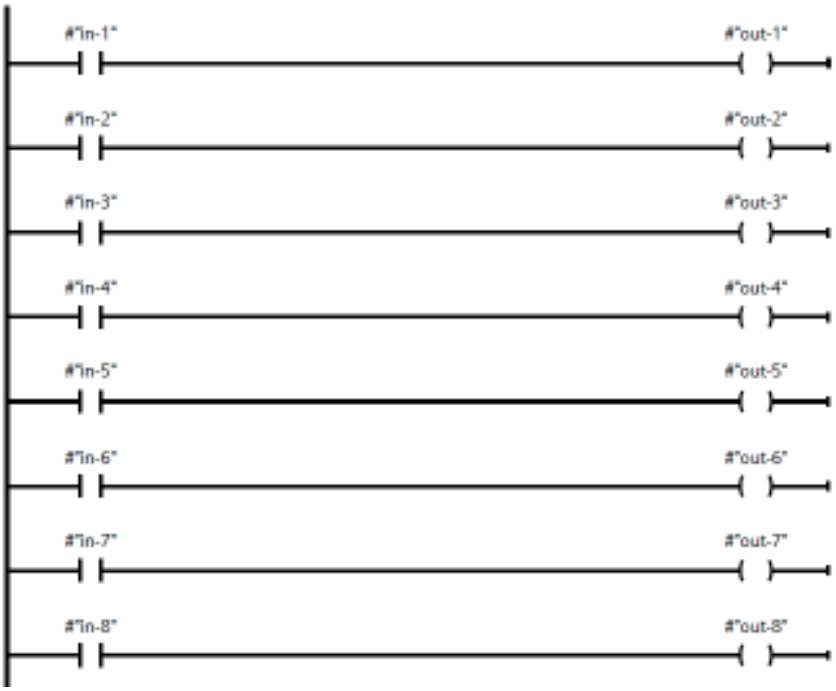


Figura 3.60 Arreglo de contactos y bobinas para las salidas digitales

Ahora se debe mover la información desde la variable global hacia el bloque de datos esto con la finalidad de enviarla hacia Arduino, como se aprecia en la Figura 3.61, mediante un bloque MOVE cada bit es enviado hacia el bloque de datos a una variable local creada en el bloque de función.

Figura 3.61 Bloque MOVE (salidas digitales)

Se arrastra el bloque de función correspondiente a las salidas digitales hacia el programa principal, ver Figura 3.62, y ahora se asigna a los pines de entrada creados las marcas que serán activadas desde una HMI o en un proceso en específico, y como salidas se tiene los bits menos significativos correspondientes a la variable “Actuadores” que es de donde se moverá la información hacia el bloque de datos.

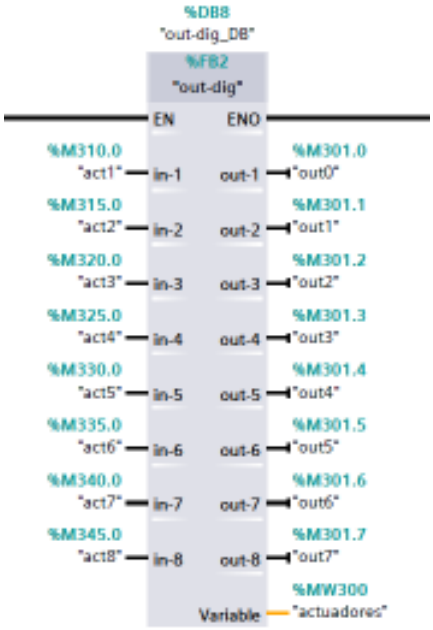


Figura 3.62 Bloque de función para las salidas digitales.

Diseño del interfaz humano máquina (HMI)

Debido a que los controladores lógicos programables que posee el laboratorio de tecnología tienen una pantalla interfaz humano maquina Siemens KTP700, se hizo una interfaz en la que constan elementos de visualización y maniobrar para comprobar que se cumple de forma adecuada la transferencia de datos entre el módulo y el controlador.

Figura 3.63 Pantalla de inicio.

En la Figura 3.63 se indica la pantalla de inicio, que consta con tres botones los cuales permiten acceder a las diferentes pantallas de visualización. Al pulsar el botón “I/O

DIGITALES” se accede a la pantalla que se muestra en Figura 3.64, la posee luces indicadoras que se encienden cuando se ha pulsado o accionado alguna de las 8 entrada digitales. De igual manera cuenta con interruptores que permiten controlar cada una de las ocho salidas digitales del módulo

Figura 3.64 Pantalla de entradas y salidas digitales.

En la Figura 3.65 se indica la pantalla que indica los datos referentes a los sensores conectados a las entradas analógicas, se observa un visualizador de curvas con el objetivo de ver como se mantiene la variable medida con respecto al tiempo, también posee tres barras indicadoras donde se observa el valor de la variable medida.

Figura 3.65 Pantalla entradas analógicas.

Finalmente se tiene una pantalla donde se configura el valor que tendrá la salida analógica del módulo como se observa en la Figura 3.66, cuenta con un visualizador de

curvas para verificar el esta valor de salida a través del tiempo, como aspecto principal es que posee un botón que permite conmutar la señal que provee a la señal analógica.

Esta puede ser externa la cual toma los valores generados por una entrada analógica del controlador y los envía para ser usados como salida analógica en el módulo de expansión, como también puede ser interna, que permite ingresar un valor entero de 0 a 100 mediante a la interfaz, lo cual equivale de 0 a 10 voltios en el módulo.

Figura 3.66 Pantalla de salidas analógicas.

Programación en Arduino IDE

Uno de los componentes importantes del sistema es el microcontrolador que posee el módulo, que es un Arduino mega 2560, el cual es el encargado de procesar la información leída por los pines del mismo y enviarla mediante vía modbus hacia el controlador.

Para que esto sea posible Arduino cuenta con librerías las cuales son “Modbus.h” y “Modbusip.h”, las cuales permiten establecer comunicación a través del protocolo de comunicación modbus con un dispositivo, únicamente asignando una dirección IP al microcontrolador y direccionando adecuadamente cada una de las variables que se pretende intercambiar con el autómata, el código de programación se adjunta en el Anexo 7.

3.4 Pruebas y Análisis de Resultados

Establecimiento de comunicación

Esto fue realizado con el propósito de verificar si todo el conjunto de componentes se encuentra en red. En la Tabla 3.7 se indica las direcciones IP asignadas.

Tabla 3.7 Direcciones IP asignadas

Componente	Direcciones IP
Ordenador	192.168.1.1
PLC Siemens S7-1200	192.168.1.10
Arduino Mega 2560	192.168.1.20
HMI KTP 700	192.168.1.30

Se conectó la placa ethernet por medio de un cable RJ45 a un conmutador y la tarjeta Arduino por medio del puerto serial hacia la PC, con el fin de cargar al programa creado. Para comprobar si la tarjeta está conectada en red con la PC y el autómata, se abre el símbolo de sistema y se ingresa el comando “ping” seguido de la dirección IP asignada en la programación de Arduino, en la Figura 3.67, se observa que el microcontrolador se encuentra conectado en red. Otra manera, es comprobar el estado de los leds “Tx” y “Rx” de la placa ethernet, ya que al estar ambos encendidos indican que el microcontrolador se ha conectado a una red.

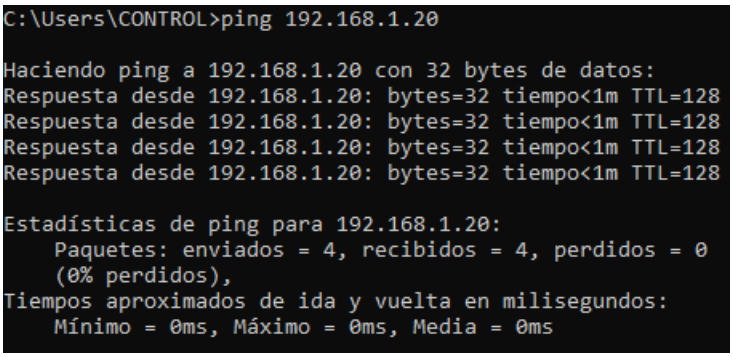


Figura 3.67 Comprobación de conexión del módulo de expansión.

De igual manera se procedió con el controlador lógico programable el cual debió tener la configuración de dirección IP cargada, para por medio del comando “ping” verificar si se encuentra en red, en la Figura 3.68, se observa los resultados arrojados que indican que se encuentra conectado en red con el ordenador.

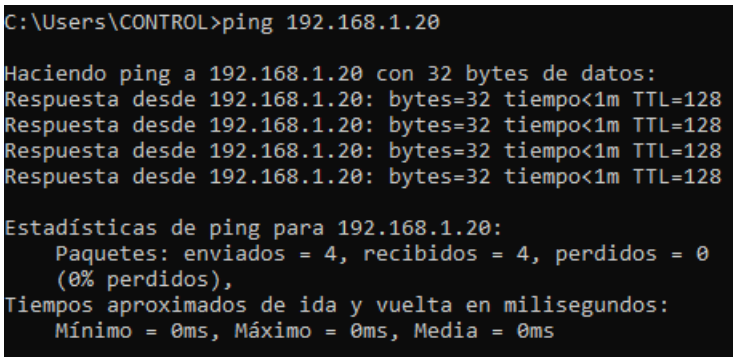


Figura 3.68 Comprobación de conexión del controlador lógico programable

Debido que se empleó un interfaz humano máquina (HMI) para control y visualización, se procedió hacer “ping” junto a la dirección IP asignada para comprobar si se encuentra en red con el sistema. En la Figura 3.69 se observa los datos arrojados por el símbolo del sistema, con lo que se verifica que la pantalla se encuentra en red.

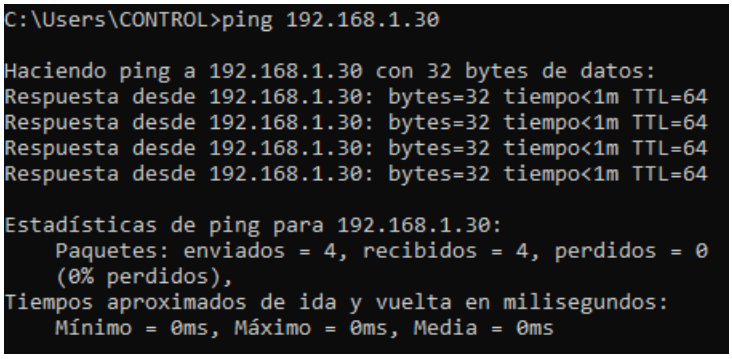


Figura 3.69 Comprobación de conexión de la pantalla HMI

Envío de datos digitales desde Arduino hacia el PLC.

Esta prueba fue realizada con el afán de comprobar si los datos digitales se envían de forma correcta desde el módulo de expansión hacia el controlador lógico programable, cabe recalcar que esto debe hacerse siempre y cuando se halla comprobado que todos los dispositivos se encuentran conectados en red.

En el módulo de expansión se accionan los pulsadores e interruptores, para verificar si existe cambio de estado se observa si los leds de la tarjeta de adaptación se encienden, como se observa en la Figura 3.70.

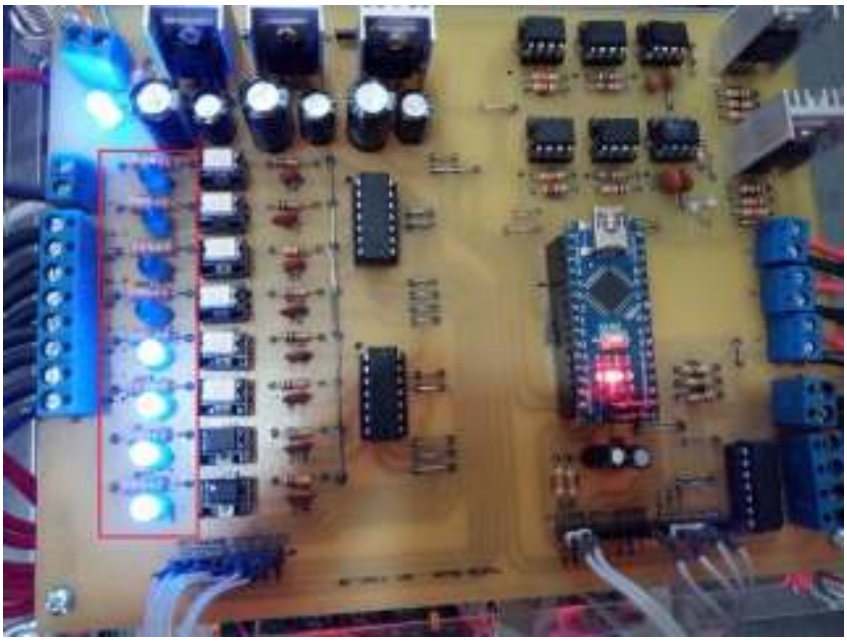


Figura 3.70 Leds indicadores de entradas digitales.

Posteriormente, se verifica en el interfaz humano máquina que se activen las luces indicadoras correspondientes a las entradas que fueron accionadas en el módulo, en la Figura 3.71, se observa que los datos digitales se envían de forma correcta desde el módulo de expansión hacia el controlador.

Figura 3.71 Luces indicadoras de entradas digitales en la HMI.

Envío de datos analógicos desde Arduino hacia el PLC.

Se realizó esta prueba con el fin de determinar que los datos analógicos son recibidos favorablemente por el autómata, se giran los potenciómetros ubicados en el panel frontal

del módulo verificando que los valores del voltaje mostrados por la pantalla LCD 20x4 coincidan con los mostrados por la interfaz del autómata.

En la Figura 3.72 se muestra que los valores de las pantallas son similares comprobando de esta manera que los datos analógicos son enviados de forma satisfactoria desde el módulo hacia el controlador.

Figura 3.72 Datos mostrados en el módulo de expansión.

De igual forma en la Figura 3.73 se observa el visualizador de curvas donde se aprecia las formas de onda de las señales de los sensores recibidas con las cuales ya es posible tomar acciones de control.

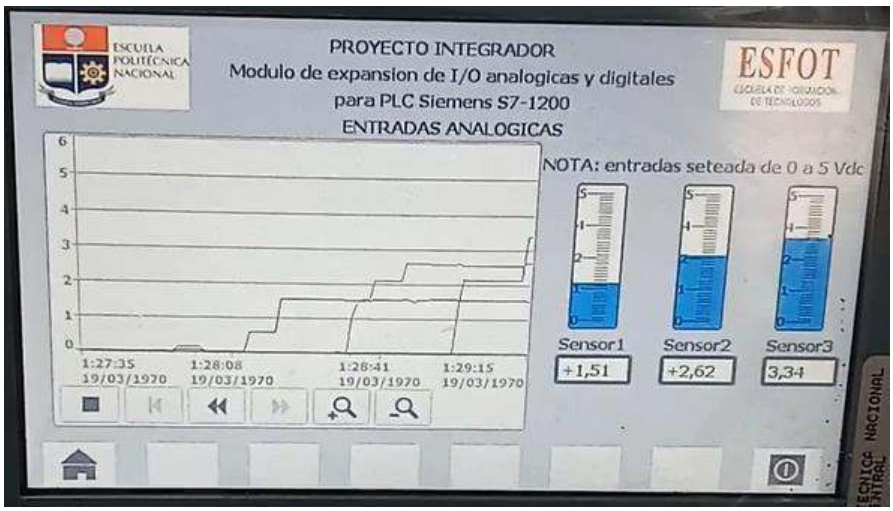


Figura 3.73 Datos mostrados en el módulo de PLC.

Envío de datos digitales desde el PLC hacia Arduino.

El objetivo de esta prueba es comprobar que los datos booleanos enviados desde el autómata, son recibidos por el microcontrolador para de accionar los relés que incorpora el módulo.

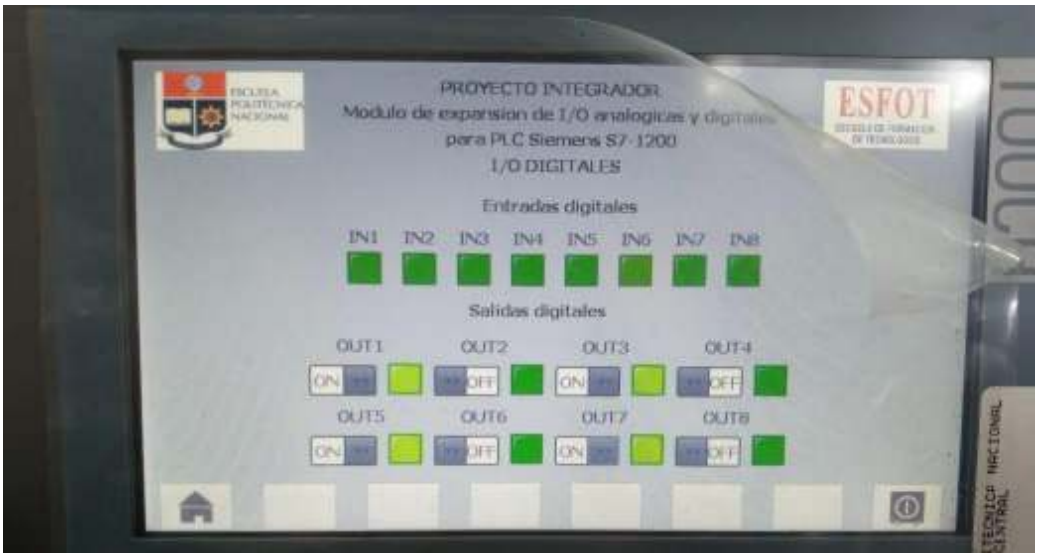


Figura 3.74 Interruptores correspondientes s salidas digitales accionados.

En la Figura 3.74, se aprecia cómo se accionan los interruptores incorporados en el interfaz humano máquina haciendo que los relés cambien de estado como se observa en la Figura 3.75, y como los contactos con capaces de soporta hasta 10 (A) es posible conectar cualquier tipo de carga hasta 230 (V_{AC}), como también conectar cargas has 30 (V_{DC}), con esto se demuestra que el envío de datos desde el autómata hacia el módulo es satisfactorio.

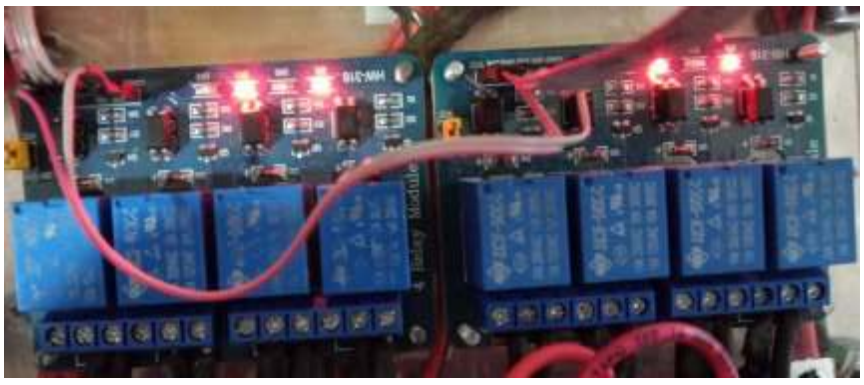


Figura 3.75 Leds indicadores del módulo relé

Envío de datos analógicos desde el PLC hacia Arduino.

Esta prueba se hizo con el fin de probar el funcionamiento las salidas analógicas que posee el módulo de expansión, en la Figura 3.76, se observa que en el interfaz humano maquina se selecciona la opción de “externa” y se varían los potenciómetros incorporados en el panel frontal del gabinete del controlador.

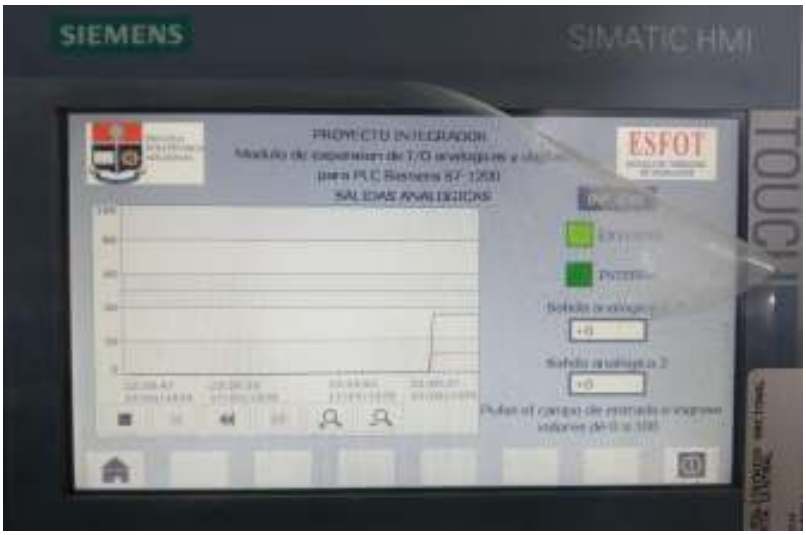


Figura 3.76 Selección de opción "externa" para control de salidas analógicas

Se observa la pantalla del módulo de expansión que los valores recibidos coinciden con los enviados por el controlador lógico programable.

Ahora se selecciona la opción de “interna” y se ingresa por medio de los campos de entrada de la interfaz un numero entre 0 y 100, los cuales representan el voltaje de la salida analógica 0 y 10 (V_{DC}), respectivamente, como se observa en la Figura 3.77.

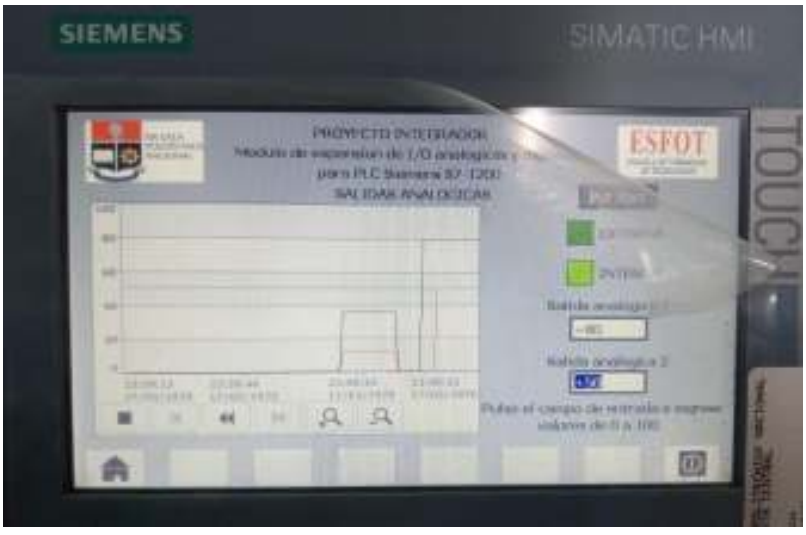


Figura 3.77 Selección de opción "interna" para control de salidas analógicas

De igual manera se comprueba que los datos ingresado por la interfaz coincidan con los datos mostrados por la pantalla del módulo, ya que los datos ingresados fueron 80 que equivale a 8 (V_{DC}) y 50 que equivale a 5 (V_{DC}), como se visualiza en la Figura 3.78.

Figura 3.78 Datos analógicos recibidos por el módulo de expansión.

Se verifica que el circuito de acondicionamiento esté funcionando adecuadamente, en primer lugar, se conecta una fuente de 12 (V_{DC}) para polarizar los transistores TIP122, mediante un multímetro se mide el voltaje en cada una de las borneras correspondientes a las salidas analógicas y se comprueba que el valor leído por el instrumento coincida con el mostrado por la pantalla del módulo.

Un punto de consideración es verificar el correcto funcionamiento del control de voltaje realizado por el microcontrolador incorporado en la tarjeta de adaptación del módulo, ya que este es un control de tipo proporcional (P) y se lo realizo obteniendo los valores leídos por el pin analógico de la consigna y por el pin analógico de la realimentación.

Esto se lo hizo por medio del puerto serial y de una macro en Excel llamada PLX-DAQ V2, la cual obtiene esta información y la tabula en una hoja de cálculo con el fin de graficarla y verificar como responde. En la Figura 3.79, se aprecia grafica correspondiente al control de voltaje que hace el microcontrolador para tratar de mantener el voltaje de salida contante.

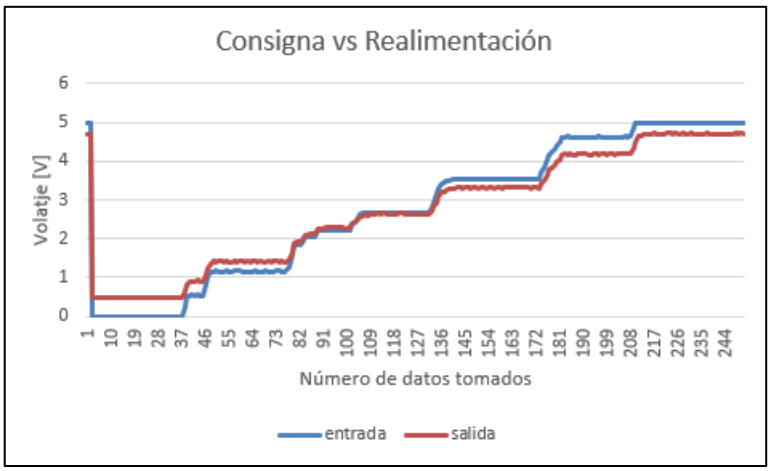


Figura 3.79 Control de voltaje - consigna vs realimentación

También es importante verificar cómo se comporta el sistema en vacío y con carga, para esto se tomaron medidas de voltaje por pasos, subiendo paulatinamente el voltaje de uno en uno desde 0 hasta 10, tanto sin ninguna conexión en las borneras de salida como también conectando un motor de corriente continua con una potencia de 24 (W) existente en el laboratorio de tecnología industrial, los datos arrojados con esta prueba se muestran en la Tabla 3.8.

Tabla 3.8 Valores tomados de las salidas analógicas.

Valor PLC	Valor módulo de expansión (V_{DC})	Salida analógica 1 - (V_{DC})		Salida analógica 2 - (V_{DC})	
		Sin carga	Con carga	Sin carga	Con carga
0	0	0	0	0	0
10	1.0	1.1	1.05	1.06	0.9
20	2.0	2.07	2.05	2.1	2.05
30	3.0	3.05	3.06	3.2	3.1
40	4.0	4.1	4.06	4.5	4.1
50	5.0	5.1	5.07	5.2	5.1
60	6.0	6.05	6.1	6.3	6.2
70	7.0	7.1	7.1	7.3	7.2
80	8.0	8.2	8.1	8.3	8.05
90	9.0	9.2	9.1	9.3	9.2
100	10.0	10.2	9.6	10.3	9.7

Por medio de la ejecución de este proyecto se evidenció que es posible aumentar el número de entradas y salidas, ya sea digitales o analógicas para un controlador lógico programable Siemens S7-1200 a bajo costo, como se puede observar en el Anexo 9.

3.5 Manual de Uso y Mantenimiento

Para revisar con más detalle la implementación del módulo de expansión, se recomienda revisar el video explicativo correspondiente al mismo, el cual es re direccionado por medio del código QR que se muestra a continuación en la Figura 3.80.

Figura 3.80 Código QR – Manual de uso.

De igual forma para tener una idea sobre el mantenimiento que se debe hacer a cada una de las partes del módulo se recomienda revisar el video que se redirecciona por medio del código QR que se indica a continuación en la Figura 3.81.

Figura 3.81 Código QR – Manual de mantenimiento.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Por medio de la implementación del módulo se comprobó que el costo total de inversión para el incremento de entradas y salidas es aproximadamente la cuarta parte de lo que costaría adquirir los módulos de fábrica, ya que el costo aproximado de estos es de alrededor de $950, considerando que cada módulo se vende por separado y el módulo de expansión implementado tuvo un costo de $230.
- En el trabajo realizado se evidencia que es posible implementar un sistema de expansión de entradas y salidas con el uso de materiales accesibles como un microcontrolador arduino mega 2560 ya que posee un gran número de pines analógicos y digitales, y por ser una plataforma de código abierto existe gran variedad de información con respecto a su programación y lo hace viable para este proyecto.
- Gracias al uso del software Proteus y Ares fue posible diseñar y dimensionar correctamente el circuito de acondicionamiento de señales, asegurando el correcto funcionamiento del mismo y de igual forma planificar el correcto costeo de los componentes que lo conforman ya que esto es parte fundamental de todo el sistema.
- Debido a que las salidas analógicas deben ser precisas se optó por la implementación de un circuito de control de voltaje con un microcontrolador Arduino nano, en la práctica los resultados obtenidos sin carga son muy aceptables y si se comparan con los obtenidos con carga, se tiene una pequeña caída de tensión a causa de la saturación del transistor por la corriente que este conduce.
- En el circuito de salidas analógicas el valor de la salida no alcanza el valor de referencia esto es causado por el tipo de controlador (proporcional) cuya desventaja es que presenta un error en estado estacionario.
- Se diseñó cada una de las partes estructurales del módulo con el programa SolidWorks esto, con el propósito de tener una idea previa de cómo se vería ya construido y además de dimensionar cada una de las partes tratando de optimizar el espacio y permitiendo observar cada una de sus partes constitutivas.
- Al usar el bloque “MB_CLIENT” se comprobó que es posible establecer una comunicación por medio del protocolo modbus TCP/IP con una topología cliente

servidor y sobre todo que fue posible conectarse con un dispositivo de plataforma libre como arduino con esto se abren las posibilidades de realizar proyectos más grandes.

- Con el objetivo de optimizar el código de programación y hacerlo de una forma más ordenada en el controlador lógico programable, se eligió el uso de bloques de función, de esta manera se puede llamar estas funciones en cualquier parte del programa principal.
- Como medio de supervisión y control se hizo un interfaz humano maquina la cual fue cargada a una pantalla Siemens KPT700, de esta manera es posible visualizar los pulsos digitales y señales analógicas provenientes del módulo y de igual forma enviar datos digitales y analógicos hacia el módulo.
- Gracias al uso de las librerías “modbus.h” y “modbusip.h”, fue posible cargarle la configuración necesaria al microcontrolador para arrancarlo como un servidor modbus y de esta manera hacer que obedezca cada una de las peticiones que le hace el autómata.
- Una de las formas más óptimas para él envió de datos digitales es el uso de registros de entrada ya que este tipo de dato posee 16 bits de los cuales se le asocian los 8 bits menos significativos a las entradas digitales que posee el módulo.
- En la actualidad el uso de las herramientas virtuales como medio académico es fundamental por su practicidad, pensando en esto se hizo la edición de dos videos en los cuales se indican los pasos a seguir para el uso y comprensión del módulo, como también el mantenimiento que debe tener cada una de sus partes, con el fin garantizar que se use el módulo de forma adecuada y verificar que cada una de las partes constitutivas no presenten signos de daño.

4.2 Recomendaciones

- Es importante verificar que la tensión de alimentación del módulo se encuentre dentro del rango esto con el propósito de proteger la fuente de alimentación que incorpora el módulo.
- Si se observa que el led indicador del botón de encendido del módulo se enciende, pero al accionarlo no arranca el sistema es probable que el fusible de protección este averiado, es recomendable verificar su estado y en el caso de estar averiado, proceder a reemplazarlo por uno de 1.5 (A).
- Como una forma visual de comprobar si la placa ethernet está conectado en red es verificar el estado de los leds “Rx” y “Tx”, ya que si los dos se encuentran encendidos es un claro indicio de que la comunicación ha sido establecida de forma satisfactoria.
- Es muy importante no superar el rango establecido de 0 a 5 (V_{DC}) para las entradas analógicas ya que los seguidores de tensión incorporados adaptan las señales para que sean leídas de forma correcta por el microcontrolador, pero si se supera su valor máximo es muy probable que el pin correspondiente sufra daños.
- Para poder tener una mayor comprensión de cómo se implementa el protocolo modbus en el PLC Siemens S7-1200, se recomienda que se observe la información sobre el bloque de comunicación que provee la pestaña de ayuda en el programa PORTAL TIA V15.
- Ya que una de las ventajas del microcontrolador usado es su gran número de pines, a futuro se podría plantear la implementación de un módulo con un mayor número de prestaciones.
- Con respecto a las salidas digitales, es importante comprobar que las cargas que se vayan a conectar cumplan con los parámetros de corriente y voltaje indicados, esto con el propósito de no averiar los relés.
- Para un correcto funcionamiento de las salidas analógicas es recomendable conectar una fuente externa de 12 (V_{DC}) que sea mayor a 3 (A), ya que cada salida tolera una carga máxima de 1.5 (A).
- En caso de mantenimiento o desconexión tener muy en cuenta la polaridad de la tarjeta de adaptación al instante de conectarla, ya que al conectarla de forma errónea podría provocar daños en algún componente del circuito.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] U. Flanklin, Interviewee, *Siemens Ecuador.* [Entrevista]. 7 Mayo 2020.

[2] A. F. Hurtado, «EEYMUC,» 20 Febrero 2020. [En línea]. Available: https://www.eeymuc.co/. [Último acceso: 16 Abril 2020].

[3] «Modbus,» [En línea]. Available: http://www.modbus.org.

[4] N. Instrument, «Información Detallada sobre el Protocolo Modbus,» Texas, 2019.

[5] T. PORTAL, «Bloque "MB_CLIENT",» Siemens, 2020.

[6] T. PORTAL, «Parametro TCON_IP_V4,» TIA PORTAL , 2020.

[7] I. Siemens, «S7 1200 CONSTROLADOR PROGRAMABLE MANUAL DE SISTEMA,» Washington, 2015.

[8] A. F. H. Banguero, «EEYMUC,» 17 junio 2020. [En línea]. Available: https://www.eeymuc.co/modulos-de-expansion-para-plc/. [Último acceso: 19 enero 2020].

[9] R. Menocal, «Rubin Menocal tecnología,» 18 Septiembre 2017. [En línea]. Available: https://rubinmenocal.wordpress.com/2017/09/18/modelo-tcpip-y-iso/. [Último acceso: 17 junio 2020].

[10] J. M. Merinero, «DISEÑO DE INFRAESTRUCTURA DE RED Y SOPORTE INFORMÁTICO PARA UN CENTRO PÚBLICO DE EDUCACION INFANTIL Y PRIMARIA,» ESCUELA UNIVERSITARIA DE INFORMÁTICA UNIVERSIDAD POLITÉCNICA DE MADRID , Madrid, 2010.

[11] C. A. S. SERNA, «Buses de campo y protocolos en redes industriales,» Universidad de Manizales, Manizales, 2011.

[12] J. J. R. Barastegui, «Diseño y desarrollo de una red MODBUS RTU basada en Arduino,» Universidad de Sevilla, Sevilla, 2017.

[13] D. F. A. ESPIN, «DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL QUE CONTENGA COMUNICACIÓN MODBUS RTU Y MODBUS TCP PARA LA IMPLEMENTACION DE SISTEMAS DE CONTROL SUPERVISORIO Y ADQUISICIÓN DE DATOS A BAJO COSTO.,» Escuela Politécnica Nacional, Quito, 2018.

[14] J. E. P. RUEDA, «DESARROLLO DE UN PROTOTIPO DE SOFTWARE PARA ADQUISICIÓN Y GESTIÓN DE DATOS DE EQUIPOS EN REDES INDUSTRIALES, PARA LA EMPRESA HIESELAT S.A,» Escuela Politécnica Nacional , Quito, 2019.

[15] R. D. D. Brucil y A. P. Guzman Herrera, «DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDACTICO PARA LA INTEGRACIÓN DE REDES DE CAMPO INDUSTRIAL: MODBUS, PROFIBUS, PARA ACTUADORES ELÉCTRICOS,» Escuela Politécnica Nacional, Quito, 2016.

ANEXOS

ANEXO 1: CERTIFICADO DE FUNCIONAMIENTO.

ANEXO 2: DISEÑO DEL CIRCUITO.

ANEXO 3: PROGRAMACIÓN DE ARDUINO NANO (CONTROL DE VOLTAJE).

ANEXO 4: PIEZAS DEL MÓDULO DE EXPANSIÓN.

ANEXO 5: ESQUEMA DE CONEXIÓN DEL MÓDULO DE EXPANSIÓN.

ANEXO 6: ALGORITMO DE CONTROL EN PORTAL TIA V15.

ANEXO 7: PROGRAMACIÓN DE ARDUINO MEGA 2560 (COMUNICACIÓN).

ANEXO 8: MASCARA FRONTAL.

ANEXO 9: LISTADO DE COSTOS.