UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería En Mantenimiento Eléctrico.

IMPLEMENTACIÓN DE UN SISTEMA HMI MEDIANTE APLICACIONES DE CÓDIGO ABIERTO PARA EL CONTROL Y MONITOREO DE UN SISTEMA DINÁMICO REAL

Trabajo de grado presentado ante la Universidad Técnica del Norte previo a la obtención del título de grado de Ingeniero en Mantenimiento Eléctrico.

Autor:

Sergio Rodolfo Torres Gualsaqui

DIRECTORA

Ing. Eliana Carolina Ormeño Mejía M.Sc.

Ibarra, 2021

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004138796		
APELLIDOS Y NOMBRES:	TORRES GUALSAQUI SERGIO RODOLFO		
DIRECCIÓN:	OATVALO		
EMAIL:	srtorresg@utn.edu.ec		
TELÉFONO FIJO:	-	**TELÉFONO MÓVIL:**	0989560791

DATOS DE LA OBRA	
TÍTULO:	IMPLEMENTACIÓN DE UN SISTEMA HMI MEDIANTE APLICACIONES DE CÓDIGO ABIERTO PARA EL CONTROL Y MONITOREO DE UN SISTEMA DINÁMICO REAL
AUTOR (ES):	TORRES GUALSAQUI SERGIO RODOLFO
FECHA: DD/MM/AAAA	20/10/2021
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERÍA EN MANTENIMIENTO ELÉCTRICO
ASESOR /DIRECTOR:	ING ORMEÑO ELIANA MSC.

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 20 días del mes de octubre de 2021

EL AUTOR:

.....

Sergio Rodolfo Torres Gualsaqui

C.I 1004139796

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ACEPTACIÓN DEL DIRECTOR

Ing. Eliana Ormeño MSc.

CERTIFICA

Que después de haber examinado el presente trabajo de investigación elaborado por el señor estudiante: Torres Gualsaqui Sergio Rodolfo que ha cumplido con las normas establecidas en la elaboración del trabajo de investigación titulado: **"IMPLEMENTACIÓN DE UN SISTEMA HMI MEDIANTE APLICACIONES DE CÓDIGO ABIERTO PARA EL CONTROL Y MONITOREO DE UN SISTEMA DINÁMICO REAL "** para la obtención del título de Ingeniero en Mantenimiento Eléctrico.

Ing. Eliana Ormeño Mejía MSc.
DIRECTOR DE TRABAJO DE GRADO

RESUMEN

Los sistemas de automatización modernos requieren de sistemas de Supervisión, Control y Adquisición de Datos (SCADA) e Interfaz Hombre Máquina (HMI) para el control y monitoreo de plantas, procesos y actividades industriales. La implementación de estos sistemas conlleva a las empresas, y proyectos académicos a invertir grandes sumas de dinero y conseguir operadores especializados para la adaptación de estos sistemas dinámicos.

En este trabajo se presentan resultados sobre el diseño e implementación de un dispositivo HMI basado en aplicaciones de código abierto para controlar y monitorear un sistema de presión y temperatura de agua dentro del laboratorio de control, en la carrera de Electricidad perteneciente la Universidad Técnica del Norte. Se utilizó el microcomputador Raspberry Pi 3B como un controlador, conectado a una pantalla LCD Touch de 5 pulgadas para formar el sistema HMI que consta del conjunto hardware y software. El diseño de la Interfaz Gráfica de Usuario (GUI) se realizó con Programación Python y la herramienta Qt Designer basado en widgets clásicos de tecnología de código abierto. La comunicación entre Raspberry y PLC S71200 se realiza mediante el protocolo Modbus TCP/IP, además, se utiliza el protocolo de comunicación Modbus RTU conexión RS-485 para la comunicación del PLC con el variador de frecuencia Altivar312.

Finalmente, la programación en Python escrito en el editor Visual Studio Code junto con la librería PyModbus permitió transferir datos de lectura y escritura en formato real flotante, desde la base de datos dentro del programa del PLC, para controlar los actuadores y monitorear las variables de temperatura, presión y nivel de agua mediante sensores.

Palabras clave: Interfaz Hombre Máquina. Raspberry Pi, código abierto.

ABSTRACT

Modern automation systems require Supervision, Control and Data Acquisition (SCADA), and HMI (Human Machine Interface) systems for the control and monitoring of plants, processes, and industrial activities. The implementation of these systems leads companies and academic projects to invest large amount of money and train operators to adapt these dynamic systems.

In this work, results are presented on the design and implementation of an HMI device based on open-source applications to control and monitor a water pressure and temperature system in the control laboratory, in the Electricity career at the Universidad Técnica del Norte. The Raspberry Pi 3B microcomputer was used as a controller, connected to a 5-inch LCD Touch screen to develop the HMI system containing hardware and software. The design of the Graphical User Interface (GUI) was carried out with Python Language and the Qt Designer tool based on classic open-source technology widgets. Communication between Raspberry and S7-1200 PLC is done using Modbus TCP/IP protocol. In addition, the communication protocol Modbus RTU connection RS-485 is used for the communication of the PLC with the Altivar312 frequency inverter.

Finally, the Python programming written in Visual Studio Code editor together with PyModbus library allowed to transfer reading and writing data in real floating format, from the database within the PLC program, to control the actuators and monitor the temperature, pressure and water level variables through sensors.

Keywords: Human Machine Interface. Raspberry Pi, open source.

DEDICATORIA

Este trabajo de investigación es dedicado a mis padres Manuel Torres y Yolanda Gualsaqui, que con mucho esfuerzo me apoyaron con todos los recursos necesarios, sus consejos y su motivación para estudiar y culminar la excelente carrera de Electricidad.

A mi tutora Eliana Ormeño por su iniciativa para encaminar este tema de investigación y permitirme trabajar bajo su dirección. Trabajo que ha tenido magníficos resultados, además, por sus ideas que han sido gran aporte en el desarrollo de este trabajo. De igual manera, a todos mis docentes de la carrera quienes me han compartido su invaluable conocimiento durante todo mi trayecto académico que me han ayudado a profundizar mis conocimientos.

A Danny, Marco, Dayana y compañeros de curso con quienes he compartido esta hermosa trayectoria en la que hemos compartido numerosos trabajos y proyectos invaluables, y a todas las personas que desinteresadamente han demostrado su apoyo a lo largo de este proceso de aprendizaje.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

A.1. CONTEXTO DE LA INVESTIGACIÓN

La automatización de procesos productivos es uno de los avances más representativos de la tecnología industrial actual (Baret, 2019). Las soluciones de automatización modernas son sistemas que, cada vez más, monitorizan activos inteligentes, máquinas y empresas para ofrecer un análisis y control (Chakraborty, 2013).

Desde la automatización industrial las empresas utilizan los sistemas de Supervisión, Control y adquisición de datos (SCADA) para supervisar y controlar los procesos industriales desde la distancia, ya que proporciona monitoreo en tiempo real e interacción directa con los sensores y actuadores (Filali, Carina S. , & Lecuona, 2015).

De la misma manera, la creación de aplicaciones tecnológicos similares de código abierto FOSS (Free or Open Source Software) han permitido el libre intercambio de información, y gran parte de tecnologías industriales subyacentes se han desarrollado a partir de este modelo de distribución que se basa en los principios de colaboración en el ámbito de automatización (Portalo, Gonzáles , Manuel, & Antonio, 2019).

El uso de FOSS también han permitido el crecimiento económico e industrial en sociedades capitalistas, al igual que la colaboración para el desarrollo intelectual. Las aplicaciones Open Source según (Atenas, Havemann, & Priego, 2015) es considerado un recurso invaluable para las comunidades académicas, en actividades de formación de doctorado, biblioteconomía, estudios clásicos, comunicaciones, ingeniería, arqueología, humanidades digitales, etc.

Con la innovación tecnológica, han aparecido diversas aplicaciones de Código abierto. Las placas como Raspberry Pi son usados para laboratorios de ingeniería, Tecnologías industriales e ideas de proyectos automatizados con el objetivo de ayudar al conocimiento vertical con respecto a materias de ingeniería (Asenjo, y otros, 2017).

Para la creación de proyectos según (Merchant & Ahire, 2017) el modelo de Raspberry Pi en comparación con otras placas (SBC) es más económico y compatible con la gestión

de tareas complejas por que se considere una placa robusta. Además, este dispositivo fiable y económico consume baja energía.

Para la adquisición de datos de una placa SBC de Código abierto según (Arias , Jiménes , & Porras, 2016) es posible usar lenguajes de programación que faciliten portabilidad y el acceso a las librerías necesarias para la representación de sistemas dinámicos.

Es así como, el diseño de un sistema HMI cumple la tarea de informar al operador de todas las actividades del proceso en las industrias actuales, por tanto, es necesario que exista una buena comunicación entre dispositivos o programas (Filali, Carina S. , & Lecuona, 2015), esta interacción se logra mediante las interfaces graficas de Usuario.

A.2. PLANTEAMIENTO DEL PROBLEMA

En la actualidad, diversos sistemas de producción, actividades industriales y los sistemas eléctricos manejan procesos automatizados. Estos procesos ayudan a mejorar los tiempos de entrega y estándar de calidad (Ambriz, Arreguín , & Ledesma, 2014), además de reducir costos de producción. La mayoría de estos sistemas necesitan de una interfaz hombre- máquina (HMI), la cual es una herramienta que usa el operario para el control y monitoreo de los procesos de producción en tiempo real. Esta herramienta obliga a las empresas a realizar grandes inversiones por los costos elevados que conlleva la adquisición de estos dispositivos.

A nivel de universidades, se realizan prácticas de control y monitoreo en los laboratorios de ingeniería. En el caso del laboratorio de Control de la carrera de Electricidad, en la Universidad Técnica del Norte, se encuentran equipos y módulos didácticos de plantas y procesos industriales pequeños, que necesitan de sistema de monitoreo con interfaz de comunicación para PLC’s, y se ve la necesidad de implementar un sistema a partir de aplicaciones de Código abierto de bajo costo que reemplacen los sistemas costosos y complejos.

A.3. FORMULACIÓN DEL PROBLEMA

¿Se puede implementar un sistema HMI de bajo costo, mediante la utilización de aplicaciones de Código abierto para el control y monitoreo de un sistema dinámico real?

A.4. OBJETIVOS

OBJETIVO GENERAL

Implementar un sistema HMI mediante aplicaciones de Código abierto para el control y monitoreo de un sistema dinámico real.

OBJETIVOS ESPECIFICOS

Realizar un análisis de tecnologías para aplicaciones basados en Código abierto.

Diseñar un sistema HMI para el control y monitoreo de un sistema dinámico real en un dispositivo de Código abierto.

Evaluar el funcionamiento del sistema HMI con una red de comunicación industrial entre el dispositivo Código abierto y un Controlador Lógico Programable PLC.

A.5. JUSTIFICACIÓN

En los últimos años, los sistemas HMI han constituido un avance de gran importancia en la automatización industrial (Pérez, 2015), debido a que es un elemento fundamental de los sistemas SCADA que han adquirido las múltiples empresas en sus líneas de producción, pues permite la interacción Hombre-maquina y cada vez es necesario que esto se realice de forma gráfica y de fácil interpretación. Por tales razones este sistema es utilizado por las empresas para monitorear y controlar sus procesos de manera segura y eficiente.

En el mercado internacional existen fabricantes como Siemens, Schneider Electric ó Kinco que fabrican paneles HMI táctiles, que son complejos y de costo elevado.

Debido a la importancia de estos sistemas, la implementación de un sistema HMI basado en Código abierto, a nivel de laboratorio permitirá realizar el control y monitoreo de equipos y módulos didácticos que representan procesos industriales de bajo costo. Además, servirá como herramienta para que los estudiantes de ingeniería mejoren sus conocimientos prácticos.

A.6. ALCANCE

El presente trabajo, tiene como finalidad la implementación de un sistema HMI utilizando dispositivos que cuentan con licencia de Código abierto, y un software especializado para el diseño de la interfaz gráfica, que pueda ser visualizado mediante una pantalla de forma local, es decir, en un módulo de control junto a la planta o sistema a controlar. A su vez la comunicación entre el PLC y el dispositivo de Código abierto requerirá del uso de algún protocolo industrial.

Este sistema HMI servirá para controlar y monitorear módulos didácticos dentro del laboratorio de control, de la carrera de Electricidad de la universidad Técnica del Norte. Por tanto, el alcance del trabajo es práctico a nivel de estudio superior y tecnológico con relación al área industrial.

CAPITULO 1

Automatización y tecnologías basado en código abierto

Este capítulo proporciona una serie de información de las diferentes tecnologías que se utilizan en la automatización de procesos en las industrias actuales, para analizar los sistemas HMI en el entorno de supervisión y control de sistemas o procesos industriales. Partiendo de la premisa de que “el desafío más importante al cual se están enfrentando los empresarios, es alcanzar mayores niveles de competitividad en un entorno cada vez más industrializado”, las tendencias de tecnología actuales llevan a las empresas a adquirir este tipo de sistemas (Medina , Castro, & Camargo , 2015).

1.1. Automatización y control industrial

Los sistemas automatizados se encuentran en diferentes grados en la mayoría de los procesos industriales, haciendo posible la transferencia de trabajo del hombre hacia las máquinas, donde el trabajo del operador se limita al monitoreo y mantenimiento de la instalación. Un sistema automatizado por lo general cuenta con sensores, actuadores, equipos de protección y la implementación de software y hardware adicional (Linares Gonzales, 2015).

Estos sistemas comprenden aplicaciones directas e indirectas que se detalla en la Figura 1.

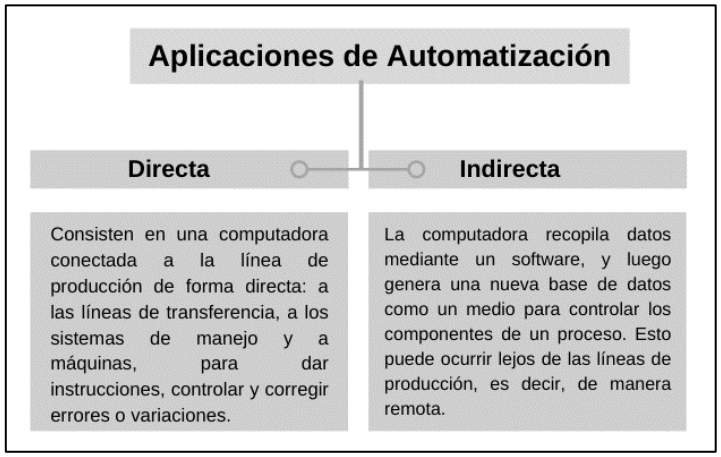


Figura 1. Aplicaciones directas e indirectas en automatización

Fuente: (Benavides, 2015)

1.2. Pirámide de automatización

(Medina , Castro, & Camargo , 2015) Plantea a la pirámide de automatización, como niveles que se comunican entre distintas familias de tecnologías para integrar la automatización de procesos como un sistema innovador. La TABLA 1, indica las características de los cinco niveles de automatización.

TABLA 1. CARACTERÍSTICAS DE NIVELES DE AUTOMATIZACIÓN

Nivel	Características
1	- Nivel de medición y control de campo - Se establecen los componentes de adquisición de datos de campo (sensores, actuadores). Controla a las máquinas y equipos de producción.
2	- Agrupa a todos controladores locales tales como: ordenadores, PLC's, etc.
3	- Formado por estaciones de operaciones o servidores de ingeniería como SCADA. Controla la secuencia de producción y/o fabricación.
4	- Nivel de gestión de la empresa, comunica distintas plantas, - Mantiene relaciones con los proveedores y clientes.
5	- Nivel de gestión administrativo y planificación mediante base de datos y sistemas informáticos.

Fuente: (Medina , Castro, & Camargo , 2015)

En cada nivel de automatización se reconocen elementos o dispositivos que interactúan con otros dispositivos de distintos niveles. La figura 2, muestra una representación gráfica de la comunicación en una industria automatizada con todos los requerimientos para una producción de calidad.

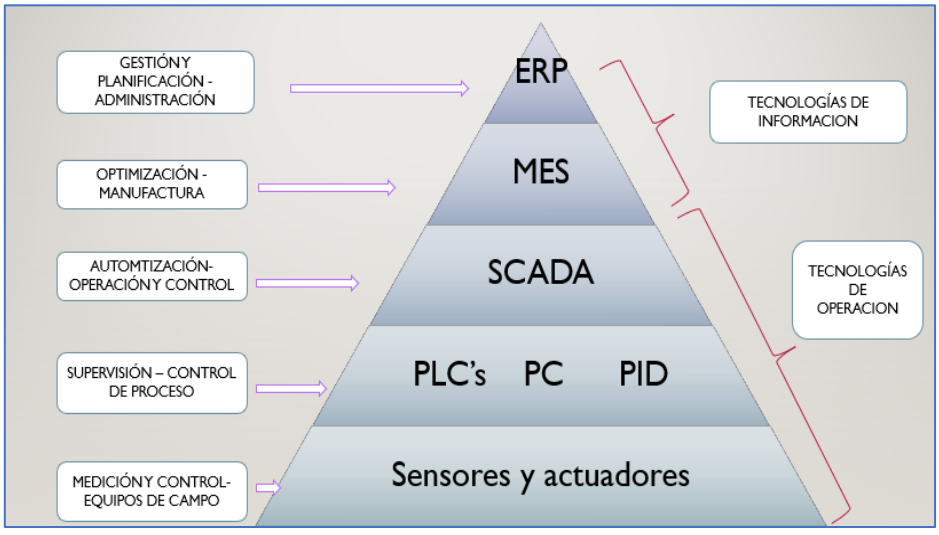


Figura 2. Pirámide de automatización

Fuente: Editado por el autor de (Medina , Castro, & Camargo , 2015)

Cada nivel de tecnología es capaz de comunicarse entre sí para mejorar los resultados en los distintos procesos industriales.

1.3. Importancia de sistemas en tiempo real

‘‘Un sistema en tiempo real ayuda a que las organizaciones transformen los procesos en una solución o en un producto innovador’’ (Behnam N. , 2009). Hoy más que nunca las empresas y compañías industriales intentan adoptar los sistemas de control en tiempo real como una herramienta poderosa de negocios, para mejorar la ventaja competitiva y aumentar la productividad, ya que los grandes avances en la tecnología hacen posible esta perspectiva.

1.4. Interfaz Hombre Máquina (HMI)

(Saquicuya, 2019) mencionan que una HMI es el medio de interacción entre un operario y un determinado hardware y/o máquina, que muestra información de un sistema de control, tales como variable de proceso, variable de control y set point. Así mismo señala que la Interfaz Hombre Máquina permite la comunicación mediante la transmisión de información, ordenes, y datos de manera bidireccional. La Figura 3. muestra al HMI como un elemento terminal importante donde hace interaccin hombre- máquina en los Sistemas SCADA.

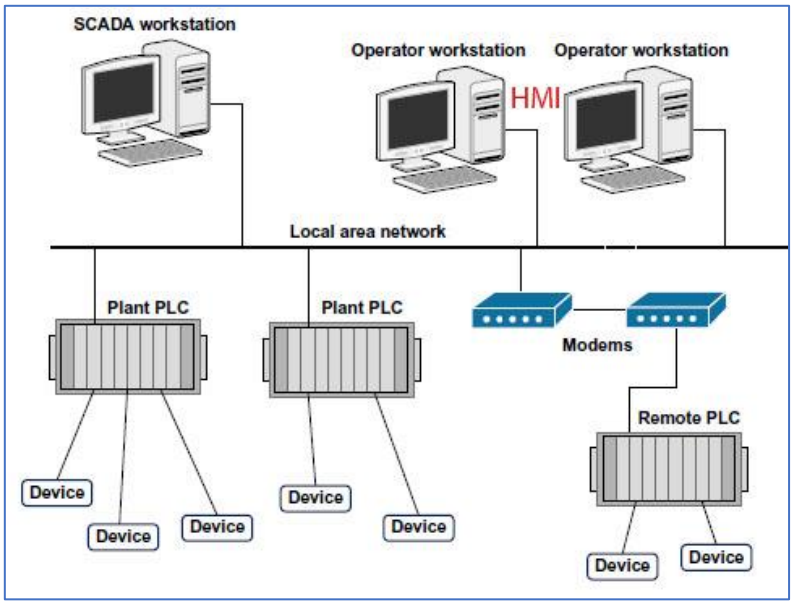


Figura 3. HMI como elemento en la Arquitectura típica de un Sistema SCADA.

Fuente: (Maccrady, 2013)

El aporte de una buena interfaz está basado en el control de las anomalías o problemas que se presentan en tiempo real, es decir, durante el tiempo que se producen las diferentes variaciones, mediante diversos ajustes de parámetros para el control de estos procesos.

1.4.1. HMI en un sistema de control

Los dispositivos HMI y los sistemas en tiempo real forman parte del tercer nivel en la pirámide de la automatización, son un componente esencial en los sistemas de Supervisión, Control y Adquisición de Datos (SCADA). Las HMI permiten la obtención de datos visuales de cualquier proceso permitiendo una interacción eficiente con el operador. Las funciones más importantes de una HMI son; la monitorización y muestreo de datos de la planta en tiempo real, la supervisión de trabajo del proceso mediante una computadora, el reconocimiento de errores o eventos excepcionales mediante alarmas y el control del proceso para mantener las variables o valores dentro de ciertos límites (Saquicuya, 2019).

La figura 4, muestra un sistema de control típico que consta de sensores que son elementos primarios que permiten conocer el estado de la planta o proceso, esta información es acondicionada y enviada al procesador como núcleo central del proceso, luego verifica la información que recibe de los sensores y es visualizada en la interfaz gráfica desarrollada mediante cualquier plataforma de diseño. En la fase final se valida la información por el operario quien toma una decisión y es enviada al procesador y finalmente reflejarse en los actuadores (Antúnez Soria, 2016).

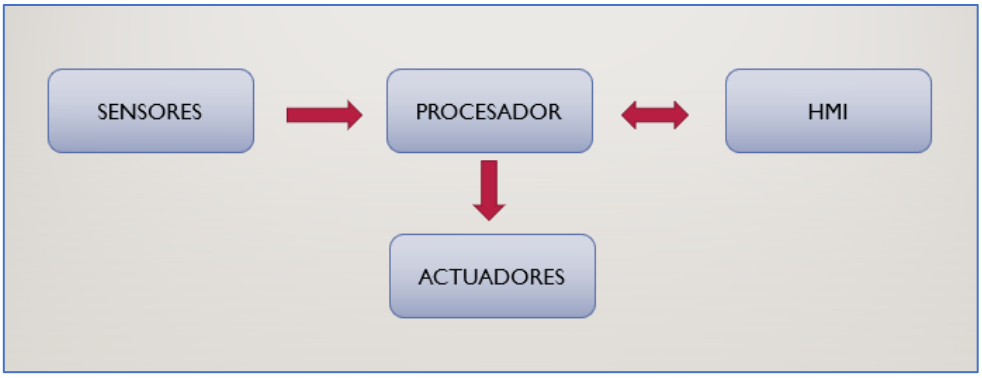


Figura 4. Fases de un sistema de control con HMI

Fuente: (Dominguez, Vargas, & Velásquez, 2015), (Antúnez Soria, 2016)

1.4.2. Tipos de HMI

A nivel industrial existen una variedad de sistemas HMI. La clasificación más representativa según (Antúnez Soria, 2016) se muestra en la TABLA 2.

TABLA 2. TIPOS DE INTERFAZ HOMBRE MÁQUINA

Tipo	Características
Interfaz basado en hardware	Se trata de un conjunto de controles o dispositivos que permiten la interacción hombre- máquina, de modo que permiten leer datos del sistema o equipo mediante pulsadores, reguladores e instrumentos.
Interfaz basado en texto	Presenta la información en formato texto sobre una pantalla de cristal líquido (LCD) o grupos de led de segmentos. Estos componentes corresponden al display del dispositivo. Cuando el display es de tipo LCD, este suele ser de tipo alfanumérico, y está definido por: número de caracteres que muestran y número de líneas.
Interfaz basado en software	Son programas o parte de ellos que permiten expresar sentencias en un ordenador para la observación de la respuesta del sistema.
Interfaz gráfica de usuario GUI	GUI (Graphic User, Interface), son todos los elementos gráficos o visuales que nos ayudan a comunicarnos con un sistema. También se le conoce como el medio de interacción entre el hombre y la máquina.
Interfaz basado en gráfico	Basados en pantallas de tecnología LCD (Pantalla de Cristal Líquido) que presenta la información en formato gráfico. Actualmente se emplean las de tipo STN y TFT. El tamaño de la pantalla viene definido por la diagonal de esta y la calidad la tecnología empleada.
Interfaz basado en panel táctil	Están representadas gráficamente en un panel de control, en una pantalla sensible que permite interactuar de manera táctil para el control del proceso.

Fuente: (Antúnez Soria, 2016)

1.5. Funciones de los sistemas HMI

Las funciones básicas que deben cumplir los sistemas de control de procesos se explican mediante la TABLA 3.

TABLA 3. FUNCIONES DE LOS SISTEMAS HMI

Funciones	Descripción
Monitorización	Visualización del estado del proceso. Obtención y muestreo de datos de la planta en tiempo real. Estos datos se pueden mostrar como números, texto o gráficos que permitan una lectura más fácil de interpretar.
Supervisión	Esta función permite junto con la monitorización la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora.
Alarmas	Capacidad para reconocer eventos excepcionales dentro del proceso y reportar eventos. Entrega de mensajes de alarma.
Control	Modifica parámetros del sistema, es decir forzar variables de salida. Ajustar valores del proceso y así mantener estos valores dentro de ciertos límites.

Fuente: (SENA, 2016), (Rodriguez, 2013), (Linares Gonzales, 2015)

Las funciones básicas mencionadas en la Tabla 3, se deben cumplir para que las interfaces permitan la interacción del operario con la máquina de una manera eficiente. Además, para que una Interfaz Hombre Máquina sea significativo debe adaptarse a los requisitos que indica mediante la Figura 5.

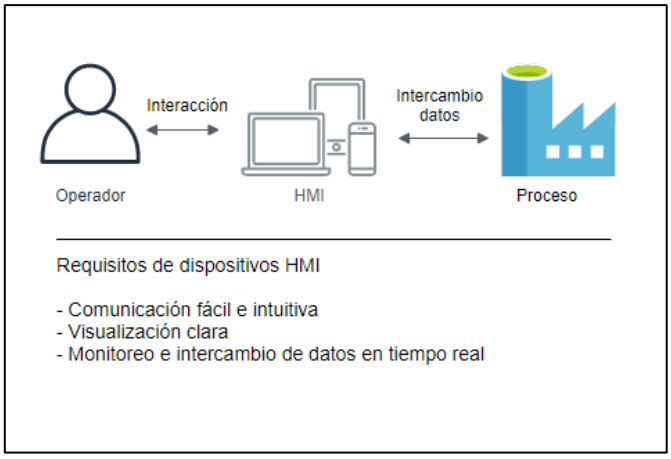


Figura 5. Requisitos primordiales de un Sistema HMI para una buena comunicación entre usuario y máquina.
Fuente: (Imbaquingo , Granda , Ortega, Guevara, & Pusdá, 2016) pp-33

1.6. Tecnologías en el ámbito de las interfaces hombre máquina- HMI

A partir de la llegada de la segunda generación de computadoras, eficientes y con menos consumo de electricidad gracias a la invención del transistor, se ha generado la necesidad de desarrollar interfaces que permitan comunicar al hombre con la máquina de una manera eficiente. Las nuevas tecnologías han permitido un gran avance en las HMI a través del desarrollo de hardware y software que facilita la comunicación con las computadoras de manera innovadora. Estas nuevas tecnologías han venido creciendo con un bajo costo en el mercado, permitiendo así una fácil adquisición y generando un gran impacto en el desarrollo de aplicaciones open Source en temas como realidad virtual, Mocap (Motion Capture) e incluso en BCIs (Brain Computer Interface).

1.7. Normas en la integración de personas con los sistemas

Las normas facilitan el trabajo colaborativo ya que representan soluciones comunes a problemas específicos. Según (AENOR, 2016) “Son la referencia para que fabricantes, proveedores, usuarios, reguladores, etc. sepan qué pueden esperar de un producto o servicio”.

Una de las normas más relevantes es los estándares ISO (Organización Internacional de Estandarización) son establecidas para establecer estándares en la calidad de sistemas en un cualquier tipo de organización

1.7.1. Normas ISO serie 9000

Es la serie más popular de la Organización Internacional de Normalización. “Se evidencia el énfasis en aspectos relacionados con la gestión de la información, el conocimiento, la innovación y el aprendizaje organizacional” (Habana, 2016).

1.7.2. Norma ISO 9241- 110

La norma internacional (UNE-EN ISO 9241-110, 2006) establece los principios de diseño y aplicación del entorno tecnológico, para diseñar, analizar y evaluar los sistemas interactivos, al mismo tiempo se preocupa de la comunicación o interacción entre el usuario y un sistema dinámico. Los lineamientos se han destinado para los siguientes tipos de usuarios:

a) Diseñadores y desarrolladores de interfaces interactivas
b) Personas que apliquen la guía para el diseño y funcionamiento del sistema

c) Compradores: Condiciones de compra de dispositivos según la norma ISO 9241
d) Evaluadores: Los responsables de asegurar que los productos para el desarrollo de estos sistemas son recomendables.

1.8. Tecnologías de código abierto

La base para la innovación tecnológica es estar a la vanguardia en los procesos y servicios, y la mejor manera es que las organizaciones estén atentas a los cambios y evolución del desarrollo tecnológico a través de un proceso de vigilancia continua de dicha tecnología (Medina , Castro, & Camargo , 2015).

El vínculo entre el control de procesos y el desarrollo de la tecnología se ha fortalecido, permitiendo la evolución de los sistemas de control desde básicos sistemas mecánicos, hasta modernos sistemas automáticos que toman decisiones ante determinados comportamientos de un proceso.

Existen millones de dispositivos móviles funcionando en el mundo, y la necesidad de satisfacer las exigencias de los usuarios ha generado la aparición de los llamados dispositivos móviles inteligentes o Smartphone. Estos equipos, permiten junto con el sistema operativo desarrollar aplicaciones basadas en tecnologías como Bluetooth, el protocolo Universal del Bus Serie (Universal Serial Bus, USB), Wi-Fi, entre otros, que facilitan la interacción entre dispositivos electrónicos para el intercambio de información y, que incorporan a la instrumentación virtual tradicional, el atributo de movilidad (Medina , Castro, & Camargo , 2015).

La mayoría de las microempresas; desarrollan propuestas fundamentadas en el uso de tecnologías abiertas lo cuales se muestran en la TABLA 4, que brinden la oportunidad de modernizar tecnológicamente.

TABLA 4. PROYECTOS DESARROLLADOS CON TECNOLOGÍA Y SOFTWARE LIBRE O DE CÓDIGO ABIERTO

País	Proyectos desarrollados con software libre
Colombia	Incorporación de tecnologías para procesos industriales
España	Sistema de embebidos con software libre
Cuba	Supervisión y control de procesos industrial con dispositivos móviles.
Perú	Desarrollo de aplicaciones de software SCADA, para control centralizado mediante HMI, de procesos industriales

Fuente: (Medina , Castro, & Camargo , 2015)

1.9. Desarrolladores de software

Son tecnologías informáticas, que permiten la integración de sistemas de adquisición y procesamientos de información. La ingeniería del software posee técnicas y herramientas que han madurado mucho actualmente se invierten grandes cantidades de recursos para potenciar la industria del software y convertirla en uno de los sectores estratégicos de crecimiento (Rozo, 2014).

(Benavides, 2015) Considera que en el mercado se puede encontrar varios desarrolladores y software de acuerdo con la necesidad del proceso a controlar, existen los que permiten ver el proceso e interactuar con él, y los permiten ver alarmas en tiempo real y registrar históricos. Las herramientas de diseño que sirven para configurar alguna aplicación y la herramienta de ejecución que sirve para simular procesos. Existen varios fabricantes de desarrolladores de software HMI, los más destacados son:

National Instruments (LabView).

Wonderware (InTouch).

WinCC del software Tia Portal

1.9.1. National Instruments (LabView)

LabVIEW es software de ingeniería que ayuda a crear aplicaciones mediante la programación gráfica. En sus inicios se creó para realizar aplicaciones para el control de instrumentos electrónicos usadas en el desarrollo de sistemas de instrumentación, lo que se conoce como instrumentación virtual. (Vaca, 2019).

LabVIEW, además, ofrecen facilidades en el desarrollo de la interfaz de usuario como las funciones de alto nivel y la incorporación de elementos gráficos, que simplifican la tarea de programación y de elaboración de una HMI o panel frontal (Laime & Almidón, 2018)

la creación de un instrumento virtual VI se logra a partir del desarrollo de diagrama de bloques como se muestra en la figura 6.

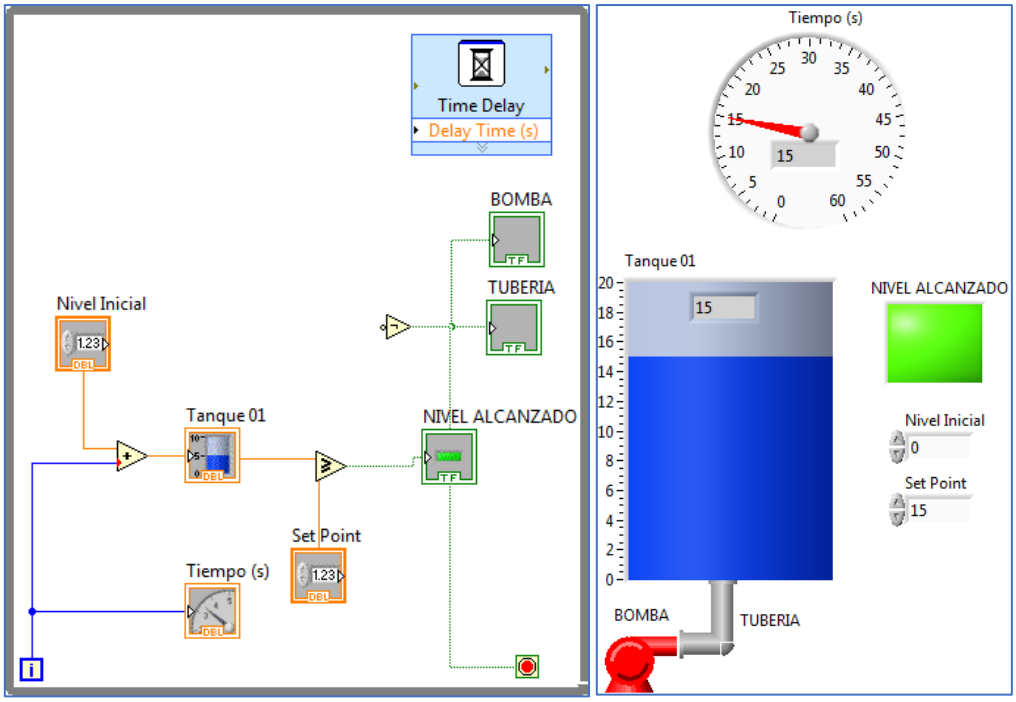


Figura 6. Control de nivel de un tanque con LabVIEW: a) representación en diagrama de bloques b) Planta frontal gráfico

Fuente: (Laime & Almidón, 2018)

El monitoreo del sistema de control en lazo cerrado, usando LabVIEW, está relacionado con el modelo de control en tiempo real.

1.9.2. Wonderware (InTouch)

InTocuh es un software de aplicación para el desarrollo de sistemas HMI utilizadas para la automatización industrial, control y monitoreo de procesos secuenciales, basado en programación G (de tipo gráfico). Este potente software puede actualizarse paralelamente a las tendencias y necesidades del usuario (Benavides, 2015).

InTocuh cuenta con dos elementos importantes para el desarrollo de interfaz, las características se detallan en la TABLA 5:

Window Viewer

Window Maker

TABLA 5. ELEMENTOS DE INTOUCH PARA EL DESARROLLO DE INTERFAZ

Window Maker	Window Viewer
- Herramienta de dibujo basado en gráfica por objetos. - Tiene un sistema de desarrollo, para crear ventanas animadas interactivas. - Permite el diseño de la HMI - Permite la conectividad con los dispositivos externos como PLC.	- Sistema que permite ejecutar las aplicaciones creadas con WINDOW MAKER

Fuente: (Benavides, 2015) & (Vaca, 2019)

La figura 7, muestra los componentes para crear aplicaciones InTouch HMI independientes, que consisten en Administrador de aplicaciones, WindowMaker y WindowViewer.

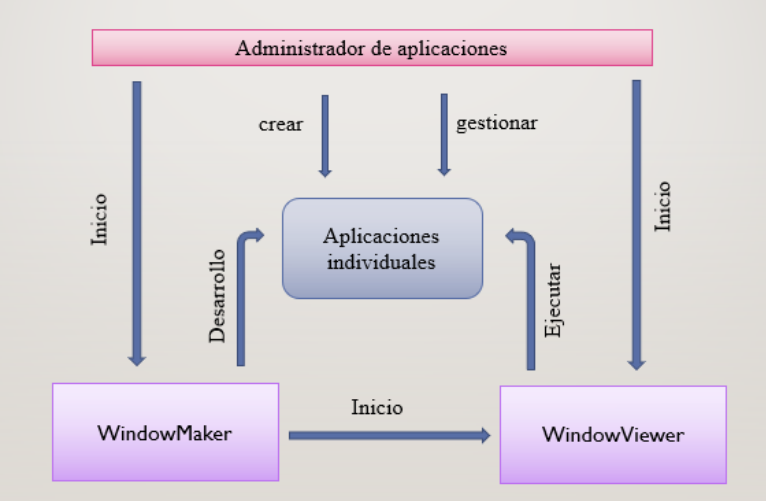


Figura 7. Elementos de la plataforma InTouch (AVEVA, 2018)

Se pude crear y administrar aplicaciones independientes con InTouch, Gestor de aplicaciones. Desarrollar aplicaciones independientes con WindowMaker y ejecutar desde WindowViewer. Además, se puede cambiar directamente entre WindowMaker y WindowViewer para probar o ejecutar sus aplicaciones y volver a hacer (AVEVA, 2018).

1.10. Raspberry PI

Es una computadora de placa única, existente en cuatro versiones: Modelo A, Modelo B rev 1, Modelo B rev 2 y Modelo B+. Cada una de estas versiones tienen características ligeramente diferentes. El sistema operativo recomendado es Raspbian (Derivación de Linux). Al igual que en una PC, la Raspberry permite desarrollar, instalar y ejecutar software GNU y con su tamaño reducido y peso ligero lo hace ideal para proyectos de sistemas automatizados (Salcedo, 2016).

La Figura 8, muestra un esquema general de las partes de una Raspberry Pi convencional.

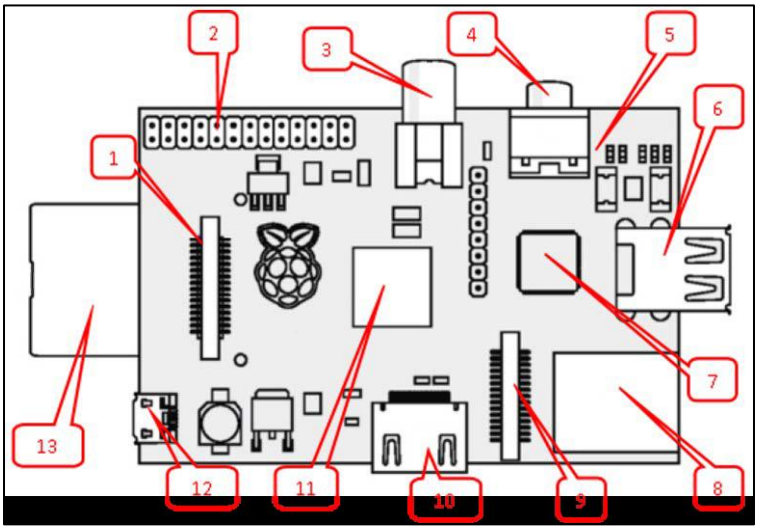


Figura 8. Esquema del Raspberry Pi Modelo B

Fuente: (Salcedo, 2016)

Los elementos más sobresalientes de la Figura 8, para realizar una interfaz se enlista en la TABLA 6, a continuación:

TABLA 6. PARTES DE LA RASPBERRY PI USADOS PARA INTERFAZ.

Elemento	Nombre	Descripción
1	Conector DSI	Conexión para pantalla LCD
2	Cabezal de expansión	26 pines, cabezal para alimentación (5V, 3,3 V y GND) Conexión de puertos GPIO del SoC al exterior configurables como (PWM, Entrada, Salida, Latch)
3	Puerto de video	Conexión tipo RCA para televisor analógico
6	Puertos USB	Puerto transmisión 2.0

7	Controlador Eterneth	Microchip
8	Puerto ethernet 10/100	Para puerto de comunicación con RJ45
9	Conector csi-2	Conector para cámara
10	HDMI	Conector HDMI conexión de monitores de alta definición.
11	System on a chip	Procesador de gráficos y señales
12	Micro USB	alimentación del dispositivo (+5 VDC, ±4 %)

Fuente: (Salcedo, 2016)

La ventaja de usar la Raspberry Pi es por sus varias entrada y salidas que permiten la conexión con otros dispositivos de manera sencilla, y a diferencia de una Laptop o PC tiene consumo relativamente bajo, alrededor de 3,5 Vatios a carga máxima.

1.11. Protocolos de comunicación

Es un sistema que tiene la capacidad de conectar y transmitir datos o información entre distintos dispositivos de un sistema. Además, permite la operación con dispositivos de campo máquinas y equipos industriales utilizados en procesos productivos (Vaca, 2019).

Los protocolos de comunicación más ampliamente utilizados en la industria son:

Profibus

Hart

Modbus

Ethernet

OPC (Open Platform Communications)

A continuación, se explica brevemente mediante la tabla 7.

TABLA 7. CARACTERÍSTICAS DE PROTOCOLOS DE COMUNICACIÓN MÁS USADOS EN LA INDUSTRIA

Protocolo de Comunicación	**Descripción**
Profibus	• Sistema libre e independiente de comunicación rápida, maneja la comunicación serie para conectar dispositivos de campo entre sí. • **Profibus DP (Periferia Descentralizada):** Orientado a comunicación entre Sensores/actuador y PLC´s • **Profibus PA (Process Automation):** Para el control de procesos, cumple con normas de seguridad (IEC- 1 1158-2, seguridad intrínseca).

	• **Profibus FMS (Fieldbus Message Specification):** Para comunicación entre equipos de automatización.
Hart	• Comunicación analógica 4-20mA o digital sobre un mismo cable permitiéndonos utilizar dos canales de forma simultánea. • Realiza una configuración remota que permita ajustes de parámetros y diagnóstico de dispositivos de campo inteligentes. • **Características:** Una Maestra puede controlar hasta 15 Remotas Permite hasta 250 variables en cada dispositivo de campo Distancia máxima: hasta 3000 m con par trenzado apantallado calibre AWG24; Medio de transmisión: par trenzado y el lazo de corriente de 4-20 mA. Interfaces asociadas: RS-232D y RS-485.
Modbus	• Desarrolaldo por Gould Modicon ahora Schneider Automation con el fin de realizar sistemas capaces de controlar y supervisar de procesos industriales (SCADA) basado en un control centralizado. • Este protocolo está compuesto por, una Estación Maestra (MTU) la cual puede comunicarse con una o varias Estaciones Remotas (RTU) con el propósito de obtener datos de campo para la supervisión y control de un proceso.
Ethernet	• Permite la interconexión de redes industriales de manera simple. • Aprovecha los medios físicos y los chips de comunicaciones Ethernet comerciales. • Este protocolo esta estandarizado en la norma internacional IEEE 802.3 y IEC 61158
OPC (Open Platform Communications)	• La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. Donde el servidor OPC es la fuente de datos (como un dispositivo hardware a nivel de planta) y cualquier aplicación basada en OPC, puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor.

Fuente: (Vaca, 2019), (Terán , 2015)

1.12. Lenguajes de programación

Los avances de la computación hacen posible, escribir programas para cualquier aparato presente en grandes industrial, automóviles, aparatos electrodomésticos, cajeros automáticos, teléfonos inteligentes, etc. Existen un sin número de procesadores físicos y

cada uno utiliza un lenguaje máquina (lenguaje que contiene unos y ceros) distinto para comprender un programa (Gutiérrez, 2015).

En términos generales un programa informático es un archivo con un conjunto de instrucciones que pude estar escrito en diferentes lenguajes de programación actuado sobre datos, sobre otros programas o sobre dispositivos físicos. Estos lenguajes han sido creados para facilitar la programación de códigos en un lenguaje más entendible por el programador (Gutiérrez, 2015).

Según el índice de popularidad de lenguajes de programación PYPL analizado por la frecuencia de búsqueda en Google Trends, se tienen los lenguajes de programación más utilizados en la actualidad en la TABLA 8.

TABLA 8. EL ÍNDICE DE POPULARIDAD DE LENGUAJES DE PROGRAMACIÓN (PYPL)

Rango	**Idioma**	**Cuota**	**Tendencia**
1	Python	29,9%	+1,2%
2	Java	17,72%	+0,0%
3	JavaScript	8,31%	+0,4%
4	C#	6,9%	-0,1%
5	C / C ++	6,62%	+0,9%
6	PHP	6,15%	+0,1%
7	R	3,93%	+0,0%
8	Mecanografiado	1,89%	+0,0%
9	Matlab	1,71%	-0,2%

(Carbonnelle , 2020)

La popularidad de los lenguajes de programación es debido que la mayoría de las aplicaciones, software y páginas web son diseñados en base a programación. Es así, que la tendencia de uso para crear aplicaciones va en aumento.

1.13. Editores de texto

Los editores de texto son programas que facilitan la escritura de diferentes lenguajes de programación, que verifican la sintaxis, orden y escritura del código.

El índice de popularidad de lenguajes de programación PYPL analizado por la frecuencia de búsqueda en Google Trends, también clasifica los más utilizados por los programadores y se detalla mediante la TABLA 9.

TABLA 9. ÍNDICE EDITORES DE TEXTO

Rank	IDE	Cuota	Tendencia
1	Visual Studio	27.88 %	+3.0 %
2	Eclipse	15.01 %	-2.6 %
3	Visual Studio Code	10.79 %	+3.7 %
4	Android Studio	9.7 %	-4.7 %
5	pyCharm	7.71 %	+1.3 %
6	IntelliJ	6.17 %	+0.6 %
7	NetBeans	5.52 %	-0.1 %
8	Sublime Text	3.71 %	-0.2 %
9	Xcode	3.51 %	-0.9 %
10	Xcode	3.51 %	-0.0 %
11	Atom	3.43 %	+0.6 %

Fuente: (Carbonnelle , 2020)

En el siguiente capítulo se utilizará el editor Visual Studio Code como editor de texto para el código Python por ser un en editor de código abierto liviano y gratuito en comparación al entorno de desarrollo integrado (IDE) Visual Studio que presenta versiones pagadas. Además, por la facilidad de integración de extensiones para un lenguaje de programación específico.

CAPÍTULO 2

Diseño del sistema Interfaz Hombre Máquina (HMI)

En este capítulo se describe el diseño de la interfaz gráfica, selección y revisión de las características de los dispositivos de código abierto o “hardware libre”, especificaciones de protocolos de comunicación con otros dispositivos como el PLC y características del software libre para el diseño de la interfaz gráfica en Qt Designer mediante programación Python. De tal manera que la comunicación entre los distintos dispositivos permita el control y monitoreo de un sistema de presión y temperatura de agua en lazo abierto dentro del Laboratorio de Control, en la Carrera de Electricidad.

2.1. Metodología para el diseño del sistema de Interfaz Hombre Máquina

El desarrollo de este capítulo se realiza mediante los siguientes pasos:

- Selección del Módulo didáctico adecuado con variables para efectuar el control y monitoreo de sus variables mediante la implementación del sistema HMI.
- Selección de hardware de código abierto Raspberry Pi y Pantalla táctil LCD, a partir de un análisis de tecnologías de uso frecuente.
- Selección del software Visual Studio para programación en Python mediante PYQt5 y Qt Designer para el diseño de la interfaz gráfica.
- Comunicación Modbus TCP/IP entre Raspberry Pi 3 y PLC Siemens S7-1200
- Comunicación Modbus RS-485 entre PLC Siemens S7-1200 y Variador Altivar 312.
- Programación en Python para el control de variables.
- Desarrollo de la interfaz gráfica, mediante programación orientada a objetos en Python.

2.2. Criterios de diseño

El diseño del sistema HMI se realiza según los requerimientos de funcionalidad y compatibilidad entre distintos dispositivos, y tomando en cuenta los siguientes criterios:

- **Aspecto físico:** Debe tener apariencia similar a Dispositivos HMI convencionales.
- **Disponibilidad y accesibilidad**: Los dispositivos de código abierto deben estar disponibles en el mercado.

- **Costo**: El dispositivo HMI basado en código abierto debe ser más económico con respecto a tecnologías industriales como los Simatic HMI de Siemens o los Páneles de control de Schneider Electric.
- **Facilidad de uso:** La interfaz gráfica debe ser intuitiva y de fácil interpretación de los datos que se obtiene del proceso en tiempo real, con una respuesta rápida.
- **Funcionalidad:** Debe manejar protocolos de comunicación compatibles entre el dispositivo de código abierto y el PLC Siemens S7-1200.
- **Tamaño**: El tamaño reducido debe permitir una instalación simple en el tablero de control.

Estos parámetros deben tomarse en cuenta antes, durante y después del proceso de diseño e ir acorde a las necesidades visuales y experiencia del usuario ya que siempre están involucrados en los sistemas de control (Acosta, 2017).

2.3. Selección de equipos

La selección de los equipos para el sistema HMI se realiza tomando en cuenta la propuesta del proyecto, y considerando dispositivos que se encuentran en tendencia para la elaboración de proyectos de experimentación, a continuación, se detalla cada uno.

2.3.1. Módulo didáctico

Las instalaciones de la carrera de Electricidad cuentan con varios laboratorios entre ella el de Control de Máquinas. Dentro de este laboratorio se encuentra el Módulo de control de presión y temperatura de agua que se muestra en la Figura 9, el cual es ideal para la implementación de un sistema HMI compacto y reemplazar por el ordenador que ocupa un gran espacio.

Este módulo cuenta con sensores que lo hacen ideal para una práctica de monitoreo de variables mediante el dispositivo HMI. También cuenta con un Variador de Frecuencia Altivar 312 para una práctica de comunicación industrial entre PLC y Variador.

Figura 9. Módulo de control de presión y temperatura de agua, Laboratorio Control
Fuente: Autor

Además, se ha considerado los elementos instalados como el PLC Siemens S7-1200 con capacidad de comunicación mediante el protocolo Modbus TCP/IP con un microcomputador Raspberry Pi.

Otros elementos importantes instalados en el módulo didáctico se detallan en la TABLA 10.

TABLA 10. ELEMENTOS INSTALADOS EN EL MÓDULO DE CONTROL DE PRESIÓN Y TEMPERATURA

	Cantidad	Elementos	Características
Control	1	PLC	Siemens S7-1200 1212C AC/DC Relay, CPU 3.1
	1	Variador de frecuencia	Altivar 312, Potencia 1,5 KW
Proceso	1	Niquelina	Alambre de 8 ohmios
	3	Electroválvula	Entrada 12 V DC
	1	Electrobomba	Entrada 220 V, velocidad 3400 rpm, 60 Hz
Realimentación	1	Sensor de presión	Salida de 4-20 mA
	1	Sensor de temperatura	Salida de 4-20 mA

Fuente: (Matango & Portilla, 2016)

Para conocer a detalle los circuitos de conexión en el tablero de control es necesario revisar en el trabajo de investigación correspondiente a (Matango & Portilla, 2016).

2.3.2. Controlador Raspberry pi

Mediante una investigación sobre controladores de código abierto se ha seleccionado la Raspberry Pi 3B plus, que se muestra en la Figura 10. Debido a sus ventajas frente a versiones anteriores como Raspberry pi modelo 2A, 2B ó 3A y en comparación a otros microcontroladores como Arduino.

Figura 10. Raspberry pi 3 modelo B plus
Fuente: (Raspberrypi.org, 2017)

Este dispositivo que es un miniordenador cuenta con periféricos de salida que permiten la conexión de pantallas o monitores mediante conector HDMI. Además, permite la instalación de un sistema operativo basado en Linux, que permite el desarrollo de interfaces gráficas de usuario (GUI) mediante programación. Para este proyecto se realiza el diseño de la interfaz mediante programación en lenguaje Python.

Cada versión de Raspberry cuenta con ciertas características que lo identifican. La TABLA 11, muestra las especificaciones de la versión seleccionada.

TABLA 11. CARACTERÍSTICAS TÉCNICAS DE RASPBERRY PI 3B PLUS

Raspberry Pi 3 B plus	
Procesador	64 bits a 1.4GHz IEEE 802.11. b

Conectividad	LAN inalámbrica de 5 GHz. Bluetooth 4.2/BLE. Gigabit Ethernet hasta 300Mbps
Entradas USB	4 × USB 2.0
Memoria	1 GB RAM
Video y Sonido	1 entrada HDMI de tamaño completo
Tarjeta SD	Formato microSD expandible para almacenamiento de datos.
GPIO	Cabecera de 40 pines
Características eléctricas	
Alimentación	5V/2.5 A DC con conector micro USB
Temperatura	Operación de 0 a 50 °C
Tamaño	85 mm x 49 mm

Fuente: (Raspberrypi.org, 2017)

En el ANEXO A: Características de Raspberry Pi 3 Modelo B+, se adjunta las características, especificaciones eléctricas y físicas completas.

2.3.3. Pantalla de visualización

Los dispositivos de manejo y visualización, de las marcas reconocidas como Siemens o Schneider Electric comprenden paneles con pantalla de 4, 7, 9 y 12 pulgadas. La variedad de tamaño es debido a que la complejidad del proceso varía de una instalación a otra. Para el proyecto se ha seleccionado un Display HDMI de 5 pulgada para conexión directa con la Raspberry. La pantalla se puede visualizar en la Figura 11. Además, es compatible con la mayoría de las versiones de Raspberry, excepto Raspberry pi Zero.

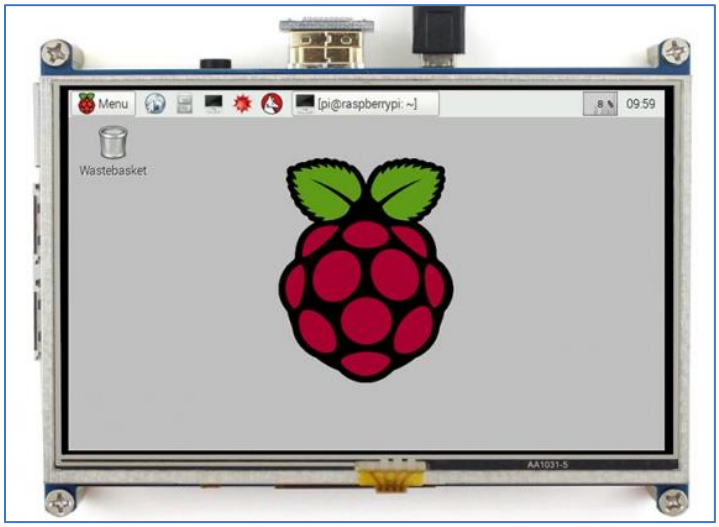


Figura 11. 5inch HDMI Display

Fuente: (Saveshare, 2020)

Las principales características de la pantalla seleccionada se muestran en la Tabla 12:

TABLA 12. CARACTERÍSTICAS DE PANTALA TOUCH 5 PULGADAS

Características	
Tamaño	5 pulgadas
Resolución	800 x 480 pixeles
Control	Táctil resistivo
Conexión	Puerto HDMI para Raspberry Y pines GPIO
Alimentación	5V DC conector Micro USB

Fuente: (Elecrow, 2016)

En el ANEXO B: Manual de usuario de pantalla HDMI 5 pulgadas, se adjunta las características completas del dispositivo.

2.4. Software para programación y diseño de interfaz gráfica de usuario (GUI)

El conjunto de software seleccionado es de código abierto y cada software es ejecutable en distintos sistemas operativos, se utilizó un software para programación, otro para la edición del lenguaje Python y un último para desarrollo de la GUI. Estos softwares son: Python, Visula Studio Code, PyQT y QTDesigner.
A continuación, se detalla las características que ayudaron en el diseño de la GUI.

2.4.1. Python

Este lenguaje de programación de alto nivel es considerado fácil y versátil ya que su sintaxis es simple, claro y visual. El código creado por el usuario es traducido por un intérprete y se ejecuta por cada línea de comando. Es compatible en varios sistemas operativos como Windows, Linux o Mac/Os. Además, soporta programación orientada a objetos permitiendo a la vez crear programas con componentes reutilizables (Fernández & Algar, 2019) y (Cuevas , 2016).

Una de las ventajas de la selección de este lenguaje es, que el sistema operativo Raspbian trae instalado la aplicación Thonny para ejecutar directamente el código de Python, esto se puede verificar en la Figura 13 en el escritorio de Raspbian.

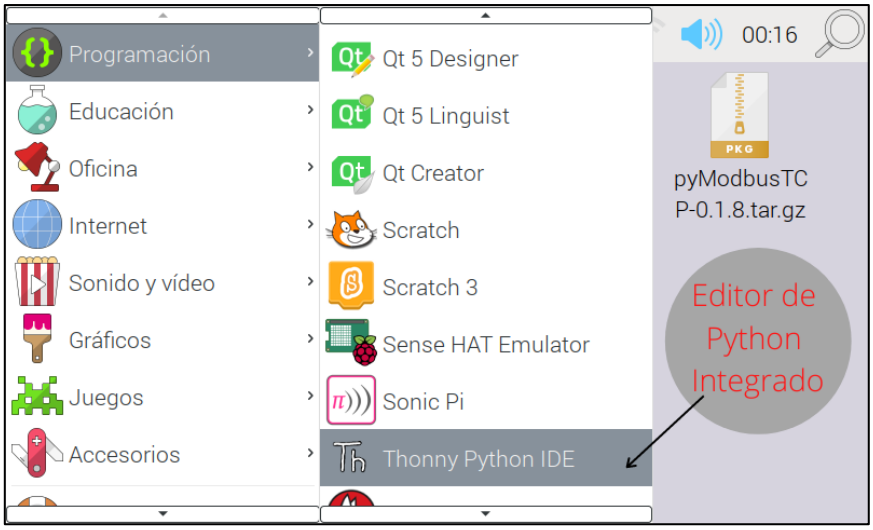


Figura 12. Ventana del sistema operativo Raspbian en Raspberry pi 3 con editor de Python Thonny

Fuente: Autor

Para sustentar la selección de Python frente a otros lenguajes de programación se toma en cuenta la Tabla 8 dentro del capítulo 1, que muestra el índice de popularidad analizado por Google Trends.

Otras ventajas de este potente lenguaje se presentan mediante la Tabla 13.

TABLA 13. VENTAJAS Y DESVENTAJAS DE PYTHON

Lenguaje de programación Python	
VENTAJAS	**DESVENTAJAS**
Software libre y gratuito	La curva de aprendizaje es muy amplia
Es un lenguaje de propósito general, interpretado y orientado a objetos.	Algunas librerías que vienen preinstalados están en desuso.
Se basa en sintaxis simple que facilita la lectura.	La ejecución de múltiples hilos utiliza más memoria RAM.
Contiene gran número de librerías	
Es multiplataforma	
El lenguaje más utilizado por los programadores actuales	

Fuente: (Cuevas , 2016), (Fernández & Algar, 2019)

2.4.2. Visual Studio Code

La selección del editor de software libre Visual Studio Code se hace tomando como referencia la tendencia de uso presentado en la Tabla 9 dentro del Capítulo 1.

Visual Studio Code es un editor de código fuente ligero muy potente compatible con los sistemas operativos Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js. Contiene extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP, Go) (code.visualstudio.com, 2021).

2.4.3. PYQT y Qt Designer

La librería Qt es una herramienta que permite la representación gráfica del código (texto en Python) mediante ventanas, botones, casillas, objetos y elementos gráficos. Esta representación se denomina GUI (Interfaz Gráfica de Usuario) el cual permite la interacción del usuario con el programa.

Pyqt es un enlace a las librerías de QT para poder utilizarlo en el código Python. En otras palabras, Pyqt sirve de enlace entre el código escrito en Python y las librerías Qt (escritas en C++) para poder generar un GUI completo.

Qt Designer es un software multiplataforma que a su vez es una herramienta de **Pyqt** que ayuda a desarrollar interfaces gráficas solo con arrastrar objetos o widgets. El entorno de trabajo se muestra en la Figura 13.

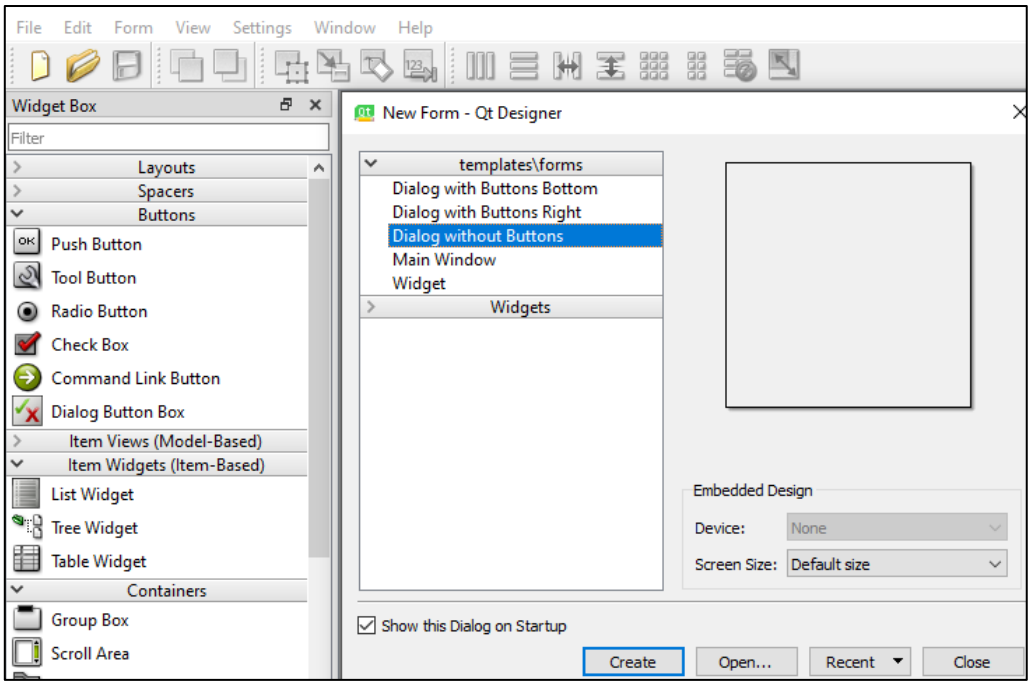


Figura 13. Entorno de trabajo de QT designer

Fuente: Autor

La interacción se logra mediante TOOLKIT QT escrito en C++ que es un entorno multiplataforma para Windows, Mac y Linux que cuenta con una colección de elementos para el control gráfico, y poder para realizar aplicaciones de tipo GUI. Esta herramienta además es comprendida como software libre a través de Qt Project (Cuevas , 2016).

2.5. Comunicación industrial

Todos los sistemas de control inteligentes se basan en la comunicación entre distintos dispositivos para el intercambio de datos o información, mediante protocolos de comunicación compatibles. A continuación, se detalla los protocolos utilizados para la implementación del sistema HMI.

2.5.1. Modbus TCP/IP entre PLC y Raspberry Pi

Permite la comunicación vie Ethernet (lectura y escritura de datos) usando el conector PROFINET de la CPU entre dispositivos servidor-cliente mediante configuraciones TCP/IP, es decir mediante asignación de dirección IP del dispositivo y puerto de enlace. Pude haber varias conexiones cliente-servidor además de la conexión con Tia portal (Siemens, 2018).

Dentro del programa, la instrucción MB_SERVER usa un DB instancia, puerto y dirección IP único, que se ejecuta individualmente por cada conexión. Mientras que el cliente Modbus (maestro) inicia la conexión cliente-servidor y la transmisión de mensaje a un servidor específico.

La comunicación directa del PLC-S71200 con Raspberry pi se realiza mediante cable Ethernet sin necesidad de módulos de comunicación adicional como se muestra en la Figura 14.

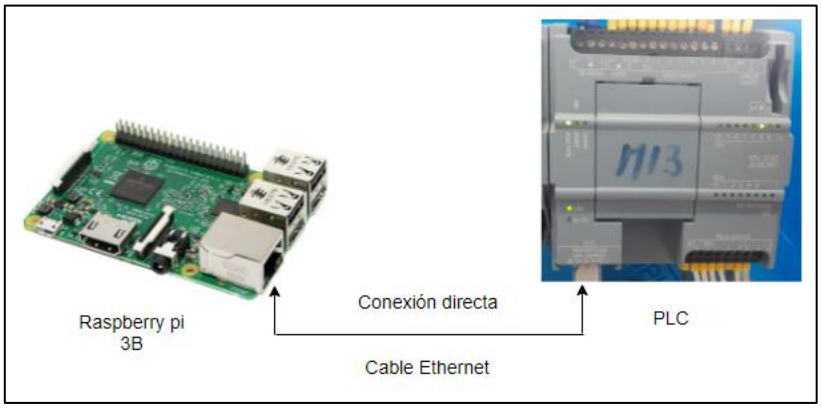


Figura 14. Conexión directa PLC- Raspberry pi por puerto Ethernet
Fuente: Autor

Para el caso de conexión entre PLC, Raspberry y el Tia portal, es necesario un Router o módulo Ethernet debido a que el CPU cuenta con un solo conector, la Figura 15 muestra un esquema de conexión.

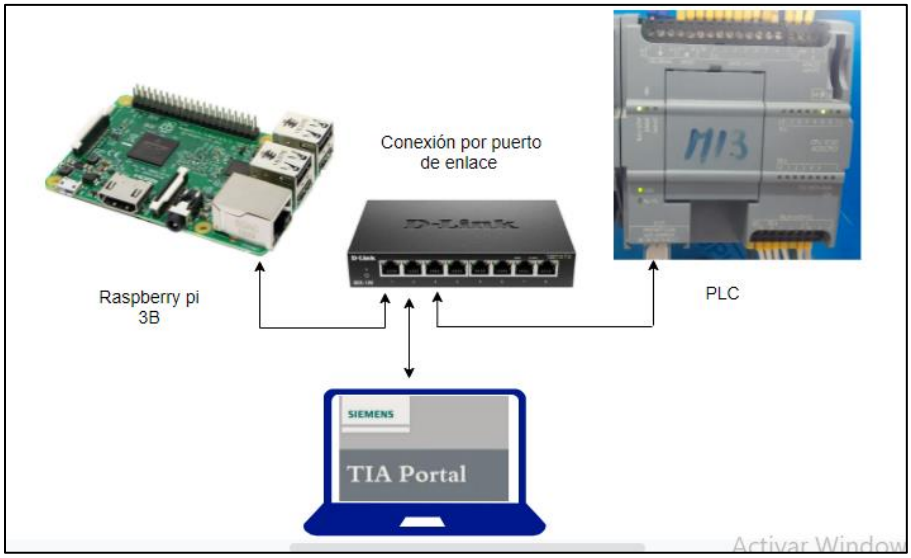


Figura 15. Conexión de PLC, Raspberry pi y Tia portal mediante un Switch Ethernet
Fuente: Autor

Para el caso particular del Router también es necesario configurar la dirección de puerta de enlace para que se pueda establecer la conexión, y el intercambio de información entre los distintos dispositivos que se encuentran conectados.

2.5.2. Modbus RTU (Remote Terminal Unit) entre PLC y Variador de frecuencia

Esta comunicación de red estándar usa conexiones RS232 o RS485 para transferencia serial de datos. Se pueden añadir puertos de red PtP (punto a punto) a una CPU con RS232, módulo de comunicación CM RS485 o tarjeta CB RS485. Modbus RTU utiliza una red maestro/esclavo en la que un solo dispositivo maestro inicia todas las comunicaciones y los esclavos solo pueden responder a una petición del maestro.

El módulo CM1241 tiene un conector hembra sub-D de 9 polos, mientras que al variador altivar cuenta con conector tipo RJ45, la descripción de los pines de los conectores se muestra en la Tabla 14.

TABLA 14. IDENTIFICACIÓN DE PINES EN CM1241 Y RJ45 DEL VARIADOR

Pines de conector Sub-D 9 CM1241		**Pines de conector RJ45 Altivar312**	
1	Masa de comunicación	1	Señal CANopen
2	Conectada para RS422	2	Señal CANopen
3	Señal B (RxD/TxD+): in/out	3	Señal CANopen
4	RTs petición de transmisión	4	D1 Modbus
5	GND	5	D0 Modbus
6	PWR	6	No conectado
7	TxD1	7	conversor RS232/RS485
8	Señal A (RxD/TxD-): in/out	8	GND
9	Shell	-	

Fuente: (Siemens, 2018), (Schneider Electric, 2010)

La comunicación Modbus del variador es una conexión semidúplex (2 hilos), mediante una señal de transmisión de datos y la otra para recepción de datos. La forma de conexión de pines se muestra en la Figura 16.

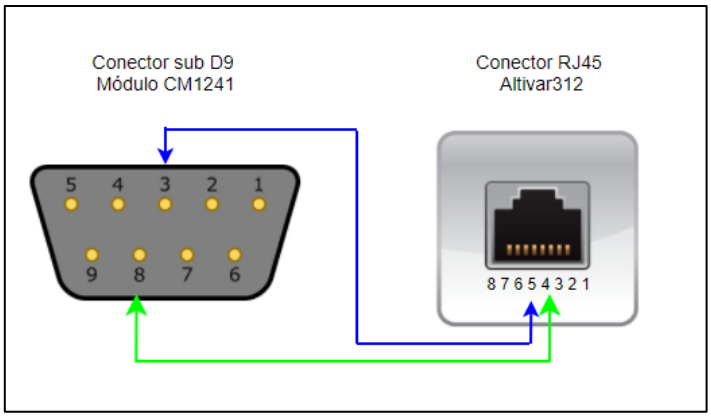


Figura 16 Conexión de pines de conector subD9 del módulo CM-1241 a RJ45 de Variador Altivar312

Fuente: Autor

Dentro del programa la instrucción “MB_MÁSTER’’ el PLC s7-1200 se comunica como maestro Modbus con el variador altivar312 de Schneider Electric en función de esclavo Modbus, mediante el módulo de comunicación punto a punto CM-1241. Para que esta instrucción pueda comunicarse con un puerto específico, debe ejecutarse previamente la instrucción "MB_COMM_LOAD" en el que se configura un puerto para la comunicación mediante el protocolo Modbus RTU.

2.6. Diagrama de bloque de conexión y diseño del sistema

En el diseño del sistema HMI se reemplaza un dispositivo HMI industrial por un panel de operador basado en Raspberry conectado a una pantalla LCD, cumpliendo así la misma función de ser la interfaz entre el operador y el proceso principal.

Para lograr el control y monitoreo del proceso conectado al PLC, se realiza la adquisición de datos mediante una red de comunicación Modbus TCP/IP entre la Raspberry que actúa como un servidor Modbus y el PLC como Cliente Modbus, gracias a su compatibilidad para comunicación mediante cable Ethernet. El estado de las entradas y salidas del PLC también son enviadas a la Raspberry para visualización en la GUI y se incluye el control de la frecuencia del Variador que interactúa con el PLC mediante una red de comunicación Modbus RTU con conexión a dos hilos. El esquema general del sistema se visualiza en la Figura 17.

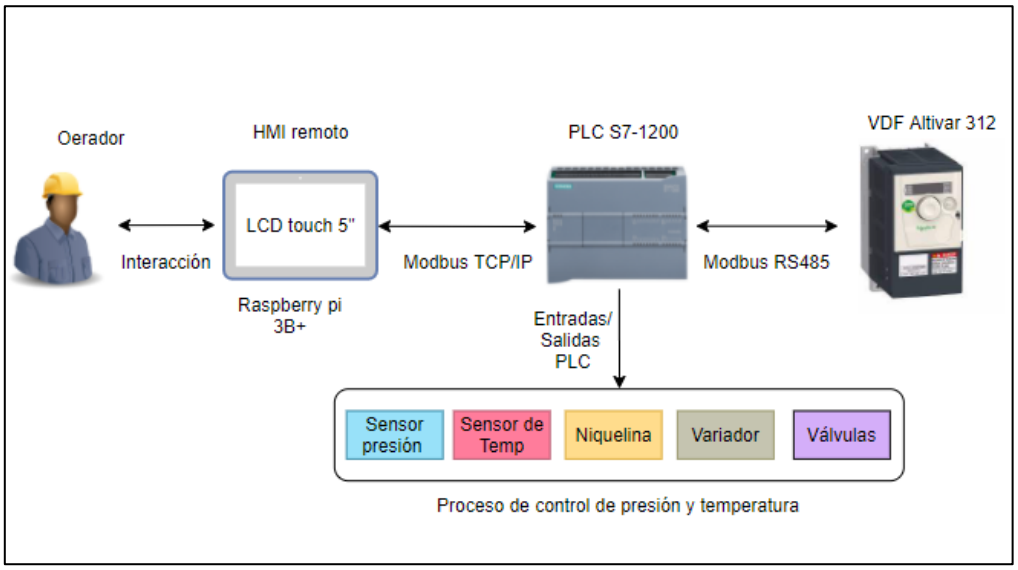


Figura 17. Diagrama general del sistema de control de presión y temperatura mediante dispositivo HMI basado en código abierto

2.7. Diagramas de flujo

Con el propósito de dar a entender el algoritmo de programación en Python para la GUI, en Tia Portal para el PLC con el modo de operación del sistema general en lazo abierto se presentan los diagramas a continuación.

2.7.1. Funcionamiento general del proceso

La programación en Ladder en el programa Tia Portal se realiza teniendo en cuenta dos modos de funcionamiento en lazo abierto: modo manual a través del tablero y modo manual a través del HMI. El diagrama de bloque de la Figura 19 indica la lógica de programación utilizada.

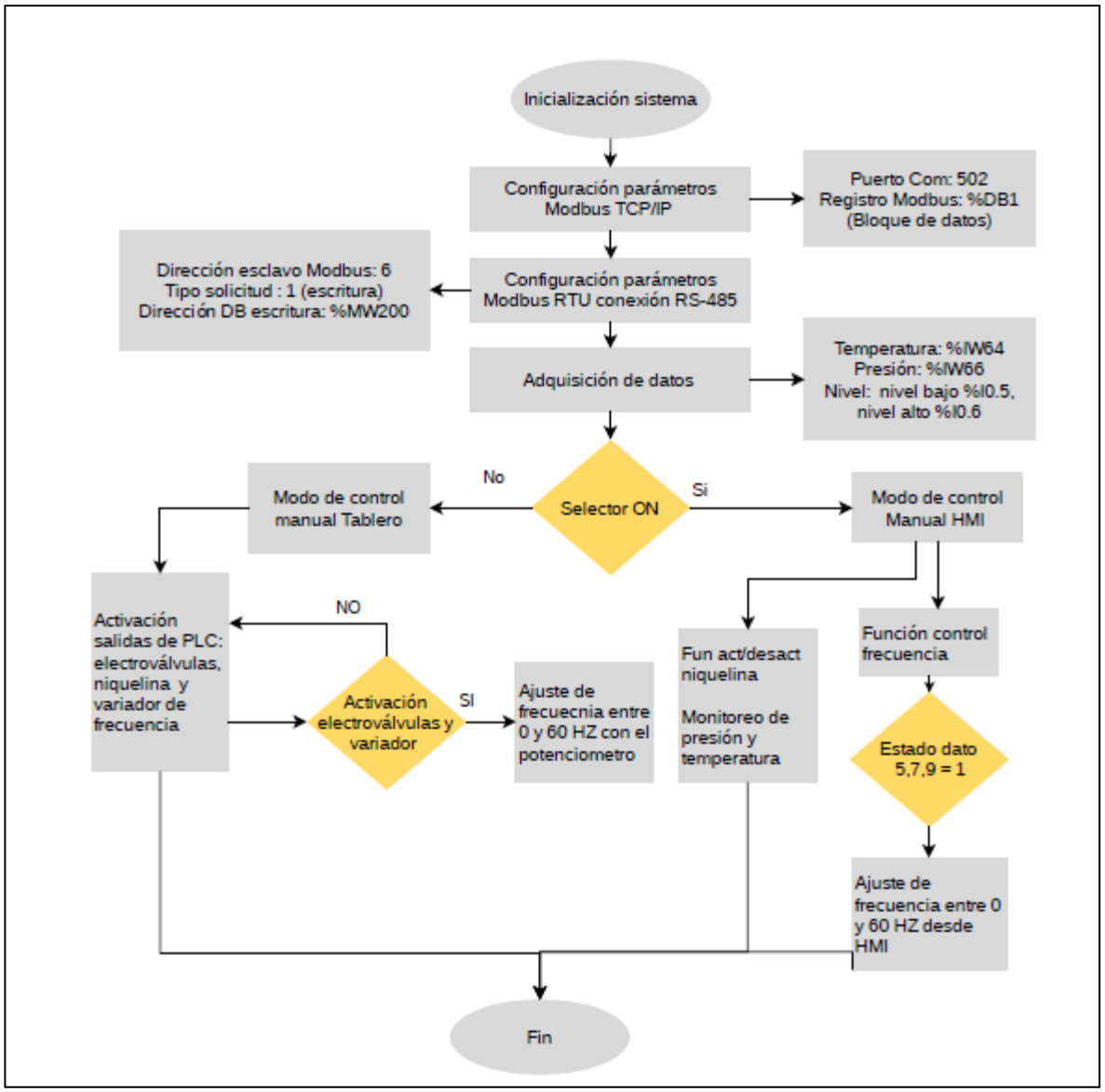


Figura 19. Diagrama programación PLC y funcionamiento general del sistema

Fuente: Autor

2.7.2. Algoritmo de programación en Python

Se presenta la lógica de programación utilizada en Python para el control y monitoreo de presión y temperatura de agua, mediante una Interfaz Gráfica ejecutado en la Raspberry pi se presenta el diagrama de flujo en la Figura 18.

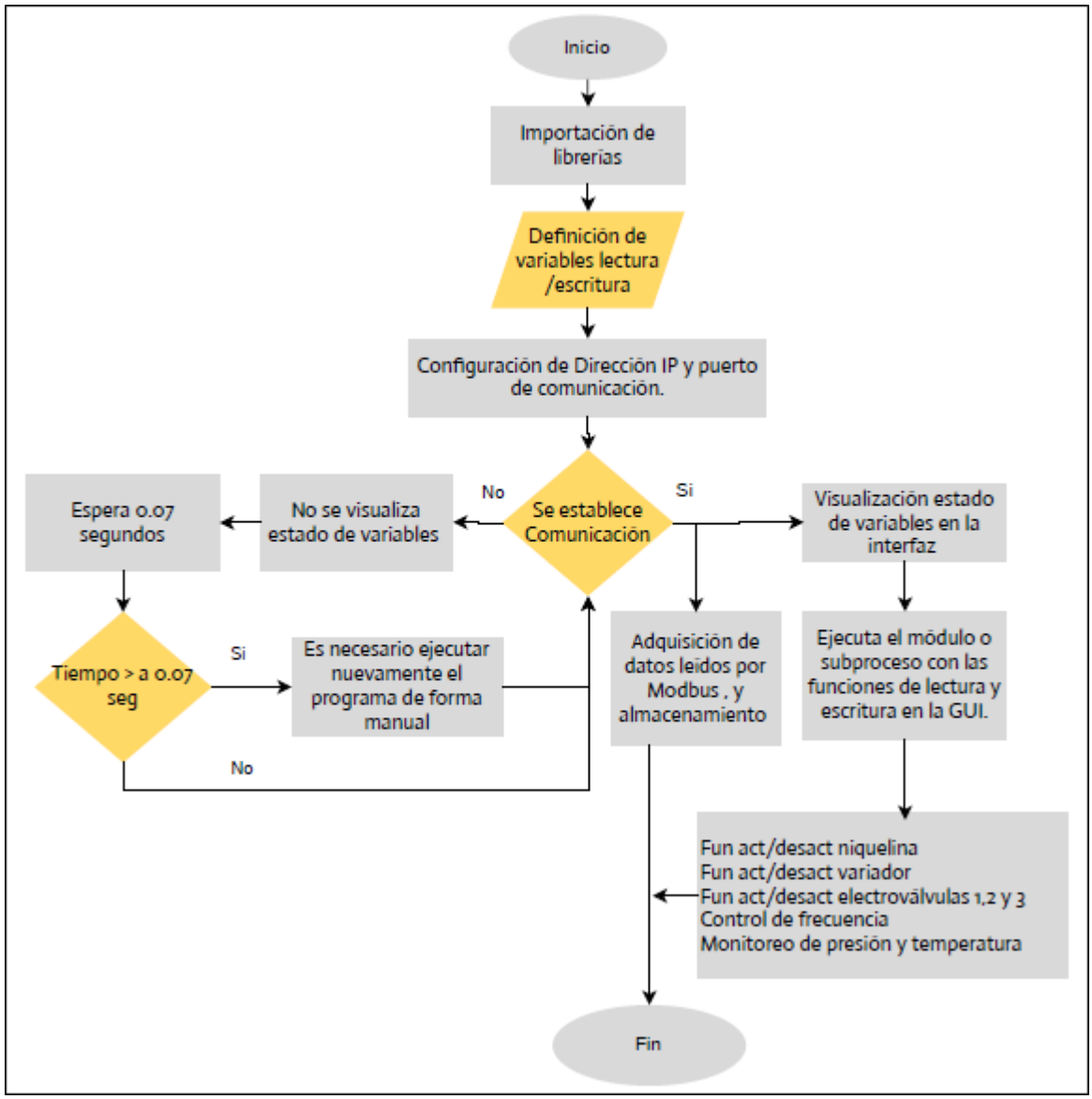


Figura 18. Algoritmo de programación para el control y monitoreo mediante GUI en Python

Para entender la programación en Tia portal del PLC de debe revisar el ANEXO E: Programación del PLC en Tia Portal, que contiene el código completo.

Una vez indicado el algoritmo de programación se detalla las variables de lectura y escritura relacionados a las variables de entrada y salida almacenadas en la base de datos DB1 del PLC S7-1200 en la TABLA 15.

TABLA 15. ASOCIACIÓN DE VARIABLES FÍSICAS CON VARIABLES DE PROGRAMACIÓN EN LA INTERFAZ GRÁFICA

Variables en Python	Variables en Base de datos de PLC	Entradas y salidas del PLC.		Elemento asociado
		Entrada	Salida	
Dato1	Activación Niquelina	I0.0	Q0.0	Niquelina
Dato2	Activación Variador	I0.1	Q0.1	Variador de F.
Dato3	Activación solenoide1	I0.2	Q0.2	Electroválvula 1
Dato4	Activación solenoide2	I0.3	Q0.3	Electroválvula 2
Dato5	Activación solenoide 3	I0.4	Q0.4	Electroválvula 3
Dato6	Estado de niquelina	I0.5		
Dato7	Estado variador	I0.6		
Dato8	Estado solenoide 1	-		
Dato9	Estado solenoide2	-		
Dato10	Estado solenoide 3	-		
Dato11	Indicador temperatura °C	-		
Dato12	Indicador temperatura °F	Analógico 0		Sensor temperatura
Dato13	Indicador presión	Analógico 1		Sensor presión
Dato14	Estado sensor tanque lleno	I0.5	-	Sensor nivel alto
Dato15	Estado sensor tanque vacío	I0.6	-	Sensor nivel bajo
Dato16	Lectura estado de selector	I0.7		Selector local/remoto
Dato17	Escritura de Frecuencia	Dirección 40803		Dirección de registro de F. en el variador

Fuente: Autor

2.8. Diseño de caja para empotrar elementos

El diseño del dispositivo HMI requiere empotrar en una caja la Raspberry conectado a la pantalla LCD, un pequeño ventilador para refrigeración debido al calentamiento rápido de la Raspberry, un switch Ethernet para conexión con varios dispositivos y un cable de alimentación para el dispositivo. Las medidas de la caja están indicadas en centímetros (cm), la Figura 19 muestra las medidas de en vista superior y lateral de la caja.

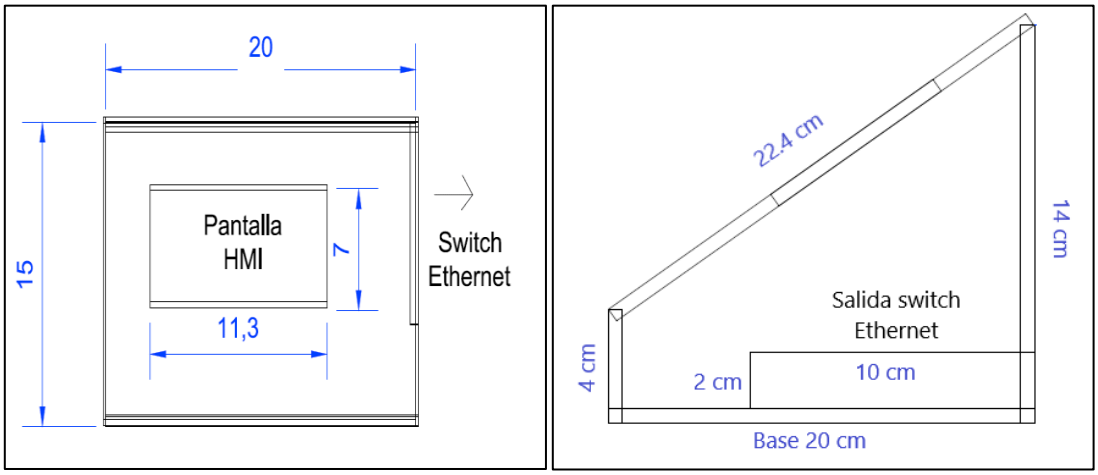


Figura 18. Vista superior y lateral de caja para insertar el Dispositivo HMI
Fuente: Autor

El modelado de la caja en 3D utilizando el software AutoCAD se muestra en la Figura 20, mostrando un tamaño adecuado para instalación en el tablero de control.

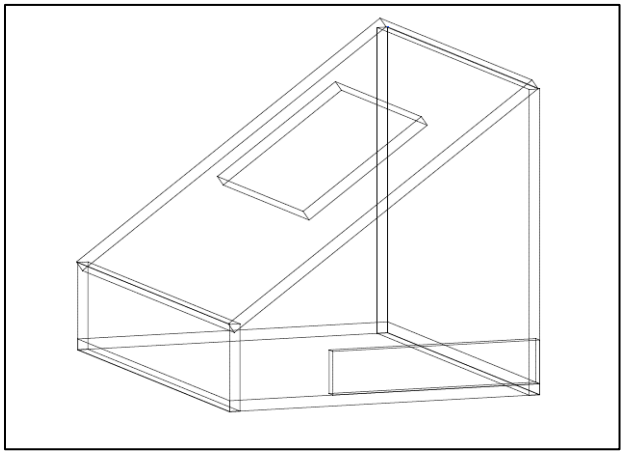


Figura 19. Vista 3D de la caja para el dispositivo HMI
Fuente: Autor

2.9. Diseño de la interfaz gráfica de usuario (GUI)

Para el diseño de la GUI fue necesario identificar el entorno de trabajo de QT Designer; los widgets u objetos con su respectiva clase y su modo de trabajo para poder darle funcionalidad en la programación Python.

2.9.1. Identificación de widgets en QT Designer

La GUI se desarrolló en el entorno de trabajo de Qt Designer presentado en la Figura 13 del capítulo 2. Cada elemento o widget utilizado para la interfaz se detalla mediante la TABLA 16.

TABLA 16. WIDGETS UTILIZADOS PARA EL DESARROLLO DE LA GUI

Cuadro de widget	**Widget**	**Clase de Widget para Python**	**Función**
Buttons	Push Button	Qbutton	Botones para encendido y apagado de niquelina, solenoide1, solenoide2, solenoide3 y variador de frecuencia.
Containers	Tabwidget	QTabwidget	Tabla para integrar el menú con las distintas ventanas (CARATULA, PLANTA, COMUNICACIÓN, VISORES, FRECUENCIA) dentro la GUI.
Input widget	Dial	QDial	Widget giratorio se utilizó para control de frecuencia en un rango de 0 a 60 HZ, de acuerdo con los datos nominales de la bomba de agua acoplada.
Display widget	Label	QLabel	Se usaron para colocar las etiquetas, rótulos, texto y especialmente para introducir luces piloto.
	LCD Number	QLCDNumber	Display digital para mostrar los datos de la temperatura y presión.
	Graphics View	QGraphicsView	Esta escena de gráficos en movimiento se utilizó para mostrar el cambio de presión y temperatura en un visor analógico.

Fuente: autor

Las configuraciones de tamaño y color de las etiquetas, texto, widget y objetos se configuran en el editor de propiedades e inspector de objetos en Qt Designer. Finalmente,

el archivo .ui de la GUI se transforma a extensión de Python .py en la terminal de Visual Studio Code para generar un archivo utilizable en Python, mediante el siguiente comando: pyuic5 nombre.ui > nombre.py

2.9.2. Diseño de interfaz en QT Designer

El desarrollo inicia con la identificación de las variables de control y variables de monitoreo que se presentó en la Tabla 15. Se agrega pulsadores para el encendido y apagado de variables (niquelina, electroválvulas y variador de frecuencia), indicadores de estado ON/OFF, Medidores analógicos de presión y temperatura de agua y un Dial para el control de frecuencia. La GUI consta de 5 ventanas, a continuación, se muestra cada una.

La ventana principal es de 480 x 400 pixeles, diseñado para encajar en la pantalla de 5 pulgadas. La primera ventana en el menú indica la CARÁTULA del proyecto y el menú de selección en la barra inferior como se muestra en la Figura 21.

Figura 20. Primera ventana GUI: Carátula

Fuente: Autor

La segunda ventana corresponde a PLANTA; cuenta con una imagen de fondo que indica los elementos del proceso. Se añade luces piloto para indicar en tiempo real el estado encendido o apagado de los elementos: niquelina, electroválvulas, sensores de nivel de agua y variador de frecuencia como indica la Figura 22.

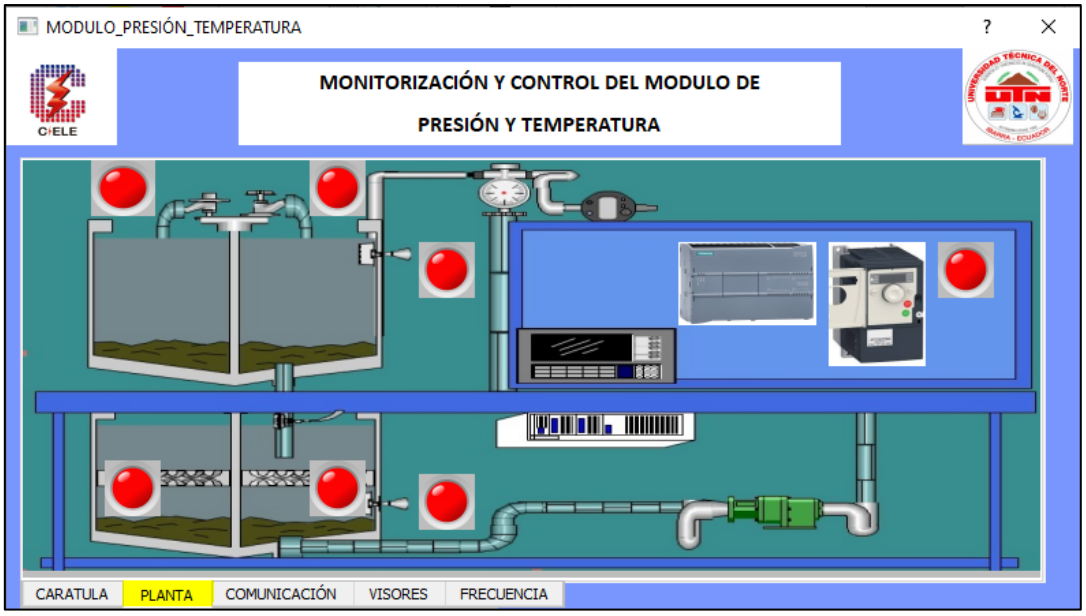


Figura 21. GUI; ventana 2 correspondiente al menú PLANTA

Fuente: Autor

La tercera ventana corresponde a COMUNICACIÓN; es la ventana de control mediante botones con función de encendido y apagado. Además, se añade luces piloto de color rojo que indica el estado OFF, que cambia de color verde en estado ON para indicar el estado de los actuadores como muestra la Figura 23.

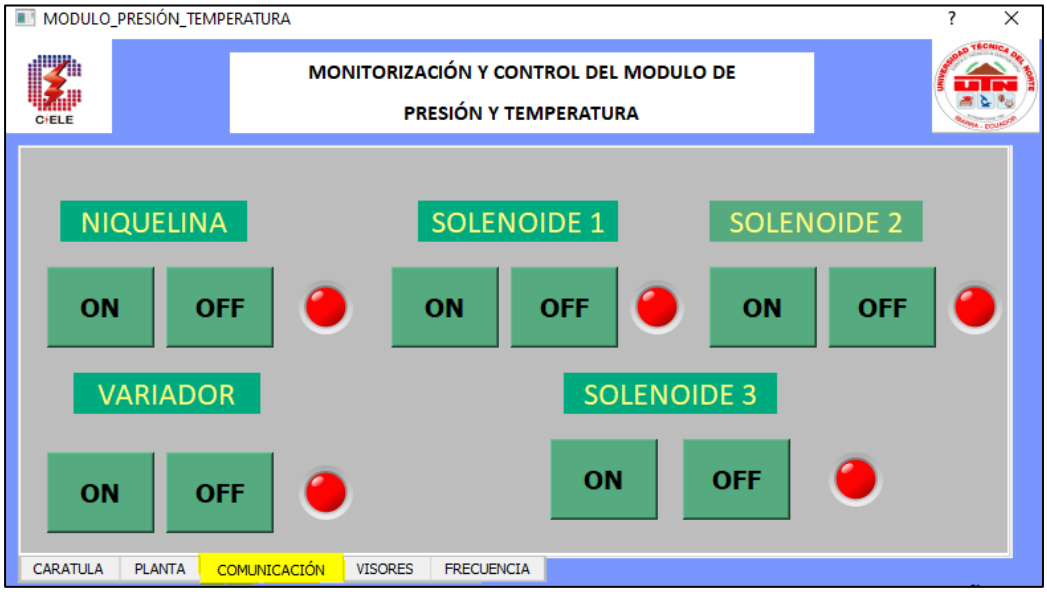


Figura 22. GUI; ventana 3 correspondiente al menú COMUNICACIÓN

Fuente: Autor

La cuarta ventana corresponde a VISORES; es la ventana de monitoreo de variables de presión y temperatura en tiempo real. Consta de dos visores analógicos conectado a un Display digital para ver los cambios constantes de las dos variables como se indica en la Figura 23.

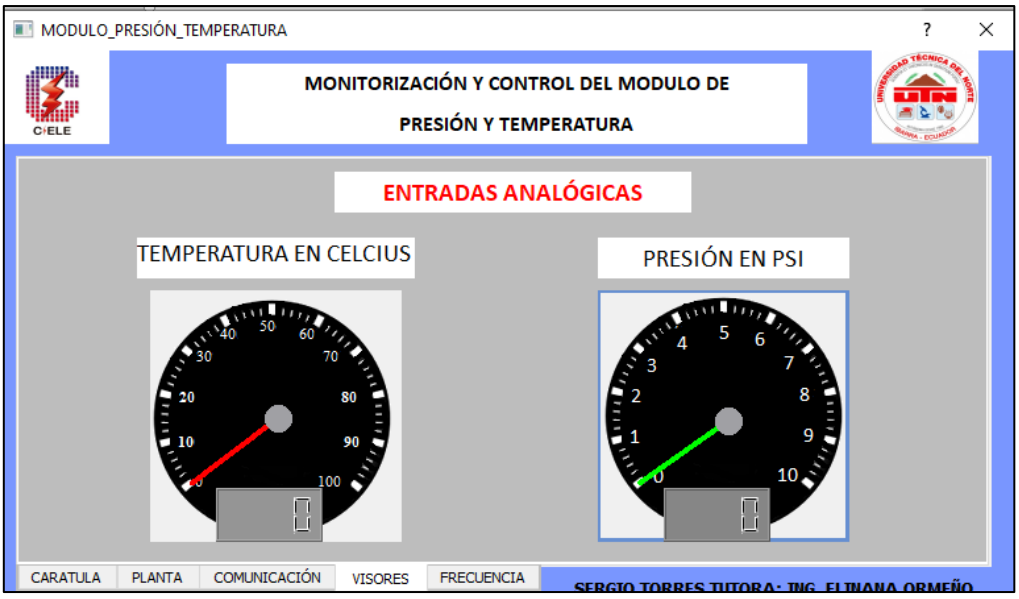


Figura 23. GUI; ventana 4 correspondiente al menú VISORES

Fuente: Autor

La quinta ventana corresponde a FRECUENCIA; es la ventana de control de frecuencia con valor ajustable entre 0 HZ y 60 HZ determinado por el valor nominal del motor instalado. El ajuste se hace mediante un widget de tipo Dial conectado a aun Display digital para visualizar el valor como indica la Figura 23.

Figura 24. GUI; ventana 5 correspondiente al menú FRECUENCIA

Fuente: Autor

Los estados de lectura y escritura de los elementos de la GUI, así como los valores analógicos de sensores se almacenan en variables de programación para intercambiar vía Modbus con el PLC S7-1200.

Por otro lado, el valor de frecuencia se almacena en una variable para ser enviado vía Modbus TCP/IP a la base de datos y posteriormente al registro Modbus del variador de frecuencia en a la dirección 48503 vía Modbus RTU.

En el siguiente capítulo se detalla la implementación de la GUI y el dispositivo HMI mediante red de comunicación Ethernet, realizando las pruebas de funcionamiento necesarios.

Discusión

Finalmente se puede analizar los resultados obtenidos en este capítulo.

La selección de equipos está basada en tecnología de uso frecuente, así en el caso de la selección del modelo 3B de Raspberry a diferencia de versiones anteriores como Raspberry Pi 2A, 2B y 3A cuenta con un puerto Ethernet con mayor velocidad de transmisión de datos, que lo hace óptimo para la comunicación con el PLC S71200, además cuenta con un puerto HDMI compatible con pantallas LCD y monitores de PC para su fácil conexión (ver figura 18).

La limitación en la memoria RAM del dispositivo que es de 1GB hasta la versión 3B, puede verse afectado en la velocidad de procesamiento de la interfaz gráfica. Sin embargo, se puede utilizar modelos como 4A o 4B, con memoria hasta 8GB a costo más elevado.

El sistema operativo de Raspberry basado en Linux que trae preinstalado un editor de texto para programación en Python ha permitido realizar y diseñar el programa de la interfaz gráfica para el control y monitoreo del sistema de presión y temperatura de agua.

La GUI permite la activación y desactivación de elementos conectados a las salidas digitales del PLC. Gracias a la programación en Python y comunicación Modbus se logra controlar y monitorear variables en tiempo real, el control de variables se puede ver en la Tabla 15. Que indica las variables asociadas tanto en la programación Python y programación en Tia portal.

CAPÍTULO 3

Implementación del sistema HMI y pruebas de funcionamiento

En este capítulo se describe la implementación del sistema HMI basado en Raspberry Pi; dispositivo en el cual se ejecuta la Interfaz Gráfica de Usuario diseñado en Qt Designer y programación Python, para controlar y monitorear un módulo de presión y temperatura de agua. Se detalla las configuraciones necesarias para realizar la comunicación entre PLC S7-1200, Raspberry Pi y variador de frecuencia Altivar 312 mediante dos protocolos de comunicación industrial; Modbus TCP/IP y Modbus RTU/RS485. Finalmente se realiza las pruebas para validar el funcionamiento y evaluar los resultados.

3.1. Montaje del dispositivo en el módulo de control de presión y temperatura

El montaje de la Raspberry Pi y pantalla de visualización se realiza en una caja de lámina de acrílico de 4 mm de grosor del diseño presentado en la figura 14 del capítulo 2. Se utilizó este material por ser excelente aislante eléctrico, para evitar cualquier contacto con las partes sensibles del controlador. El montaje se presenta en la figura 25.

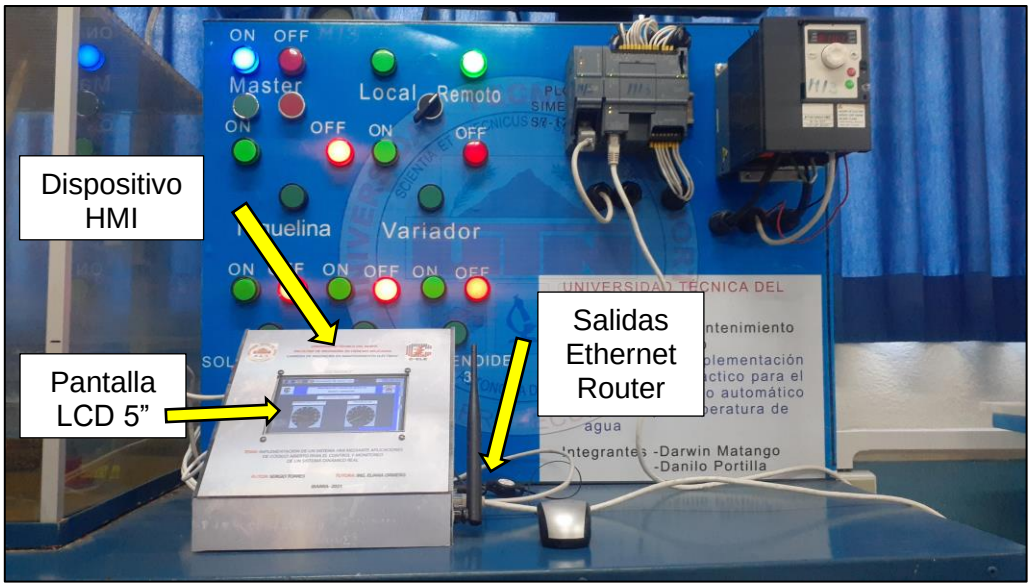


Figura 25. Montaje del Dispositivo HMI

Fuente: Autor

En el interior del dispositivo HMI se instaló un tomacorriente para conectar las fuentes para el Router y la Raspberry Pi, un pequeño ventilador para disipar el calor que

desprende el chip y un interruptor para activación del sistema. Los elementos mencionados se pueden ver en la Figura 26.

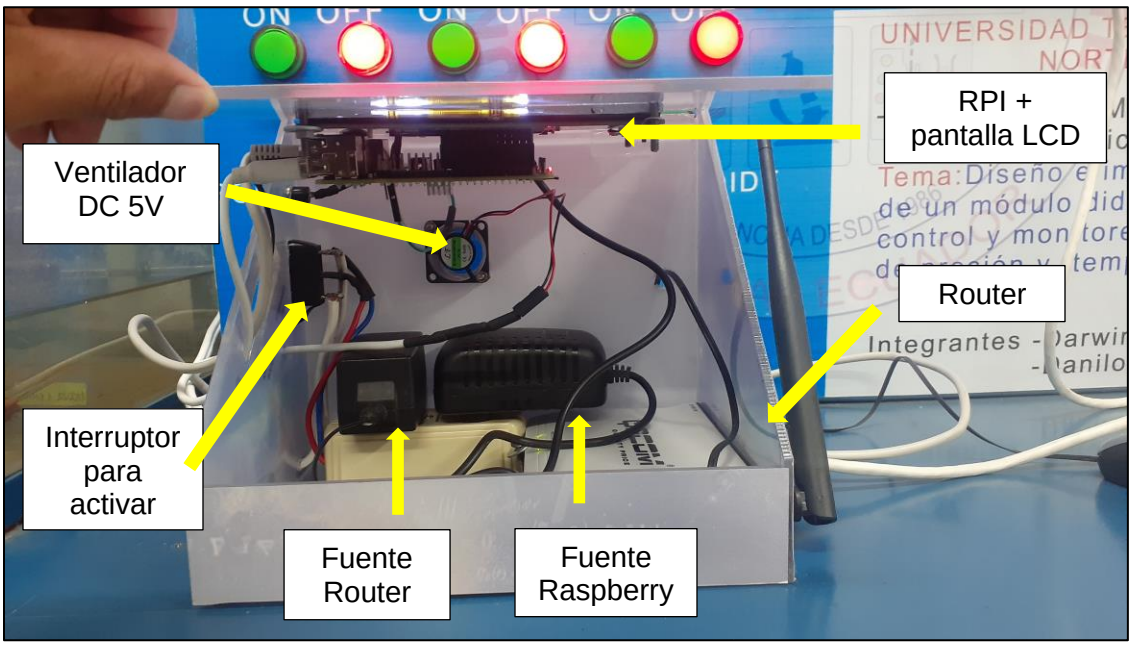


Figura 26. Elementos en el interior del dispositivo HMI

Fuente: Autor

La Raspberry Pi y la pantalla utilizan una sola fuente de alimentación DC, para ello se utiliza una fuente de 5 voltios DC a 2 Amperios, de acuerdo con las especificaciones de ambos dispositivos presentado en la TABLA 2.26.1 y la TABLA 2.27.1.

Finalmente, la conexión para la comunicación con el PLC se realiza siguiendo el esquema de la Figura 14, presentado en el capítulo 2.

3.2. Configuración de Raspberry pi

Para la utilización del microcomputador en conjunto con la pantalla LDC de 5 pulgadas es necesario la instalación del software y realizar configuraciones dentro de su terminal. El ANEXO C: Configuraciones de Raspberry, detalla paso a paso la puesta en marcha del dispositivo. Además, se realiza la instalación del controlador táctil para utilizar la pantalla LCD. El ANEXO B: Manual de usuario de pantalla HDMI 5 pulgadas, contiene los detalles de la instalación.

Por otro lado, se instaló la librería PyQT5 para ejecutar la GUI y PyModbus para la comunicación con el PLC S7-1200, la TABLA 17, muestra los comandos de instalación.

TABLA 17. COMANDOS PARA INSTALACIÓN DE LIBRERÍAS EN RASPBERRY PI

Librería	Comando	Función
Pyqt5	sudo apt-get install qt5-default	Instala las librerías con los paquetes de widgets para ejecutar el programa.
	sudo apt-get install qtcreator	Instala la aplicación Qt para desarrollar interfaces.
Modbus	sudo apt-get pip install pyModbus TCP	Instala las herramientas necesarias para comunicación con dispositivos Modbus.

Fuente: Autor

La instalación de librerías y aplicaciones necesarias para cualquier proyecto en el sistema operativo Raspberry Pi OS, se realizan mediante comandos en la terminal del dispositivo.

3.3. Configuración Comunicación Modbus TCP/IP

Para la configuración se establece inicialmente al PLC como servidor Modbus y a la Raspberry Pi como Cliente Modbus considerando el puerto de comunicación estándar 502. La Figura 27 muestra el bloque implementado con direccionamiento a DB1 que es un bloque de datos en donde están los registros de los estados de las variables al que puede acceder la Raspberry para enviar o recibir información del PLC.

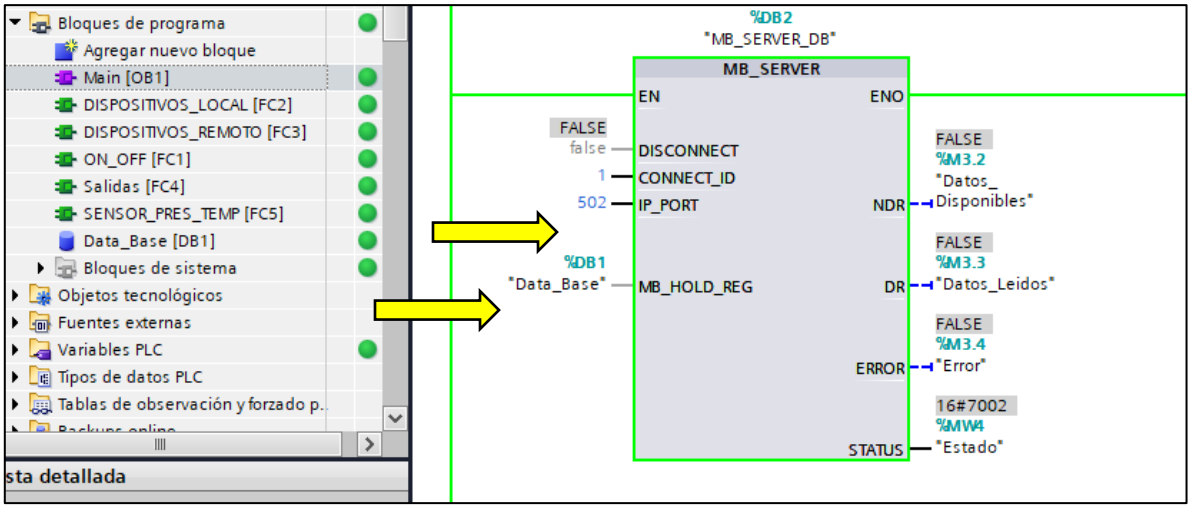


Figura 27. Bloque Servidor Modbus en Tia Portal

Fuente: Autor

Las salidas del bloque NDR y ERROR se pondrán en TRUE en caso de tener datos disponibles y error de comunicación respectivamente, cuando el programa se está ejecutando.

3.4. Configuración Modbus RTU conexión RS-485

Inicialmente se configuró el puerto para la comunicación mediante el protocolo Modbus RTU a través de la instrucción "MB_COMM_LOAD", los valores configurados en el bloque deben coincidir con los valores configurados en el Variador de frecuencia. Esta instrucción se podrá utilizar si se cuenta con los módulos punto a punto (PtP) CM 1241 RS485 o CM 1241 RS232

La Figura 28 indica el bloque MB_COMM_LOAD estableciendo el módulo CM1241 como puerto de comunicación, la velocidad de transmisión 19200 bits/s, paridad par y estableciendo una referencia al bloque de datos de la instrucción "MB_MASTER".

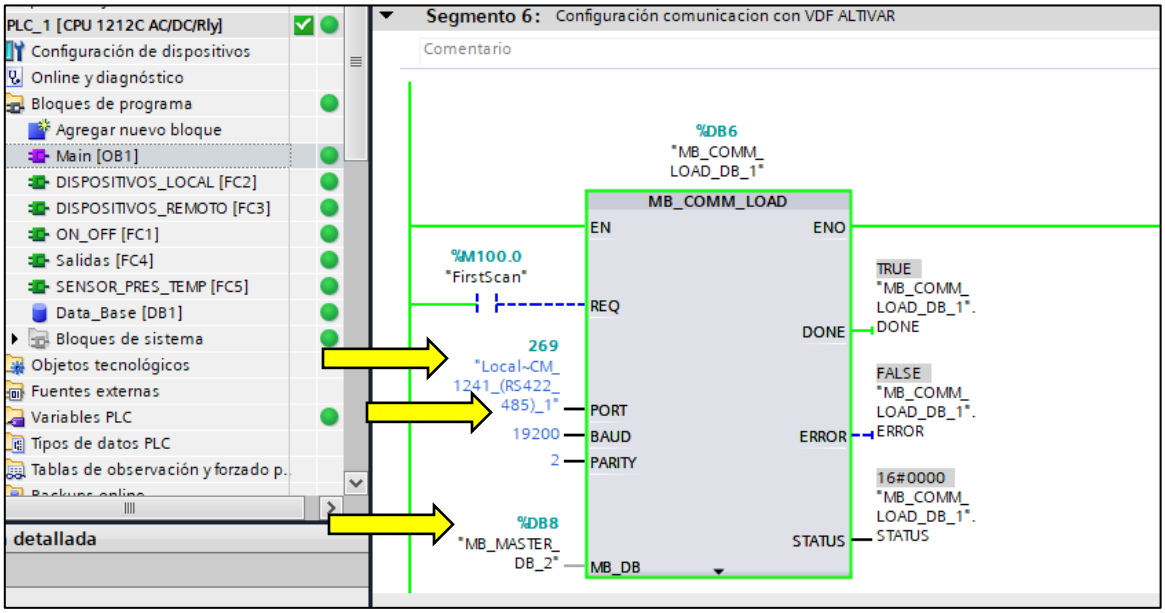


Figura 28. Instrucción MB_COMM_LOAD para configurar puerto de comunicación con Modbus RTU

Fuente: Autor

Luego la instrucción "MB_MASTER" permite al programa comunicarse como maestro Modbus. En donde se configura la dirección del esclavo Modbus, registro Modbus para control de Hz, longitud de datos y entrada de frecuencia como indica la Figura 29

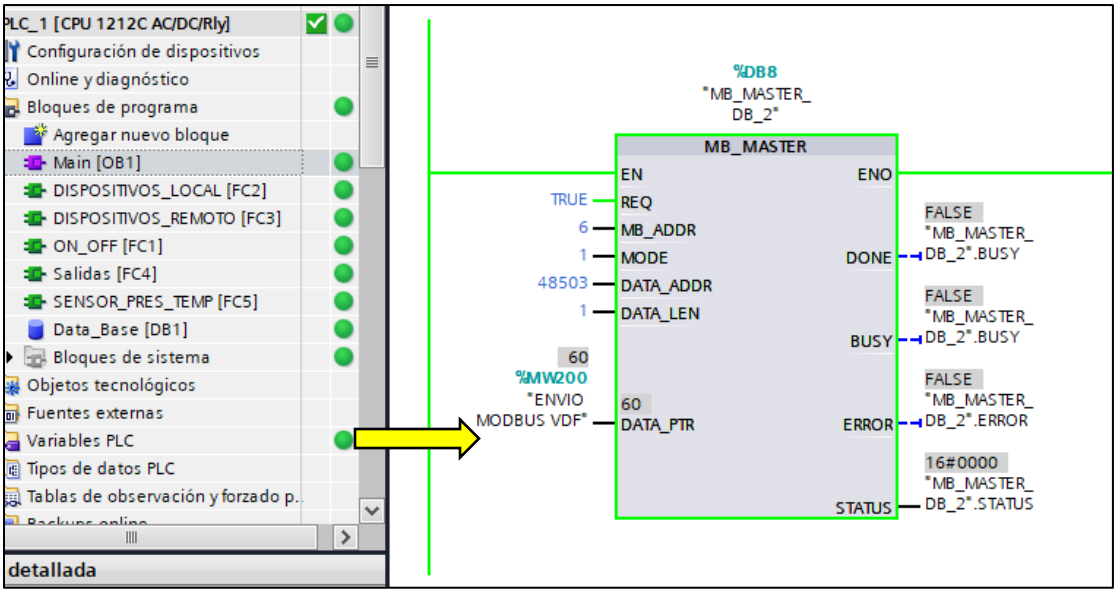


Figura 29.Configuración del bloque MB_MASTER

Fuente: Autor

Los datos que envía la Raspberry hacia la DB1 (bloque de datos del PLC) se guardan en formato Real de 8 bits, luego el envío del valor de frecuencia que ingresa en DATA_PTR de la Figura 28 se realiza en formato entero de 8 bits, mediante una conversión de datos.

3.5. Configuración direcciones IP

La comunicación Modbus TCP/IP necesita asignar una dirección IP a todos los dispositivos que se desee intercambiar información. La TABLA 18, detalla las direcciones asignadas.

TABLA 18. ASIGNACIÓN DE DIRECCIÓN IP A LOS DISPOSITIVOS

Dispositivo	**Dirección IP**
PLC S7-1200	192.168.1.5
PC	192.168.1.241
Raspberry Pi	192.168.1.12

3.6. Configuración Variador de Frecuencia Altivar 312

Dentro del Variador de frecuencia se realizó la configuración en las funciones: set (rango de frecuencia), drC (características del motor), ctl (control) y Con (parámetros de comunicación) en condiciones de parada sin orden de marcha. Los códigos de ingreso y valores establecidos se muestran en la TABLA 19.

TABLA 19. CONFIGURACIÓN DEL VARIADOR DE FRECUENCIA

Función	Código	Descripción	Valor establecido
set	LSP	Frecuencia mínima	0
	HSP	Frecuencia máxima	60
	itH	Intensidad nominal del motor	4,9
drC	bFr	Frecuencia estándar del motor	60
	UnS	Voltaje nominal motor	230
	FrS	Frecuencia nominal del motor	60
	nCr	Intensidad nominal del motor a 230 V	4,9
	nSp	Velocidad nominal en rpm	3450
ctl	LAC	Gestión de los canales.	L3
	Fr1	Canal de referencia potenciómetro	AIUI
	Fr2	Canal de referencia Modbus	Mbs
	rFC	Selección canal actual	Fr1
	CHCF	(canales de control separados de los canales de consigna)	Sep
	Cd1	Canal de consigna potenciómetro	ter
	Cd2	Canal de consigna Modbus	Mbd
	CCS	Selección de canal	Cd2
con	Add	Dirección Modbus Entre 0 y 254	6
	tbr	Velocidad de transmisión	1900 bits/s
	tFo	Formato Modbus: 8 bits paridad par	8E1

Fuente: Autor

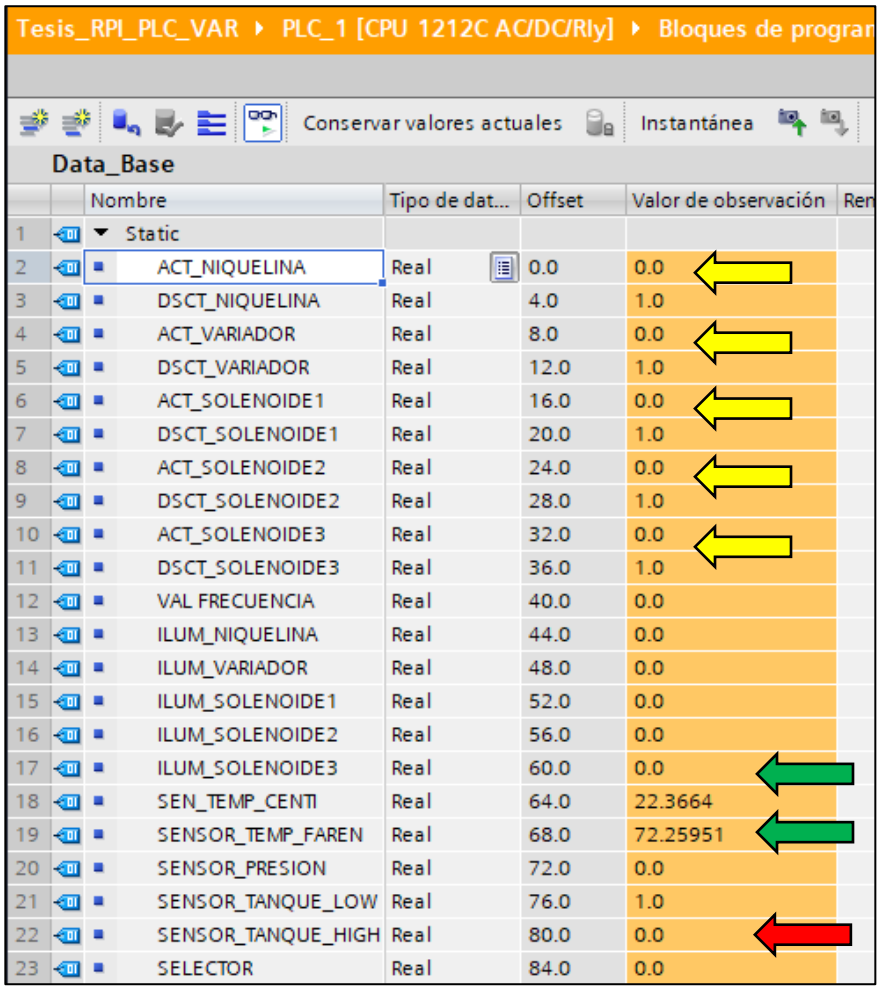
Otros ajustes con respecto al modo de arranque o frenado no están considerados dentro de la tabla.

3.7. Pruebas de funcionamiento

Para validar el correcto funcionamiento y transferencia de datos del sistema se consideró la evaluación de la respuesta en modo de operación local a través de elementos de entrada del tablero y a través del dispositivo HMI para activar y desactivar los elementos conectados a las salidas del PLC. Ambos tipos de operación representan un control en lazo abierto.

3.7.1. Estado inicial del sistema y monitoreo en la pantalla HMI y TIA Portal

La Figura 30, muestra los estados de las variables del PLC en estado inicial, es decir; cuando todas las salidas están desactivadas. Por otro lado, las entradas correspondientes al sensor de presión y temperatura se leen en tiempo real como se visualiza en las filas 18 y 20. El estado del selector del tablero será 0. Y se podrá utilizar los pulsadores físicos del tablero para activar y desactivar las salidas del PLC. Para visualizar el estado de las variables se utilizó el software Tia Portal estableciendo conexión con el PLC.



	Nombre	Tipo de dat...	Offset	Valor de observación
1	Static			
2	ACT_NIQUELINA	Real	0.0	0.0
3	DSCT_NIQUELINA	Real	4.0	1.0
4	ACT_VARIADOR	Real	8.0	0.0
5	DSCT_VARIADOR	Real	12.0	1.0
6	ACT_SOLENOIDE1	Real	16.0	0.0
7	DSCT_SOLENOIDE1	Real	20.0	1.0
8	ACT_SOLENOIDE2	Real	24.0	0.0
9	DSCT_SOLENOIDE2	Real	28.0	1.0
10	ACT_SOLENOIDE3	Real	32.0	0.0
11	DSCT_SOLENOIDE3	Real	36.0	1.0
12	VAL FRECUENCIA	Real	40.0	0.0
13	ILUM_NIQUELINA	Real	44.0	0.0
14	ILUM_VARIADOR	Real	48.0	0.0
15	ILUM_SOLENOIDE1	Real	52.0	0.0
16	ILUM_SOLENOIDE2	Real	56.0	0.0
17	ILUM_SOLENOIDE3	Real	60.0	0.0
18	SEN_TEMP_CENTI	Real	64.0	22.3664
19	SENSOR_TEMP_FAREN	Real	68.0	72.25951
20	SENSOR_PRESION	Real	72.0	0.0
21	SENSOR_TANQUE_LOW	Real	76.0	1.0
22	SENSOR_TANQUE_HIGH	Real	80.0	0.0
23	SELECTOR	Real	84.0	0.0

Figura 30. Estado inicial de las variables de salida

Fuente: Autor

La señal recibida por los sensores en las filas 18 y 20, se envían a la pantalla HMI y se visualiza en la ventana “VISORES” en forma analógica y digital como indica la Figura

31. Estos valores estarán disponibles mientras haya comunicación entre PLC y la Raspberry.

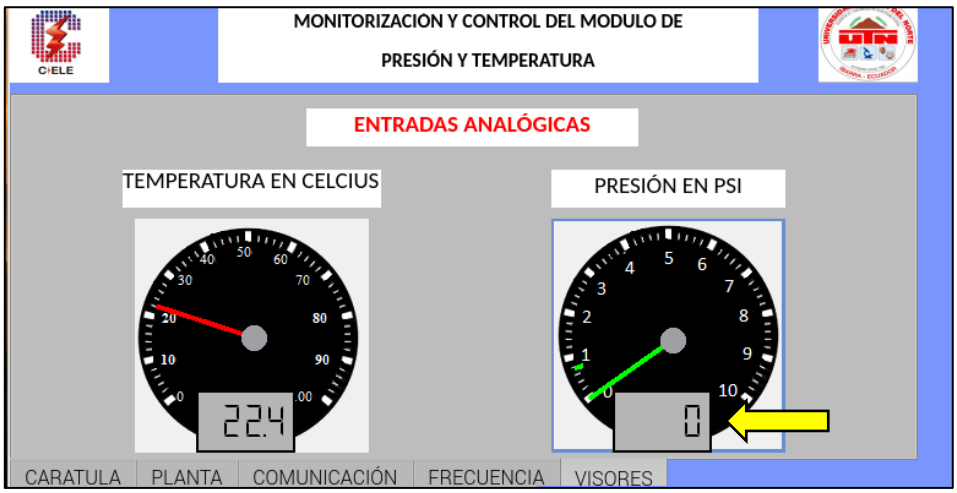


Figura 31. Visualización de presión y temperatura en estado inicial
Fuente: Autor

En la Figura 32, la ventana “PLANTA” muestra que todos los LED’s indicadores están apagados a excepción del LED correspondiente al selector “nivel bajo”, el cual indica que el tanque inferior contiene agua para bombear al tanque superior.

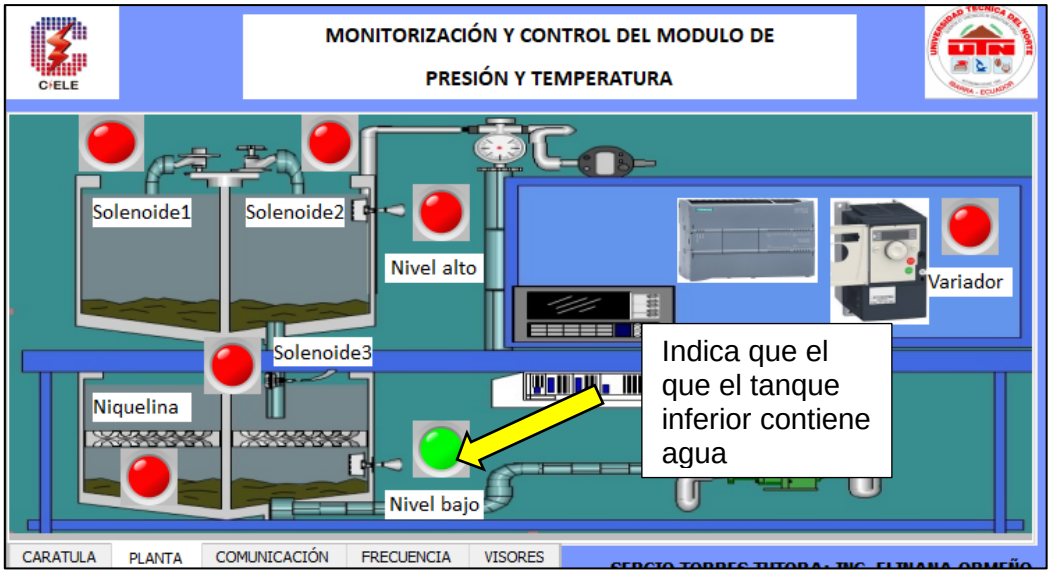


Figura 32. Estado inicial apagado de las salidas del PLC
Fuente: Autor

El estado de cada LED se mantiene en rojo hasta activar alguna salida del PLC

3.7.2. Prueba 1: Activación de las salidas solenoide 1, 2, 3 y variador de frecuencia

La Figura 33, muestra la activación de las electroválvulas y el variador de frecuencia en el tablero por medio de los pulsadores.

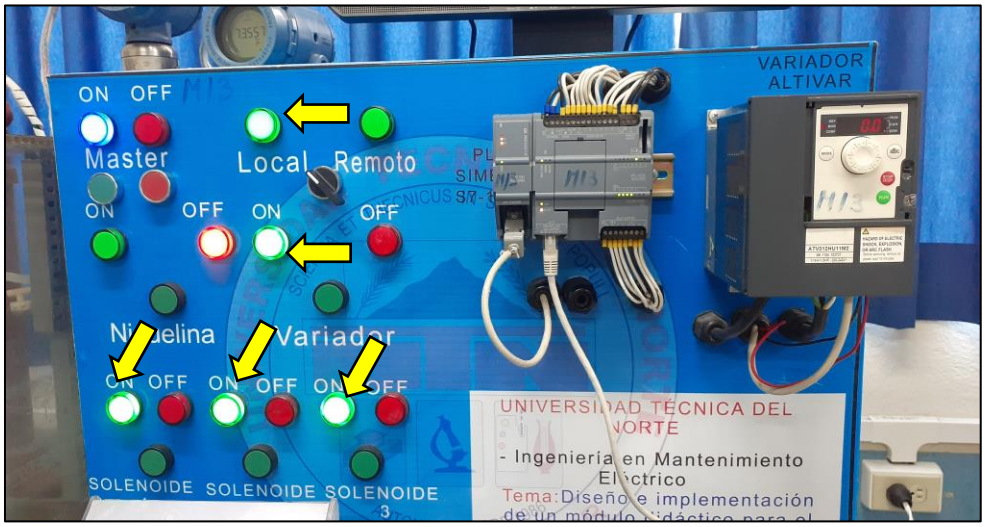
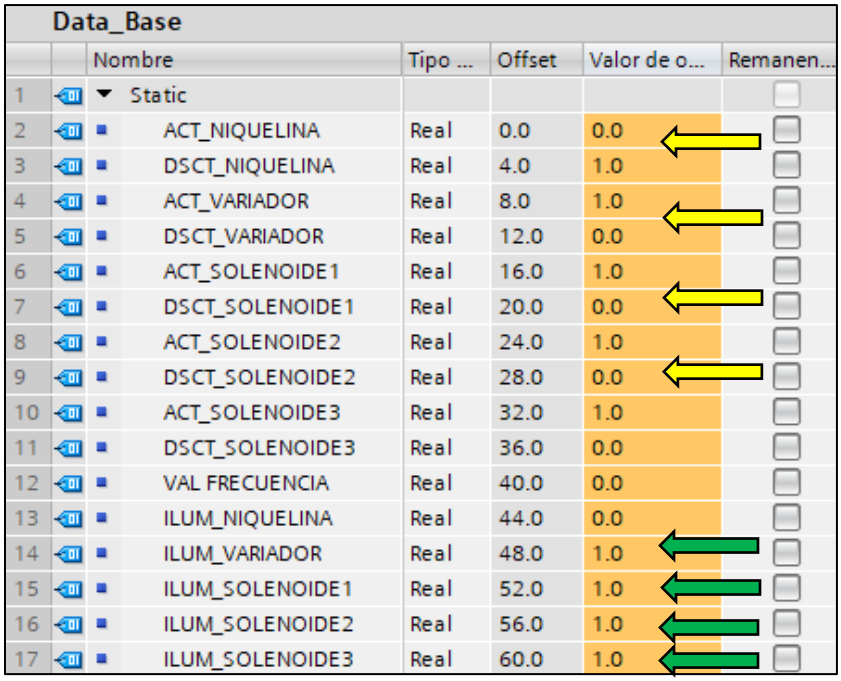


Figura 33. Activación de salidas del PLC
Fuente: Autor

El estado de las salidas se indica en la Figura 34, en el software TIA Portal. Las filas 4,6,8 y 10 corresponden a la activación de las salidas de las electroválvulas y el variador, mientras que las filas 14,15,16 y 17 leen el estado de las salidas para enviar los datos hacia los LED´s indicadores en la pantalla HMI.



Data_Base

		Nombre	Tipo ...	Offset	Valor de o...	Remanen...
1		Static				
2		ACT_NIQUELINA	Real	0.0	0.0	
3		DSCT_NIQUELINA	Real	4.0	1.0	
4		ACT_VARIADOR	Real	8.0	1.0	
5		DSCT_VARIADOR	Real	12.0	0.0	
6		ACT_SOLENOIDE1	Real	16.0	1.0	
7		DSCT_SOLENOIDE1	Real	20.0	0.0	
8		ACT_SOLENOIDE2	Real	24.0	1.0	
9		DSCT_SOLENOIDE2	Real	28.0	0.0	
10		ACT_SOLENOIDE3	Real	32.0	1.0	
11		DSCT_SOLENOIDE3	Real	36.0	0.0	
12		VAL FRECUENCIA	Real	40.0	0.0	
13		ILUM_NIQUELINA	Real	44.0	0.0	
14		ILUM_VARIADOR	Real	48.0	1.0	
15		ILUM_SOLENOIDE1	Real	52.0	1.0	
16		ILUM_SOLENOIDE2	Real	56.0	1.0	
17		ILUM_SOLENOIDE3	Real	60.0	1.0	

Figura 34. Activación de solenoide 1 y solenoide 2

Fuente: Autor

Los valores de salida señalados con las flechas amarillas también se activan en la pantalla HMI para mantener el estado cuando se cambia el modo de operación. La Figura 34, indica la activación de salidas con LED´s de color verde.

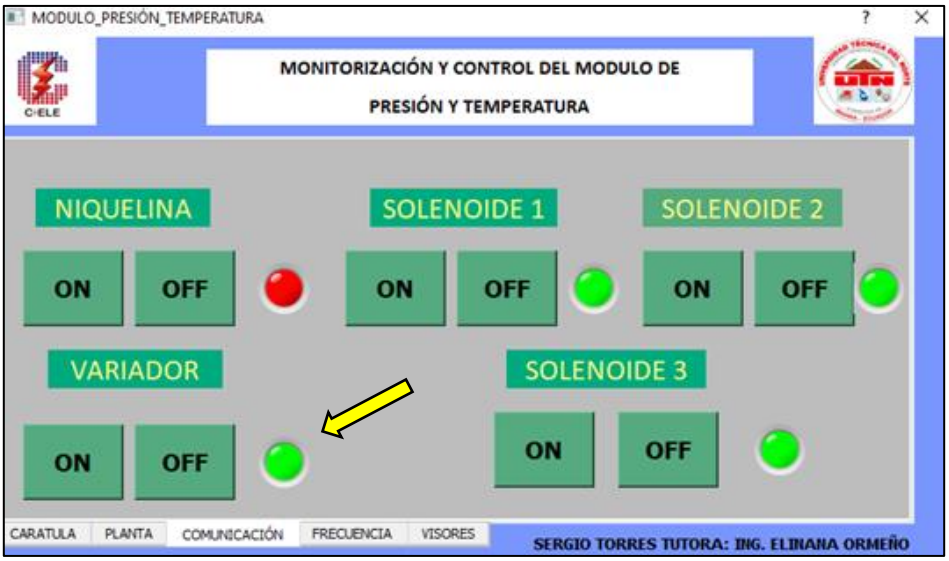


Figura 35. Activación de salidas de PLC: electroválvulas y variador

Fuente: Autor

La Figura 25 indica la ventana “PLANTA” del HMI, en el cual también se puede ver aquellos elementos que se encuentran activados. Para activar o desactivar las salidas restantes desde el HMI solo será necesario cambiar la posición del selector.

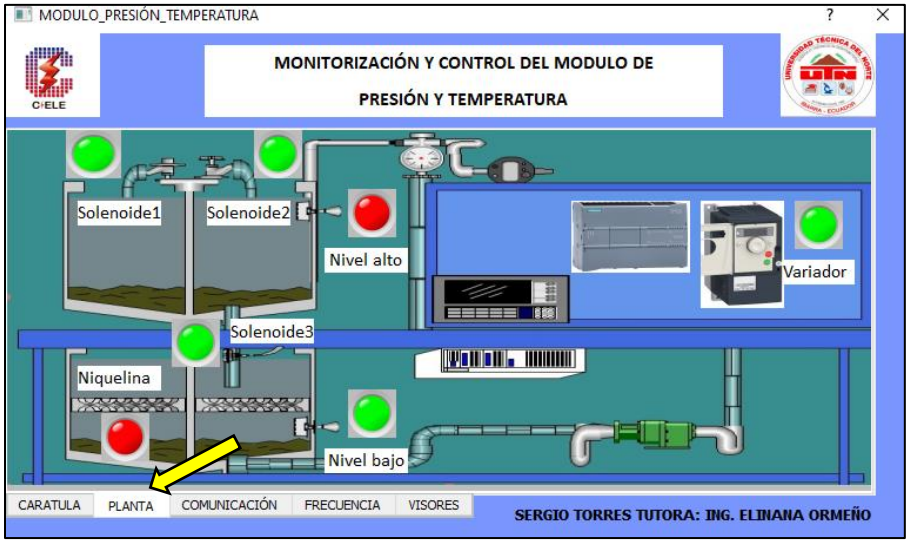


Figura 36. Monitoreo de estado de salidas del PLC

Fuente: Autor

3.7.3. Prueba 2: Control de presión por cambio de frecuencia desde el HMI

El control de presión de agua desde el HMI se realiza en la ventana “FRECUENCIA”, en el que se puede establecer con el potenciómetro un valor entre 0 y 60 Hz como indica la Figura 37.

Figura 37. Control de Frecuencia a través del HMI
Fuente: Autor

El valor establecido de 20 HZ se puede visualizar en el Display y en la pantalla del Variador de frecuencia el mismo valor.

La Figura 38, muestra como la presión sube a 1,84 PSI al establecer 20 HZ, además se visualiza en la pantalla del sensor 2 PSI. La diferencia de valores es debido a que el sensor solo muestra valores enteros y el HMI muestra incrementos en decimales.

Figura 38. Salida de presión a 20 Hz

Fuente: Autor

El valor de frecuencia establecido desde el HMI se envía al bloque de datos del PLC y este se encarga de escribir dicho valor en la dirección 8502 del variador de frecuencia.

3.8. Tabla de control de presión por variación de frecuencia

La Figura 30 muestra la gráfica de cambio de presión por variación de la frecuencia entre 0 y 60 Hz, en el que se obtiene presión de salida entre 0 y 13 PSI. Además, muestra una línea de tendencia para definir que el cambio de presión es no lineal.

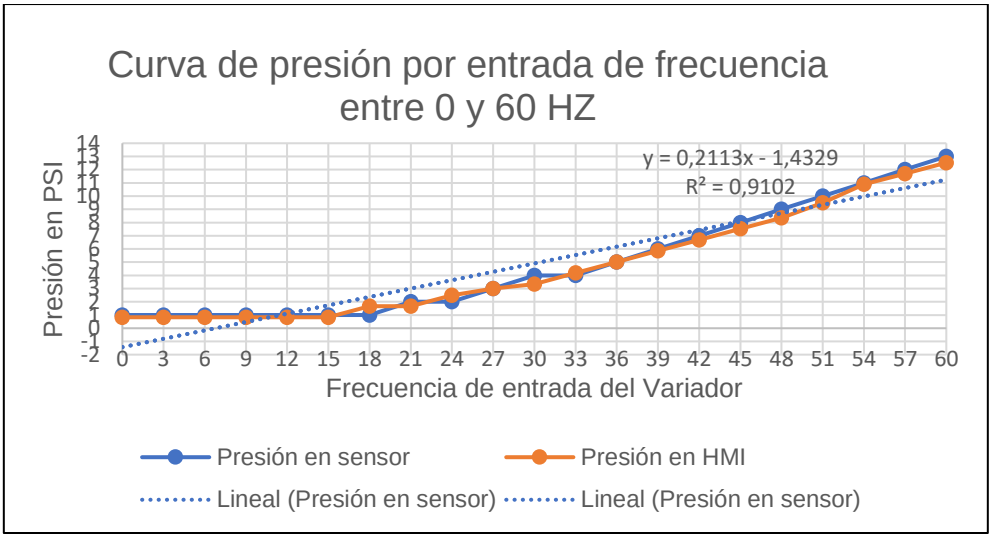


Figura 39. Control de presión por variación de presión

Fuente: Autor

La TABLA 20, indica los valores de presión de salida dependiendo de los valores seteados en el variador de frecuencia.

TABLA 20. CONTROL DE PRESIÓN POR VARIACIÓN DE FRECUENCIA

Control de frecuencia con potenciometro del Variador			
potenciometro (0-100 %)	Entrada HZ	Presión en sensor	Presión en HMI
0	0	1	0,83
5	3	1	0,83
10	6	1	0,83
15	9	1	0,83
20	12	1	0,83
25	15	1	0,83
30	18	1	1,66
35	21	2	1,66
40	24	2	2,5
45	27	3	3,01
50	30	4	3,33
55	33	4	4,19
60	36	5	5,02
65	39	6	5,85
70	42	7	6,68
75	45	8	7,52
80	48	9	8,35
85	51	10	9,5
90	54	11	10,9
95	57	12	11,7
100	60	13	12,53

Fuente: Autor

Los datos de salida resultan en que la resolución del visor del HMI es mejor ya que los datos de los sensores son calculados con decimales, a diferencia del sensor industrial que indica solo valores enteros.

3.9. Tabla de control de temperatura por encendido de niquelina

El módulo didáctico presentado en la Figura 9, del capítulo 2 tiene dos tanques de 64000 centímetros cúbicos cada uno. Al activar la niquelina de 8 ohmios el cambio de temperatura es como se muestra en el gráfico de la Figura 40, en donde se comparan los valores que indica la pantalla del sensor de presión y el valor en la pantalla HMI.

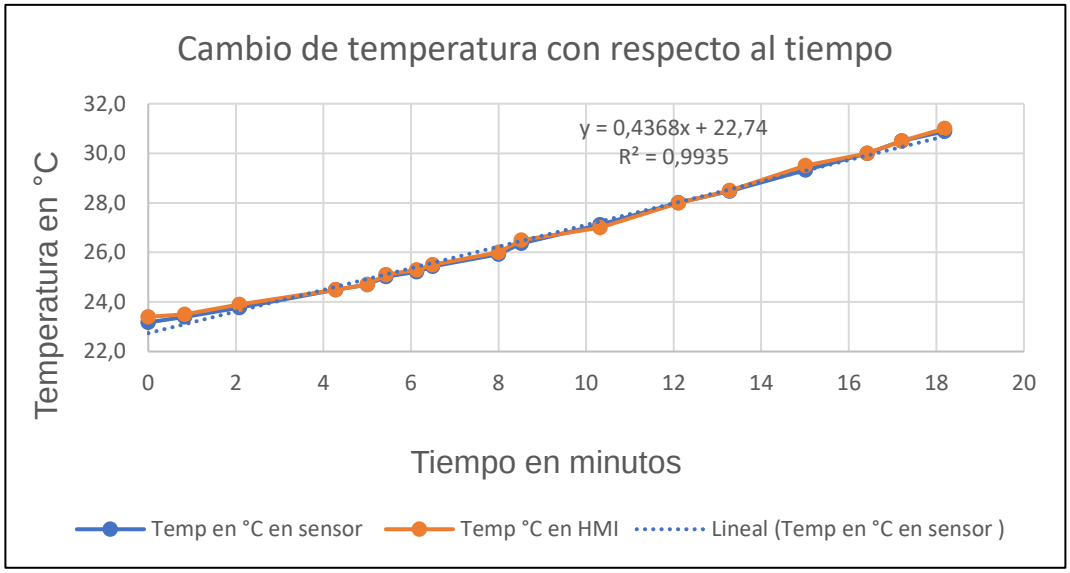


Figura 40. Cambio de temperatura con respecto al tiempo

Fuente: Autor

Los valores resultantes son muy similares, lo que indica que la pantalla HMI muestra un valor muy cercano al valor medido por el sensor. Además, se verifica que el cambio de temperatura es lento pero lineal puesto que los valores tampoco se alejan de los valores de la línea de tendencia el cual se encuentra con puntos azules.

La TABLA 21, contiene la temperatura de salida con respecto al tiempo en el que se tomó datos encendiendo la niquelina durante 18 minutos. La diferencia máxima que existe entre los valores que indica el sensor y el que indica el HMI es 0,2 °C y 0,5 °C con respecto a la recta lineal.

TABLA 21. CAMBIO DE TEMPERATURA CON RESPECTO AL TIEMPO

Tiempo en min	Temp en °C en sensor	Temp °C en HMI	Diferencia	Diferencia con recta lineal
0	23,2	23,4	0,2	0,5
0,83	23,4	23,5	0,1	0,3
2,08	23,8	23,9	0,1	0,2
4,28	24,5	24,5	0,0	0,0
5,01	24,7	24,7	0,0	-0,2
5,43	25,0	25,1	0,1	0,0
6,13	25,2	25,3	0,1	-0,1
6,5	25,4	25,5	0,1	-0,1
8	25,9	26	0,1	-0,2
8,52	26,4	26,5	0,1	0,0
10,32	27,1	27	-0,1	0,0
12,11	28,0	28	0,0	0,1
13,28	28,5	28,5	0,0	0,1
15,01	29,3	29,5	0,2	0,2
16,42	30,0	30	0,0	0,2
17,21	30,5	30,5	0,0	0,4
18,19	30,9	31	0,1	0,4
Promedio			0,1	0,1

Fuente: Autor

La diferencia entre los valores en promedio son de 0,1 °C lo cual indica que hay una desviación mínima y que se leen los datos en tiempo real con mucha precisión en la pantalla HMI. Por otra parte, para un incremento de temperatura en 1 °C se necesita que la niquelina este encendido durante 2,20 minutos en promedio.

3.10. Análisis de costo entre HMI basado en Raspberry y HMI industrial

Para realizar el análisis de costo se presenta la TABLA 22, con las marcas de dispositivos HMI comerciales dentro de Ecuador. Se consideró el tamaño de la pantalla, compatibilidad de comunicación con otros dispositivos y funcionalidad.

Si bien los dispositivos comerciales son adoptados en los procesos industriales por ser más seguros y confiables, los dispositivos de código abierto nos brindan mucha flexibilidad en los tipos de soluciones que se puede proporcionar. Esto le da una posición importante en aplicaciones de proyectos de investigación y experimentación, por lo cual su aplicación depende mucho del costo.

TABLA 22. COSTO PROMEDIO DE DISPOSITIVOS HMI COMERCIALES

Panel HMI	**Tamaño**	**Costo promedio USD**	**Diferencia USD**	**Ahorro %**
SIEMENS	4,3”	260	80	44,4
Delta	7”	279	99	55
KINCO	7”	220	30	16,7
HMI Raspberry	5”	180	80	0

Fuente: Autor

Con la implementación del dispositivo HMI basado en aplicaciones de código abierto, que cumple con la funcionalidad de control y monitoreo básico, se logra un ahorro del 16,7 % con respecto a los dispositivos de la marca Kinco. Por otro lado, frente a marcas más avanzadas como Siemens se logra tener un ahorro de hasta 44,4 % lo cual hace que el proyecto sea muy rentable dependiendo de la aplicación.

Finalmente, el ahorro frente a panel HMI de mayor tamaño que brinda más resolución en sus pantallas y la incorporación de más funcionalidades supone un ahorro de hasta el 55%. Dándole una excelente competitividad con respecto a los dispositivos comerciales.

Conclusiones

- Mediante el análisis de tecnologías de código abierto, se determinó que el software Visual Studio Code y Qt Designer en conjunto con el lenguaje de programación Python permiten trabajar con facilidad en cualquier plataforma o sistema operativo, además por su bajo consumo de memoria RAM e instalación sencilla. Por otra parte, el hardware Raspberry Pi tiene una incidencia muy fuerte en proyectos académicos, para este trabajo en particular facilitó la implementación y ejecución de una GUI con una comunicación estándar con el PLC S7-1200 por su compatibilidad para conexión con Ethernet mediante protocolo Modbus. Demostrando que la implementación con estas aplicaciones supone un ahorro considerable respecto a dispositivos industriales.

- De acuerdo con el software y hardware utilizados para el diseño del sistema HMI, se determinó que la utilización de la librería QT multiplataforma permite desarrollar una GUI con elementos y widgets básicos; de esta manera se puede incorporar con facilidad botones, indicadores digitales o analógicos, luces piloto, potenciómetros, etc. A cada elemento del entorno gráfico se puede introducir una funcionalidad mediante programación y ejecutar en cualquier dispositivo e intercambiar datos en formato real con cualquier PLC compatible con el protocolo Modbus TCP/IP.

- A partir de la implementación del sistema HMI para controlar y monitorear la presión y temperatura de agua, se verificó que el monitoreo de datos de lectura se realiza en tiempo real en la pantalla HMI ya que el programa GUI se ejecuta cada 0,2 segundos. De acuerdo con las tablas presentadas en las pruebas de funcionamiento los datos de presión y temperatura de agua se presentan con precisión hasta dos decimales a diferencia de los sensores industriales que muestran valores en enteros. Sin embargo, la diferencia promedio entre valores de presión son 0,17 PSI y para la temperatura 0,1 grados Celsius demostrando que la lectura en el HMI tiene una exactitud considerable. Del mismo modo el envío de datos control para activar y desactivar las salidas del PLC se realizan con la misma rapidez con la que se ejecuta el programa.

- A partir del análisis de costo, se considera que la implementación del dispositivo HMI de código abierto para aplicaciones de investigación, suponen un ahorro del 16,6 % frente a dispositivos que cumplen con las funciones básicas de control. Y se tiene un ahorro del 44% frente a dispositivos de gama media, finalmente; se obtiene un ahorro de hasta 55% en comparación a paneles HMI avanzados usados para sistemas HMI/SCADA lo cual supone una excelente rentabilidad para su adquisición e implementación.

Recomendaciones

- Considerando el uso de tecnologías de código abierto, se puede encaminar a investigar la comunicación de Raspberry Pi con PLC´s de distintas marcas para determinar las ventajas y limitaciones de intercambiar datos con dispositivos industriales.

- En estudios futuros se podría analizar la factibilidad para la implementación de un Sistema Scada utilizando aplicaciones de código abierto, para incorporar alarmas, accesos de usuario, gráficas en tiempo real, etc.

- Se puede considerar el desarrollo e implementación de una Interfaz Gráfica de usuario, utilizando otra biblioteca gráfica dentro de Python como el Tkinter para analizar la funcionalidad y complejidad de programación respecto al PyQT.

REFERENCIAS

Acosta, K. (2017). Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo. (T. d. Rica, Ed.) doi:10.18845/tm.v30i5.3223

AENOR. (2016). AENOR Revista de la Normalización y Certificación, 60. Obtenido de http://bit.ly/36h0MPm

Ambriz, F., Arreguín , A., & Ledesma, R. (2014). Ciencias de la Ingeniería y Tecnología. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=4859806

Antúnez Soria, F. (2016). Puesta en marcha de sistemas de automatización industrial. Andalucia, España: IC Editorial. Obtenido de http://ebookcentral.proquest.com/lib/utnortesp/detail.action?docID=5635945

Arias , J., Jiménes , A., & Porras, H. (2016). DESARROLLO DE APLICACIONES EN PYTHON PARA EL APRENDIZAJE DE FÍSICA COMPUTACIONAL. Revista Ingeniería, Investigación y Desarrollo, 17.

Asenjo, R., González, S., Corbera, F., Navarro, Á., Rodríguez, A., Villalva, J., & Hendrix, E. (2017). La plataforma Raspberry Pi como base para la coordinación vertical. Dialnet, 16. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=6110758

Atenas, J., Havemann, L., & Priego, E. (Diciembre de 2015). Datos abiertos como recursos educativos abiertos: Hacia habilidades transversales y ciudadanía global. 7. doi:http://dx.doi.org/10.5944/openpraxis.7.4.233

AVEVA, G. (2018). InTouch HMI Getting Started Guide. Obtenido de https://sw.aveva.com/hubfs/assets-2018/pdf/Program-guide/ITGettingStartedGuide.pdf

Baret, M. (2019). Automatización Industrial en el siglo XXI. Rockwell Automation.

Behnam N. , T. (2009). La empresa en tiempo real: cómo transformar su organización para mejorar sus resultados. México: McGraw-Hill Interamericana. Obtenido de https://ebookcentral.proquest.com/lib/utnortesp/detail.action?docID=3191813&query=%20Ingl%C3%A9s%20%20Copiar%20automation%20and%20control%20systems

Benavides, J. (2015). DISEÑO DE UNA INTERFAZ HUMANO MÁQUINA PARA EL LABORATORIO DE CONTROL INDUSTRIAL E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO EN LA CARRERA DE INGENIERÍA EN MANTENIMIENTO ELÉCTRICO. Ibarra: Universodad Técnica del Norte. Obtenido de http://repositorio.utn.edu.ec/handle/123456789/4853

Carbonnelle , P. (2020). PYPL PopularitY of Programming Language. Obtenido de https://pypl.github.io/PYPL.html

Chakraborty, S. (2013). Verification of security intelligence for a resilient SCADA system. BEE (Jadavpur University),. Obtenido de http://93.174.95.29/_ads/5E813064D808A886138184BC12038FA3

code.visualstudio.com. (14 de mayo de 2021). visual studio code. Obtenido de https://code.visualstudio.com/docs

Cuevas , A. (2016). Python 3 Curso práctico. (R.-M. Editoria, Ed.) Obtenido de https://elibro.net/es/ereader/utnorte/106404

Dominguez, L., Vargas, J., & Velásquez, F. (2015). Diseño de una interfaz gráfica de usuario para el control de un prototipo de banda. Ingenium, 17(34). Obtenido de https://pdfs.semanticscholar.org/c830/33ee0b8f3eb93ac2d77d6be21dcbbd956444.pdf

Elecrow. (31 de agosto de 2016). 5INCH HDMI DISPLAY : MANUAL USER. Obtenido de https://www.elecrow.com/wiki/images/6/69/5inch_HDMI_Display_User_Manual.pdf

Fernández, M., & Algar, J. (2019). Introducción práctica a la programación con Python. (U. d. Alcalá, Ed.) Obtenido de https://elibro.net/es/ereader/utnorte/124259

Filali, S., Carina S. , G., & Lecuona, C. (2015). Estándares HMI/SCADA para el diseño de interfaces en los centros de datos: El centro de control y operaciones como caso de estudio. Obtenido de https://www.redalyc.org/articulo.oa?id=49642141023

Gutiérrez, Á. (2015). Python paso a paso. (RA-MA, Ed.) Madrid. Obtenido de https://elibro.net/es/ereader/utnorte/107213?page=24

Habana, C. (2016). Las normas ISO 9000. 11(2). Obtenido de http://scielo.sld.cu/pdf/cofin/v10n2/cofin02216.pdf

Imbaquingo , D., Granda , P., Ortega, C., Guevara, C., & Pusdá, M. (2016). Innovación Tecnológica. Ibarra, Imbabura, Ecuador: Universidad Técnica del Norte. Facultad de Ingeniería en Cienias Aplicadas. Obtenido de http://bit.ly/36mlnBY

Knapp, E. (2011). Industrial Network Security. U.S.A: Elsevier. Obtenido de http://93.174.95.29/_ads/4D03C682B147C3F5A8606EF7DC293938

Laime, E., & Almidón, Á. (2018). Manual de programación LabVIEW 9.0. Obtenido de https://www.researchgate.net/publication/328404013_Manual_de_programacion_LabVIEW_90

Linares Gonzales, V. (2015). UF:2238 Diagnosis de averías y mantenimiento correctivo de istemas de automatización industrial. Málaga: IC Editorial. Obtenido de https://elibro.net/es/ereader/utnorte/59237

Maccrady, S. (2013). Designing SCADA Application Software. Canadá: Elsevier. Obtenido de http://93.174.95.29/_ads/9FF8123525CCDACB35AE5C0D88C3B1A4

Matango, D., & Portilla, F. (2016). DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO PARA EL MONITOREO Y CONTROL AUTOMÁTICO DE PRESIÓN Y

TEMPERATURA DE AGUA. Ibarra: Universidad Técnica del Norte. Obtenido de http://repositorio.utn.edu.ec/handle/123456789/7568

Medina , B., Castro, S., & Camargo , L. (2015). TECNOLOGÍAS DE CÓDIGO ABIERTO PARA LA GESTIÓN DE UN PROCESO INDUSTRIAL. Revista Gerencia Tecnológica. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=5161783

Merchant, H. K., & Ahire, D. D. (2017). Industrial Automation using IoT with Raspberry Pi. International Journal of Computer Applications. Obtenido de http://eiu.thaieei.com/box/Technology/65/Industrial%20Automation%20using%20IoT%20with%20Raspberry%20Pi%20.pdf

Pérez, E. (octubre de 2015). Los sistemas SCADA en la automatización industrial. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=5280242

Portalo, J., Gonzáles , I., Manuel, C., & Antonio, C. (2019). Comparativa de entornos Open-Source para sistemas de supervisión aplicables a Smart Grids/Smart Micro-Grids. (U. d. Extremadura, Ed.) Repositorio Universidad de Coruña, 9. Obtenido de https://ruc.udc.es/dspace/bitstream/handle/2183/23752/2019_Portalo-Calero_Comparativa-entornos-open-source-sistemas-supervision-smart-grids.pdf?sequence=3&isAllowed=y

Raspberrypi.org. (2017). Raspberry pi 3 model B. Obtenido de https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

Rodriguez, A. (2013). Sistemas SCADA (Vol. 3). México: Alfaomega Grupo Editor. Obtenido de http://masserv.utcluj.ro/~florind/cursuri/Manuale/SCADA/Sistemas%20SCADA%203ed%20-%20Rodriguez%202013.pdf

Rozo, J. (2014). Metodología de Desarrollo de Software. INGENIARE. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=5980502

Ruiz, E., Chung, A., & Inche, J. (2008). Desarrollo de una interfaz hombre máquina orientada al control de procesos. Industrial Data Revista de Investigación , 11. Recuperado el 2019, de http://www.redalyc.org/articulo.oa?id=81611211010

Salcedo, M. (2016). USO DEL MINICOMPUTADOR DE BAJO COSTO “RASPBERRY PI” EN ESTACIONES METEOROLÓGICAS. Revista Eléctronica de Estudios Telemáticos, 15(1), 24. Obtenido de http://www.redalyc.org/articulo.oa?id=78445977004

Saquicuya, H. (2019). Guía para el diseño de HMI. Recuperado el 2019, de https://es.scribd.com/document/56390297/Diseno-de-HMI

Schneider Electric. (2010). DeviceNet communication manual.

SENA. (8 de Abril de 2016). Diseño de Sistemas HMI. (R. Espinosa, Ed.) Colombia. Obtenido de http://bit.ly/2Ro4D92

Siemens. (08 de 2018). Controlador programable S7-1200: MANUAL DE SISTEMA.

Terán , R. (2015). IMPLEMENTACIÓN DE UNA RED DE COMUNICACIÓN INDUSTRIAL PARA EL MÓDULO DIDÁCTICO DEL PLC S7-1200 PARA EL LABORATORIO DE LA CARRERA DE INGENIERÍA EN MANTENIMIENTO ELÉCTRICO (Trabajo de Pregrado). Ibarra: Universodad Técnica del Norte.

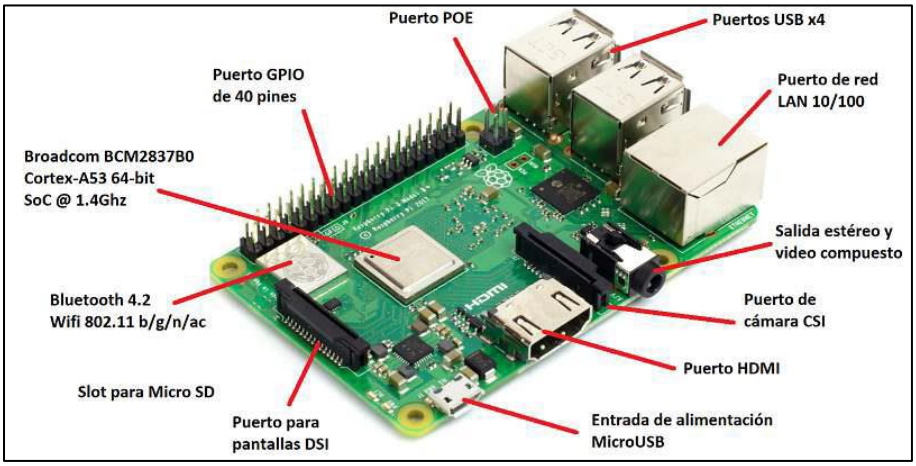
UNE-EN ISO 9241-110. (2006). Ergonomía de interacción persona-sistema. Parte 110. Obtenido de https://www.sis.se/api/document/preview/907276/

Vaca, E. (2019). Implementación de un sistema scada mediante el software intouch para el control y visualización de procesos industriales (trabajo de Pregrado). Universidad Técnica del Norte, Ibarra. Obtenido de http://repositorio.utn.edu.ec/handle/123456789/8916

ANEXOS

ANEXO A: Características de Raspberry Pi 3 Modelo B+

A.1. Puertos de conexión



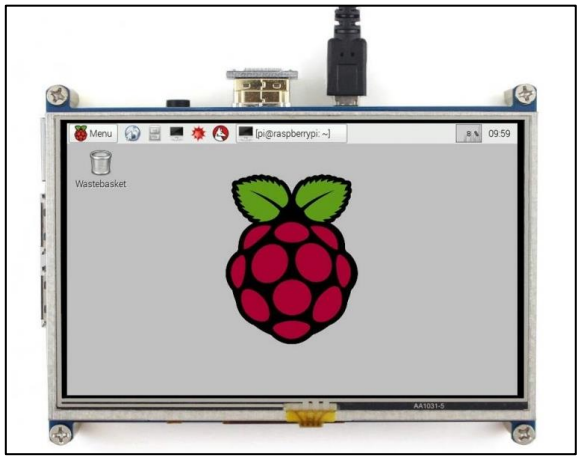
A.2. Especificaciones técnicas

Procesador	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
	Bluetooth 4.1 (Bluetooth Classic and LE)
Memoria RAM	1GB LPDDR2
Conectividad	2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac Wireless LAN, Bluetooth 4.2, BLE Gigabit Ethernet sobre USB 2.0 (rendimiento máximo de 300 Mbps) 4 × Puertos USB 2.0
Sistema Operativo	Arranca desde la tarjeta Micro SD, ejecutando una versión del sistema operativo Linux o Windows 10 IoT
Dimensiones	85 x 56 x 17mm
Alimentación	Micro USB socket 5V1, 2.5A
Video y sonido	Puerto 1 x HDMI, puerto de pantalla DSI, MIPI CSI para módulo de cámara. Salida de audio estéreo y puerto de video compuesto

Notas:

- Este producto solo debe conectarse a una fuente de alimentación externa DC de 5 V y 2,5A.
- Este producto debe operarse en un ambiente bien ventilado y, si se usa dentro de un gabinete, el mismo no debe estar cubierto por completo.

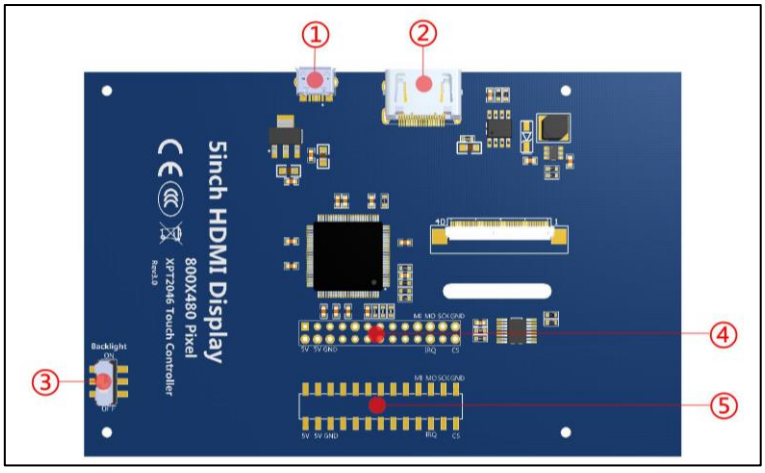
ANEXO B: Manual de usuario de pantalla HDMI 5 pulgadas



B.1. Características Display

Característica	Descripción
Tamaño	5 pulgadas
Resolución	800 x 480 pixeles
Pantalla	touch capacitivo
Luz fondo	Admite el control de la luz de fondo, se puede apagar para ahorrar energía
Interfaz	Entrada HDMI compatible con Raspberry Pi (3ra, 2da generación)
Soporte	Funcional con sistema operativo Windows, Linux, Raspbian, Ubuntu
Dimensiones	121.11 x 77.93(mm)

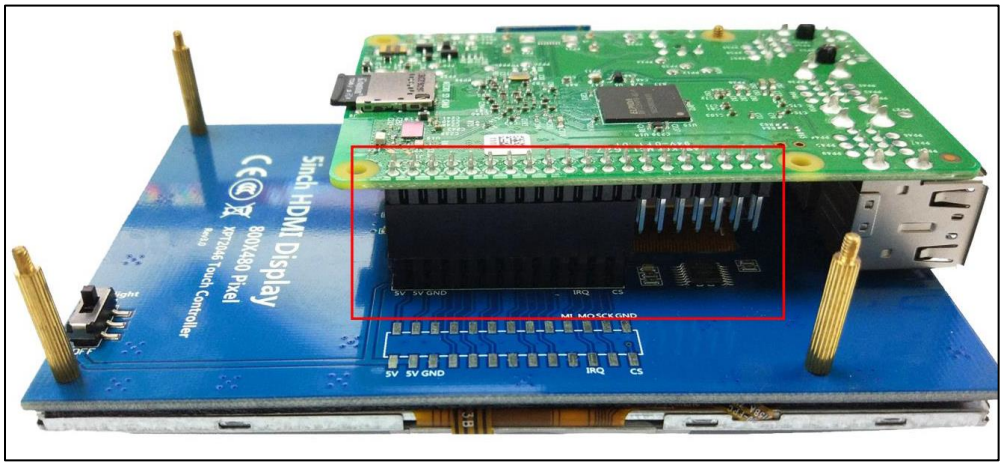
B.2. Descripción de hardware (vista trasera)



Partes de la pantalla

Elemento	Nombre
1	Entrada micro USB para alimentación 5v dc
2	Entrada HDMI
3	Interruptor: control luz de fondo
4	Pines 13 x 2 transmisión señal táctil
5	Pines para expansión

B.3. Modo de conexión con la tarjeta Raspberry Pi



B.4. Instalación controladora táctil

La Raspberry debe estar conectado a una red de internet. Se ejecuta las siguientes líneas de comando en la terminal de Raspberry Pi, para descargar controlador actualizado.

Comando	Descripción
sudo rm -rf LCD-show git clone https://github.com/goodtft/LCD-show.git chmod -R 755 LCD-show	Comandos para acceder y descargar controlador actualizado de internet.
cd LCD-show/	controlador para Display de 5 pulgadas capacitivo

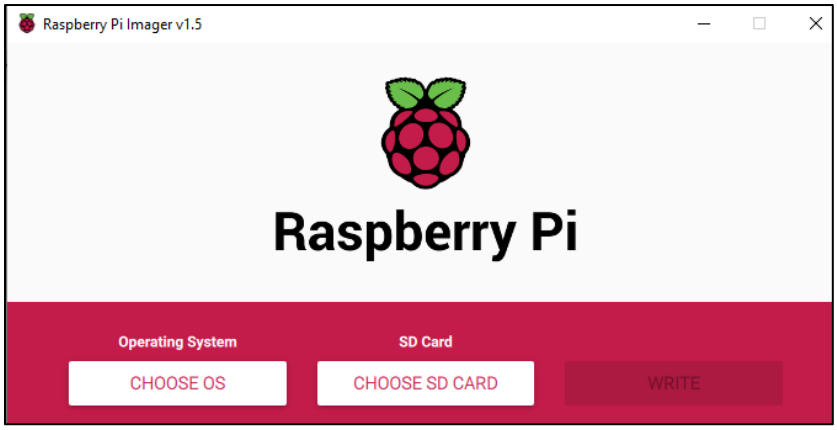
Luego de ejecutar todos los comandos se debe reiniciar el dispositivo para finalizar el proceso de instalación del controlador táctil.

ANEXO C: Configuraciones de Raspberry

Los pasos para la utilización del dispositivo se detallan a continuación.

a) Descargar el sistema operativo Raspberry pi Os de la página oficial

b) Montar la imagen ISO en una tarje microSD de clase 10, con capacidad mínima de 8 GB, mediante software de instalación Raspberry pi imager.



c) Insertar la tarjeta SD en la ranura de Raspberry. Es necesario incorporar un mouse y un teclado para los siguientes pasos.

d) Configurar idioma y conexión a una red mediante wifi o ethernet

e) Ejecutar los comandos en la terminal de Raspberry de acuerdo con las funciones que se requiera según la Tabla X.

Comando general	**Función**
sudo apt-get update	Buscar actualizaciones del sistema
sudo apt-get upgrade	Instalar actualizaciones
sudo raspi-config	Panel de configuraciones
Comando instalación de librerías	
pip install matplotlib	Instalación de librería para gráficas

ANEXO D: Programación en Python

```
#%%
from pyModbusTCP.client import ModbusClient
from pyModbusTCP import utils
# ____________________________________________________
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtWidgets import QLCDNumber, QDialog, QApplication, QGraphicsPixmapItem, QGrap
hicsScene, QDial
from PyQt5 import QtWidgets
from PyQt5 import QtCore, QtGui
from PyQt5.QtGui import QPen, QPixmap
from PyQt5.QtCore import Qt
from threading import Event
from PyQt5.QtCore import pyqtSlot
from PyQt5.QtCore import pyqtSignal
from PyQt5.QtCore import QThread
from COMPLETO import *
import struct
import time
import threading as th
import sys
import serial
import datetime

ICON_RED_LED = "imagenes/led-red-on.png"
ICON_GREEN_LED = "imagenes/green-led-on.png"

# Datos de escritura
global dato1, dato2,dato3,dato4,dato5,dato6,dato7,dato8,dato9,dato10,dato11
dato1 = 0.0  # act_niquelina DB 3 2
dato2 = 1.0   # dsct_niquelida DB 3
dato3 = 0.0   # act_variador DB1
dato4 = 1.0 # dsct_variador
dato5 = 0.0  #act_solenoide1
dato6= 1.0  # desct_solenoide1
dato7= 0.0  # act_solenoide 2
dato8 = 1.0 # dsct_solenoide2
dato9 = 0.0 #act _solenoide3
dato10= 1.0  # dsct_solenoide3
dato11=0.0  #envio_frecuencia

# Datos de lectura....
global dato12,dato13,dato14,dato15,dato16,dato17,dato18, dato19, dato20, dato21, dato22

dato12 = 0.0  # iluminacion niquelina
dato13 = 0.0    #iluminacion variador
dato14 = 0.0   #iluminacion solenoide 1
dato15 = 0.0  #iluminacion solenoide 2
dato16= 0.0  # solenoide3
dato17= 0.0   # Temperatura grados centigrados
dato18= 0.0   # temperatura Farenheit
dato19= 0.0 # sensor de presion
dato20= 0.0  # Sensor tanque bajo
dato21= 0.0 # Sensor tanque alto
dato22= 0.0 # estado de selector
global datos
datos = []

# Comunicación Modbus TCP
```

```
def funcion1():
    global dato1
    global dato2
    global dato3
    global dato4
    global dato5
    global dato6
    global dato7
    global dato8
    global dato9
    global dato10
    global dato11 #_____________________________ variador de frecuecnia

    global dato12
    global dato13
    global dato14
    global dato15
    global dato16
    global dato17 #_____________________________selector I0.7 PLC
    global dato18
    global dato19
    global dato20
    global dato21
    global dato22

    # Para leer datos de plc a raspberry  Hilo para comunciacion
    class FloatModbusClient(ModbusClient):
        def read_float(self, address, number=1):
            reg_l = self.read_holding_registers(address, number * 2)
            if reg_l:
                return [utils.decode_ieee(f) for f in utils.word_list_to_long(reg_l)]
            else:
                return None

    # Sirve para escribir datos de la raspberry al plc
        def write_float(self, address, floats_list):
            b32_l = [utils.encode_ieee(f) for f in floats_list]
            b16_l = utils.long_list_to_word(b32_l)
            return self.write_multiple_registers(address, b16_l)

    #host direccion IP de PLC y puerto de comunciacion Modbus 502 por defecto
    c = FloatModbusClient(host='192.168.1.5', port=502, auto_open=True)

    # write 10.0 at @0  Escribir de la Raspberry al PLC
    c.write_float(0, [dato1]) #niquelina
    c.write_float(2, [dato2])
    c.write_float(4, [dato3]) #variador
    c.write_float(6, [dato4])
    c.write_float(8, [dato5]) #solenoide 1
    c.write_float(10,[dato6])
    c.write_float(12, [dato7]) #solenoide 2
    c.write_float(14, [dato8])
    c.write_float(16, [dato9]) #solenoide 3
    c.write_float(18, [dato10])
    c.write_float(20, [dato11]) #envio valor de frecuencia con el dial entre 0-60 Hz
    #c.write_float(10,[dato12])
    #niquelina
    #variador
    #sole1
    #sole2
```

```
    #sole3
    # envio de frecuencia

    #dato17= round(dato17[0], 2)
    # read @0 to 9   Del PLC a la Raspberry
    dato12 = c.read_float(22, 1)
    dato12 = round(dato12[0], 2)
    dato13 = c.read_float(24, 1)
    dato13 = round(dato13[0], 2)
    dato14 = c.read_float(26, 1)
    dato14 = round(dato14[0], 2)
    dato15 = c.read_float(28, 1)
    dato15 = round(dato15[0], 2)
    dato16 = c.read_float(30, 1)
    dato16 = round(dato16[0], 2)
    dato17 = c.read_float(32, 1)
    dato17 = round(dato17[0], 2)
    dato18 = c.read_float(34, 1)
    dato18 = round(dato18[0], 2)
    dato19 = c.read_float(36, 1)
    dato19 = round(dato19[0], 2)
    dato20 = c.read_float(38, 1) #sensor de tanque HIGH
    dato20 = round(dato20[0], 2)
    dato21 = c.read_float(40, 1) # 1 dato de la direccion 30  SELECTOR
    dato21 = round(dato21[0], 2) # con 2 decimales del numero flotante
    dato22 = c.read_float(42, 1)
    dato22 = round(dato22[0], 2)


    datos = [dato12,dato13,dato14,dato15,dato16,dato17,dato18, dato19, dato20, dato21,dat
o22]
    datosE= [dato1,dato2,dato3,dato4,dato5,dato6,dato7, dato8, dato9, dato10,dato11,]
    print(datos)
    print(datosE)
    print(dato9,dato10)

    time.sleep(0.1) # 0.2 anterior
    t = th.Thread(target=funcion1)
    t.start()
t = th.Thread(target=funcion1)
t.start()

# inicio de programación en pyQT
while True:
    class proyecto(QDialog):
        def __init__(self):
            super().__init__()
            self.dialogo = Ui_MODULO_PRESIN_TEMPERATURA()
            self.dialogo.setupUi(self)
            self.strip = 0
            pen = QPen(Qt.red)  # color de la aguja
            pen2 = QPen(Qt.green)
            pen.setWidth(4)  # grosor de la aguja
            pen2.setWidth(4)
            pen.setCapStyle(Qt.RoundCap)  # borde redondeado de la aguja
            pen2.setCapStyle(Qt.RoundCap)
            scene = QtWidgets.QGraphicsScene()
            scene2 = QtWidgets.QGraphicsScene()
            pen.setCosmetic(True)
            pen2.setCosmetic(True)
```

```
            scene.addPixmap(QPixmap('./visor_temperatura.png'))
            scene2.addPixmap(QPixmap('./visor_presión.png'))
            # angulo de apertura y cierre
            self.item = scene.addLine(60, 170, 97, 97, pen)
            self.item2 = scene2.addLine(60, 170, 97, 97, pen2)
            pen = QtGui.QPen(QtGui.QColor(QtCore.Qt.gray))
            pen2 = QtGui.QPen(QtGui.QColor(QtCore.Qt.gray))
            brush = QtGui.QBrush(pen.color().darker(100))
            brush2 = QtGui.QBrush(pen2.color().darker(100))
            scene.addEllipse(87, 87, 20, 20, pen, brush)  # centro del grafico
            scene2.addEllipse(87, 87, 20, 20, pen2, brush2)
            self.item.setTransformOriginPoint(97, 97)  # centro del eje de la aguja
            self.item2.setTransformOriginPoint(97, 97)
            self.dialogo.Grafik.setScene(scene) #temperatura
            self.dialogo.Grafik2.setScene(scene2) # presión

            self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_RED_LED))   # ventana co
municacion
            self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.BS_TANQUE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.BS_TANQUE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))


            # Botones activados-desactivados_COMUNICACION
            self.dialogo.Q1ON.clicked.connect(self.fn_actNiquelina)
            self.dialogo.Q1OFF.clicked.connect(self.fn_desactNiquelina)
            self.dialogo.Q2ON.clicked.connect(self.fn_actVariador)
            self.dialogo.Q2OFF.clicked.connect(self.fn_desactVariador)
            self.dialogo.Q3ON.clicked.connect(self.fn_actSolenoide1)
            self.dialogo.Q3OFF.clicked.connect(self.fn_desactSolenoide1)
            self.dialogo.Q4ON.clicked.connect(self.fn_actSolenoide2)
            self.dialogo.Q4OFF.clicked.connect(self.fn_desactSolenoide2)
            self.dialogo.Q5ON.clicked.connect(self.fn_actSolenoide3)
            self.dialogo.Q5OFF.clicked.connect(self.fn_desactSolenoide3)
            self.dialogo.dial_frecuencia.valueChanged.connect(self.fn_frecuencia) # dial
frecuencia

            self.stop_flag_RS232 = Event()
            self.getanalogico = ControlRaspberry(self.stop_flag_RS232)
            self.getanalogico.newValue.connect(self.temperatura)  # Valor Temperatura
            self.getanalogico.newValue2.connect(self.presion)  # pValor Presion
            self.getanalogico.newValue3.connect(self.Entrada_Digitales)  # pValor Presion
            self.getanalogico.start()

        def fn_actNiquelina(self):
            global dato1,dato2
            dato1 = 1.0
            dato2 = 0.0
            self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_GREEN_LED)) # ventana de
 comunicación / control

        def fn_desactNiquelina(self):
            global dato1,dato2
            dato1 = 0.0
            dato2 = 1.0
            self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_RED_LED))
```

```
        def fn_actVariador(self):
            global dato3,dato4,dato5,dato7  # sole1 , sole 2

            if dato5==1.0 or dato7==1.0:
                dato3 = 1.0
                dato4 = 0.0
                self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

        def fn_desactVariador(self):
            global dato3,dato4    #variador
            dato3 = 0.0
            dato4= 1.0
            self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        def fn_actSolenoide1(self):
            global dato5,dato6   # solenoide 1
            dato5 = 1.0
            dato6 = 0.0
            self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

        def fn_desactSolenoide1(self):
            global dato5,dato6   # solenoide 1
            dato5 = 0.0
            dato6 = 1.0
            self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        def fn_actSolenoide2(self):
            global dato7,dato8   # solenoide 2
            dato7 = 1.0
            dato8 = 0.0
            self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

        def fn_desactSolenoide2(self):
            global dato7,dato8   # solenoide 2
            dato7 = 0.0
            dato8 = 1.0
            self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        def fn_actSolenoide3(self):
            global dato9,dato10   # solenoide 3
            dato9 = 1.0
            dato10 = 0.0
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

        def fn_desactSolenoide3(self):
            global dato9,dato10   # solenoide 3
            dato9 = 0.0
            dato10 = 1.0
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        def fn_frecuencia(self):
            global dato11
            getValue= self.dialogo.dial_frecuencia.value()
            dato11=getValue


        #lectura de planta , monitoreo remoto
        def Entrada_Digitales(self, niquel,variador,sole1,sole2,sole3,low,high,selector):
            global dato1, dato2
            if niquel == 0.0 and selector==0.0 : # esta en modo local
```

```
                self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_RED_LED))

                dato1 = 0.0
                dato2 = 1.0
            if niquel== 1.0 and selector== 0.0 :
                dato1=1.0
                dato2=0.0

            if niquel == 1.0:
                self.dialogo.BNIQUELITA.setPixmap(QtGui.QPixmap(ICON_GREEN_LED)) #ventaan
de monitoreo PLANTA
                self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            else:
                self.dialogo.BNIQUELITA.setPixmap(QtGui.QPixmap(ICON_RED_LED))
                self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_RED_LED))

            if variador == 0.0 and selector==0.0 :
                global dato3,dato4
                dato3 = 0.0
                dato4= 1.0
                self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))

            if variador == 1.0 and selector==0.0 :
                dato3 = 1.0
                dato4= 0.0
                self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

            if variador ==1.0:
                self.dialogo.B_VARIADOR.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
                self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            else:
                self.dialogo.B_VARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))
                self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))

            if sole1 == 0.0 and selector==0.0 : # cambio de hmi a local del mosulo
                global dato5, dato6
                self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))
                dato5 = 0.0
                dato6 = 1.0

            if sole1== 1.0 and selector== 0.0 :
                dato5=1.0
                dato6=0.0
            if sole1 == 1.0:
                self.dialogo.BSOLONOIDE1.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
                self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

            else:
                self.dialogo.BSOLONOIDE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))
                self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))

            if sole2 == 0.0 and selector==0.0 : #test sergio 22 julio
                self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))
                global dato7, dato8
                dato7 = 0.0
                dato8 = 1.0
            if sole2== 1.0 and selector== 0.0 :
                dato7 = 1.0
                dato8 = 0.0
            if sole2== 1.0:
```

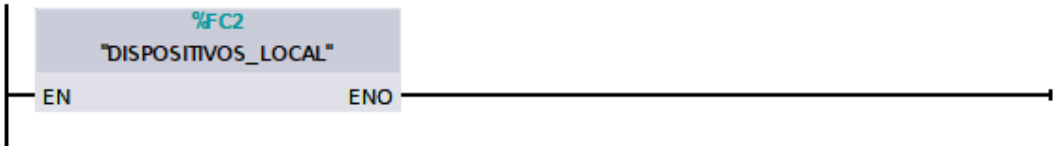
```
            self.dialogo.BSOLENOIDE2.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
        else:
            self.dialogo.BSOLENOIDE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        if sole3 == 0.0 and selector==0.0 : #test sergio 22 julio
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            global dato9, dato10
            dato9 = 0.0
            dato10 = 1.0
        if sole3== 1.0 : #and selector== 0.0 :
            dato9 = 1.0
            dato10 = 0.0
            self.dialogo.BSOLENOIDE3.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))

        if sole3 == 1.0:
            self.dialogo.BSOLENOIDE3.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
        else:
            self.dialogo.BSOLENOIDE3.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        if low == 1.0:
            self.dialogo.BS_TANQUE2.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
        else:
            dato2 = 0.0
            self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))


        if high == 1.0:
            self.dialogo.BS_TANQUE1.setPixmap(QtGui.QPixmap(ICON_GREEN_LED))
            dato2 = 0.0
            self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))
        else:
            self.dialogo.BS_TANQUE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))

        # Lectura de la posición del selector  la entrada I0.7 del PLC
        #if selector ==0:
            #dato1 =0.0
            #dato2 =0.0
            #dato4 =0.0
            #dato5 =0.0

            #self.dialogo.SSELENOIDE3.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            #self.dialogo.SSELENOIDE2.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            #self.dialogo.SSELENOIDE1.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            #self.dialogo.SVARIADOR.setPixmap(QtGui.QPixmap(ICON_RED_LED))
            #self.dialogo.SNIQUELITA.setPixmap(QtGui.QPixmap(ICON_RED_LED))


    def temperatura(self, poti1, rotacion1):    # lectura sensor 1
        self.dialogo.EANT1.display(poti1)
        self.item.setRotation(rotacion1)

    def presion(self, poti2, rotacion2):    # lectura sensor 1
        self.dialogo.EANT2.display(poti2)
        self.item2.setRotation(rotacion2)
        self.show()
```

```
    #guardar la lectura del estado de variables del modulo leido por Modbus
    class ControlRaspberry(QThread):
        newValue = pyqtSignal(object, object)
        newValue2 = pyqtSignal(object, object)
        newValue3 = pyqtSignal(object,object,object,object,object,object,object,object,ob
ject)

        def __init__(self, event):
            QThread.__init__(self)
            self.stopped = event
            self.altValue = 0

        def run(self):

            while not self.stopped.wait(0.03):  # 1 max 0.3
                self.ArduinoLoop()

        def ArduinoLoop(self):
            #variablers de lectura
            global dato12
            global dato13
            global dato14
            global dato15
            global dato16
            global dato17
            global dato18
            global dato19
            global dato20
            global dato21
            global dato22

            value = int(26 + (dato17* 2.5))  # Sensor Temperatura
            value2 = int(28+(dato19 * 25)) # Sensor Presion
            self.newValue.emit(dato17, value)
            self.newValue2.emit(dato19, value2)
            self.newValue3.emit(dato12,dato13,dato14,dato15,dato16,dato19,dato20,dato21,
dato22)  #var desde PLC

    if __name__ == '__main__':
        app = QApplication(sys.argv)
        dialogo = proyecto()
        dialogo.show()
        sys.exit(app.exec_())
```

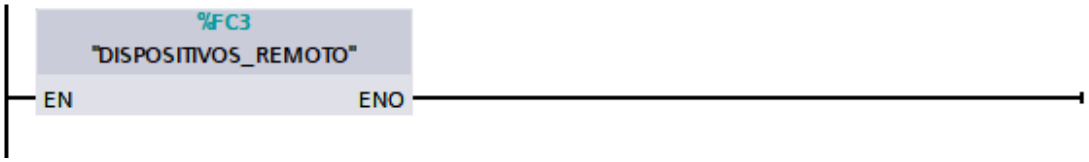
ANEXO E: Programación del PLC en Tia Portal

Bloques de programa: Main [OB1]

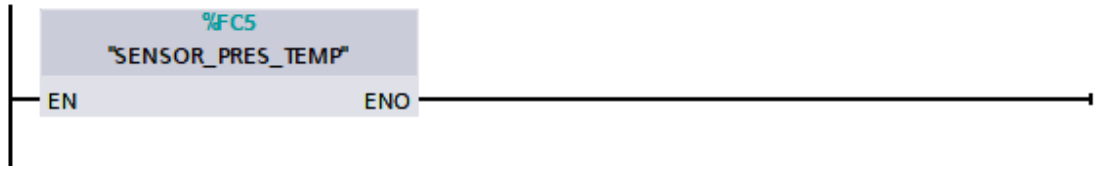
Segmento 1: Bloque de salida modo local



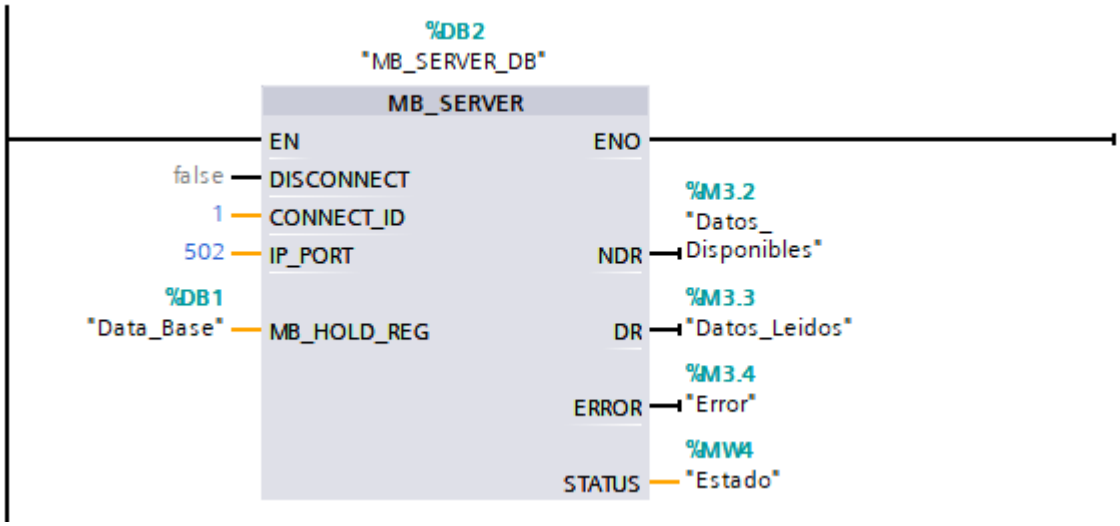
Segmento 2: Bloque de salida modo remoto HMI



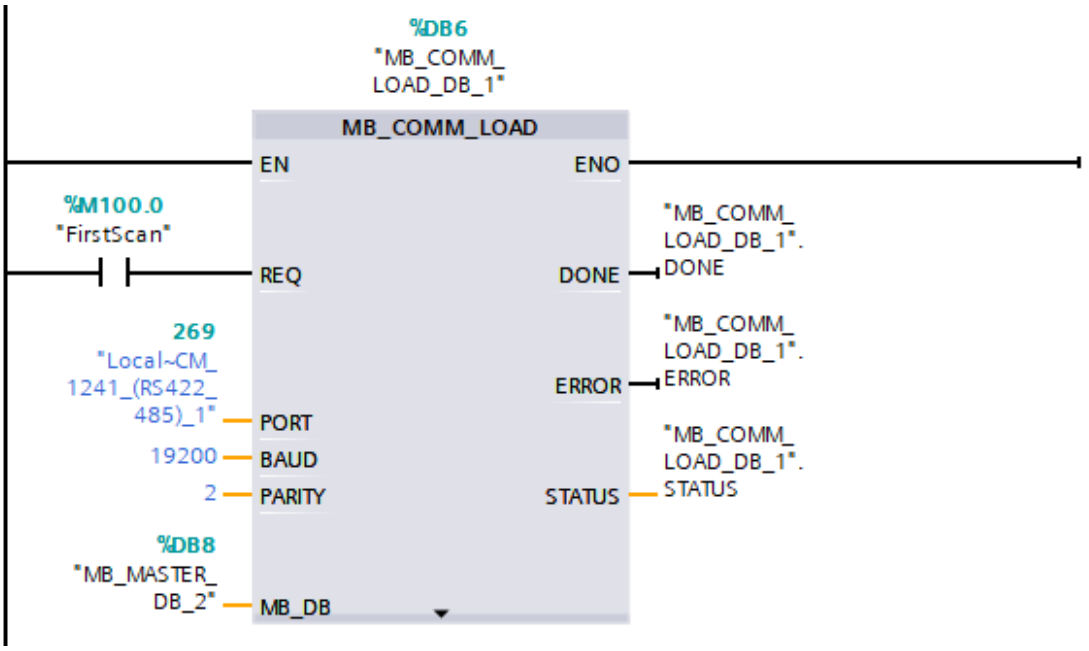
Segmento 3: Bloque de salida sensores



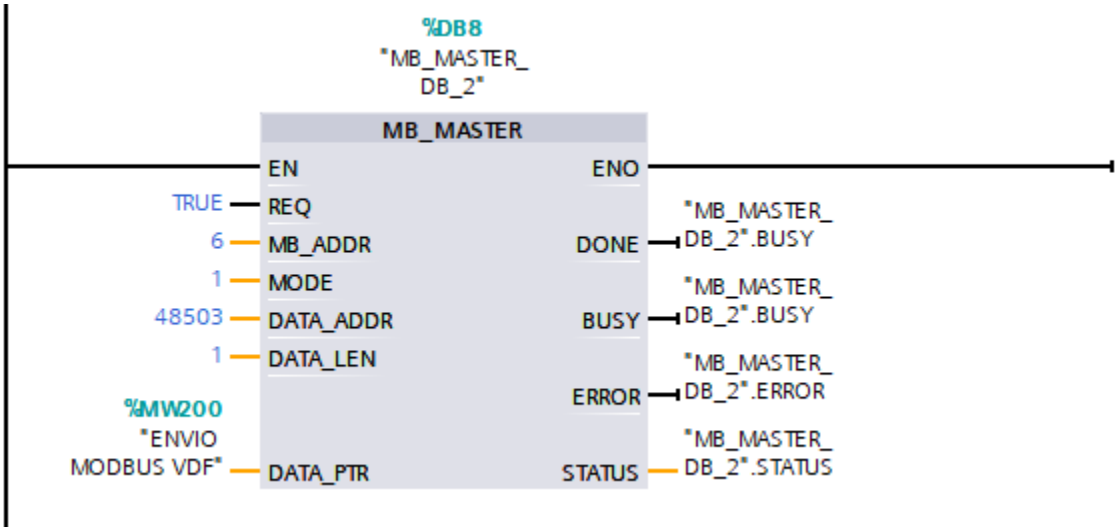
Segmento 4: Bloque de comunicación Modbus Server para Raspberry Pi



Segmento 5: Bloque para configuración de puerto de comunicación entre PLC y variador

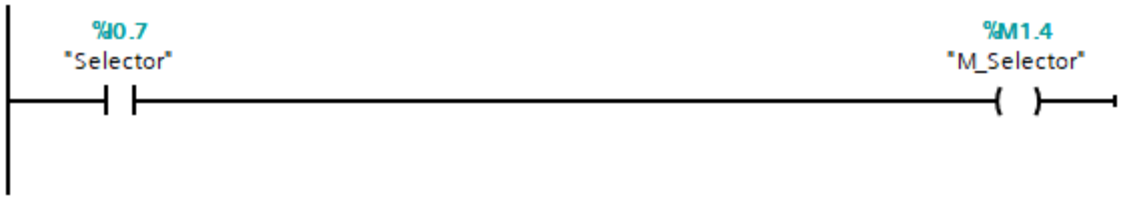


Segmento 6: Bloque de comunicación PLC y Variador de frecuencia

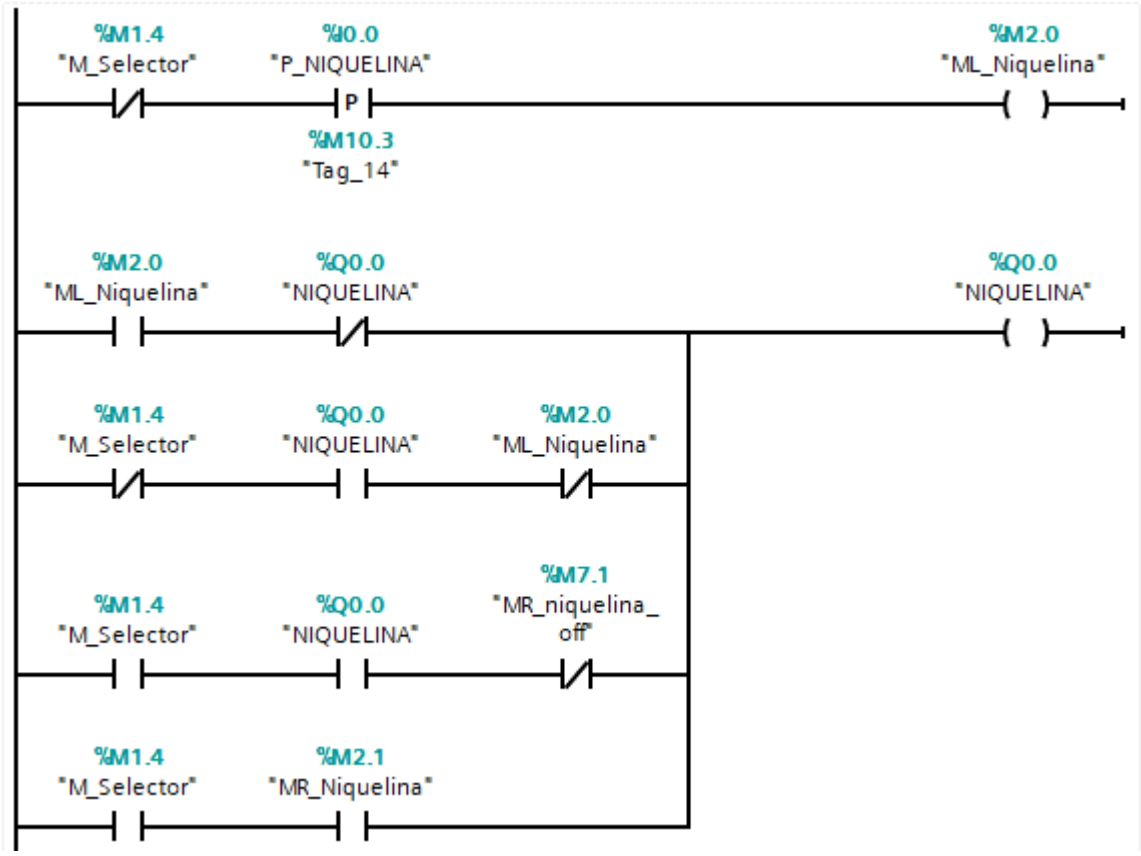


Bloques de programa: DISPOSITIVOS_LOCAL [FC2]

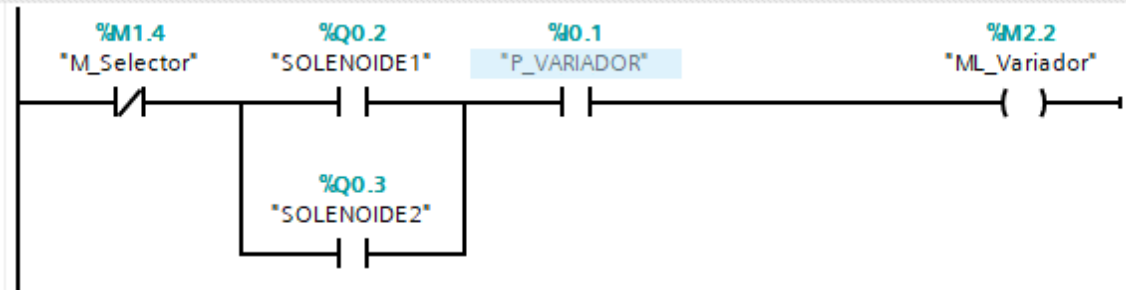
Segmento 1: Lectura estado de selector

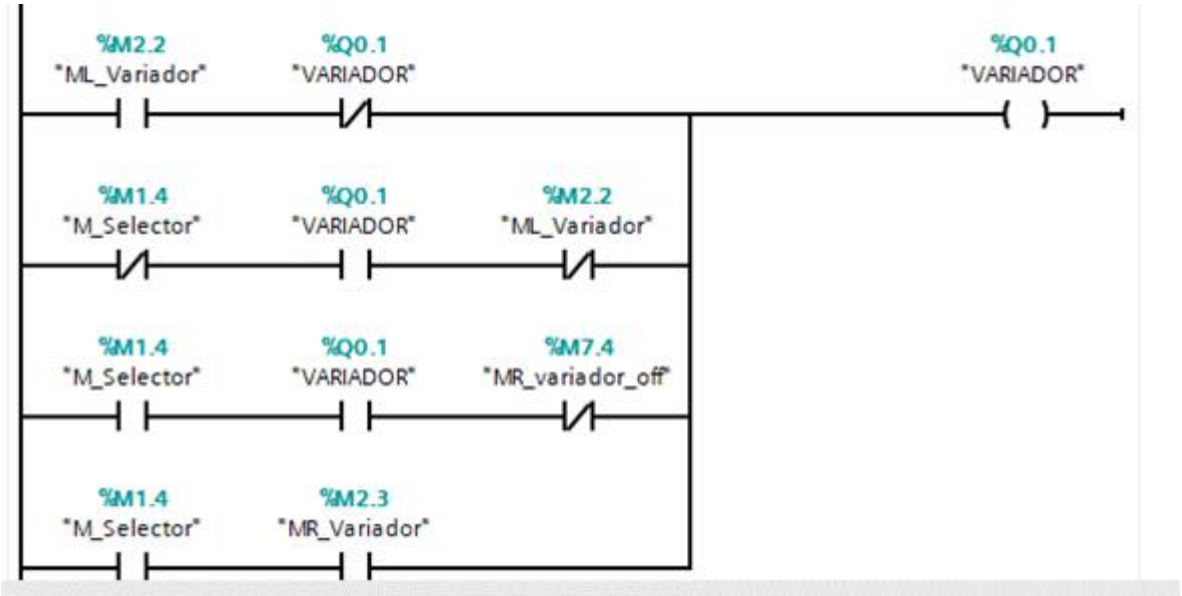


Segmento 2: Activar y desactivar niquelina modo local y HMI

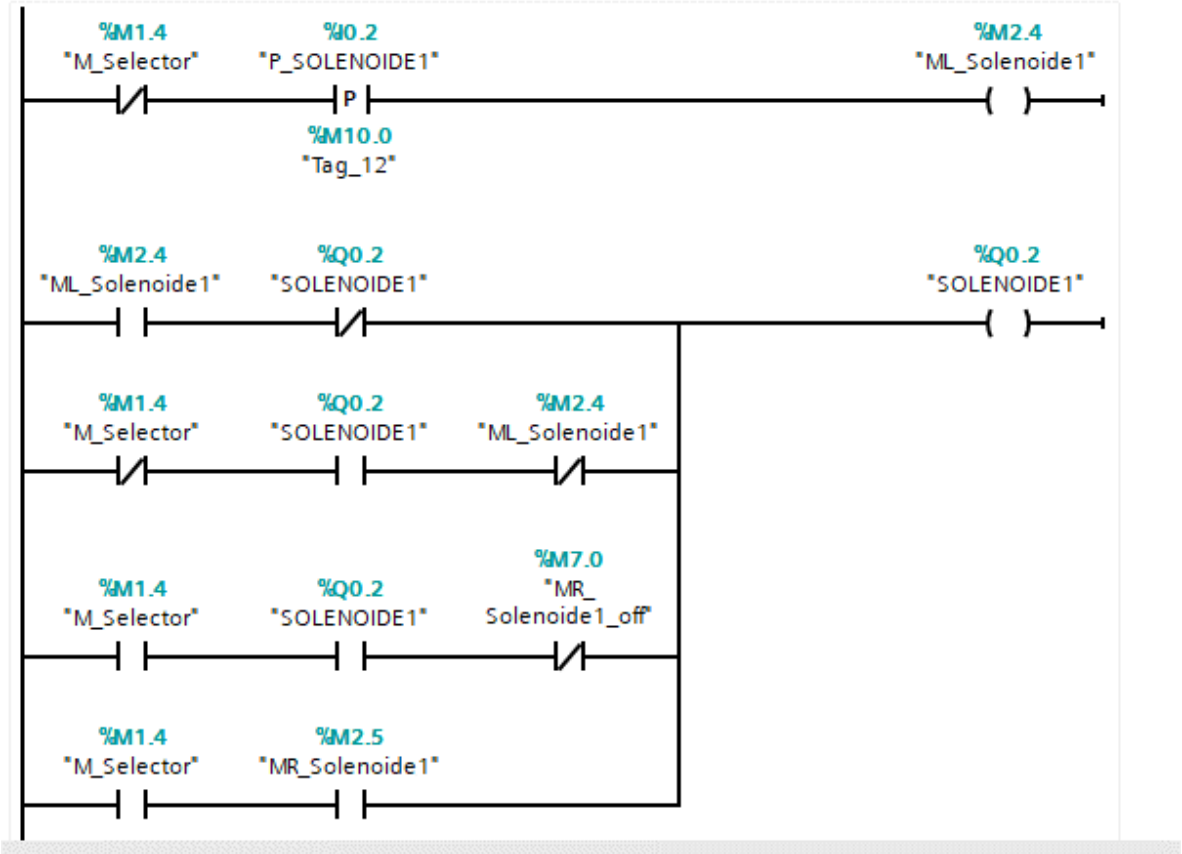


Segmento 3: Activar y desactivar variador de frecuencia modo local

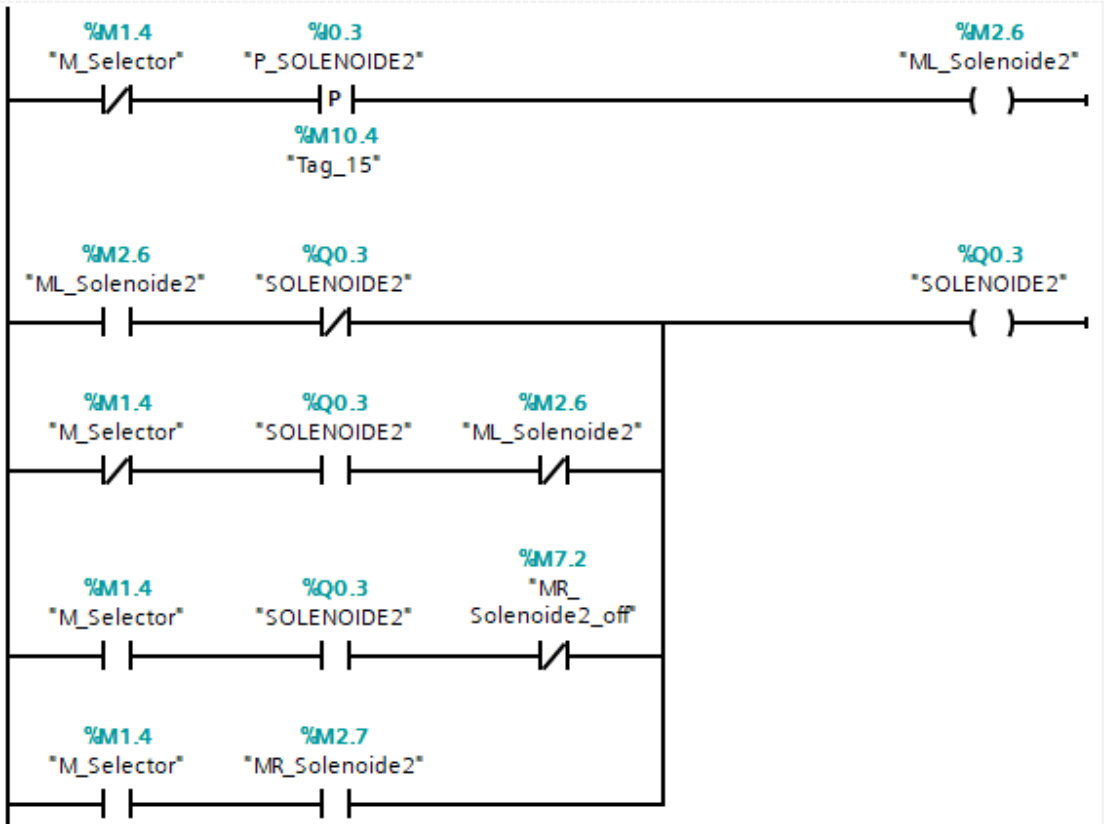




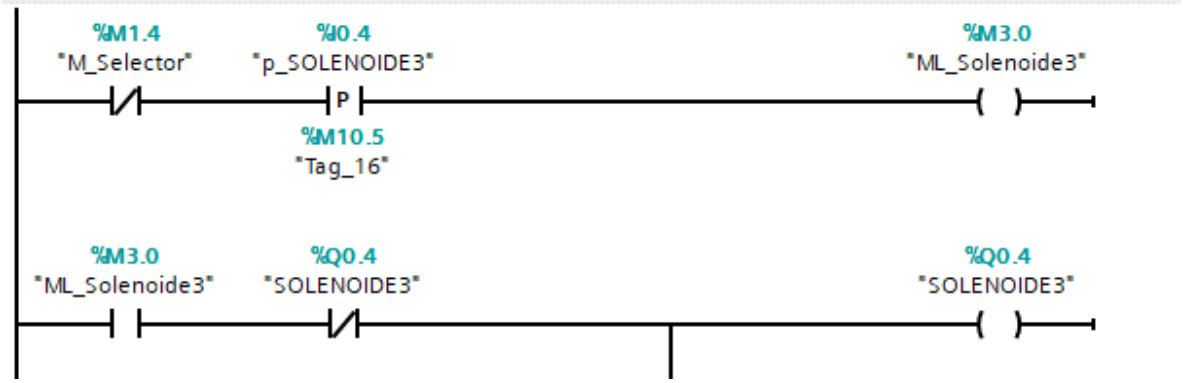
Segmento 4: Activar y desactivar solenoide 1 modo local

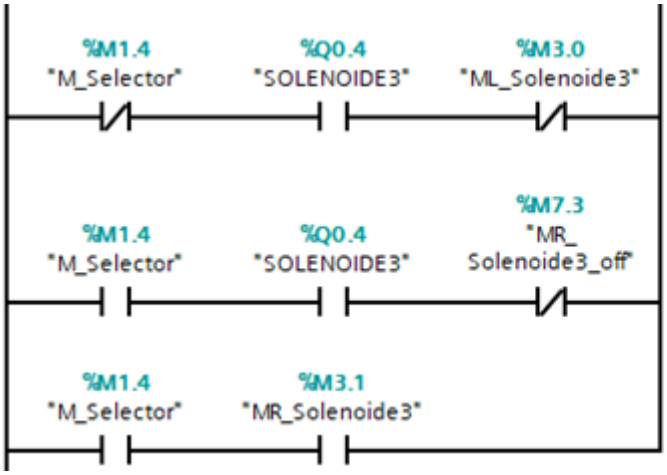


Segmento 5: Activar y desactivar solenoide 2 modo local

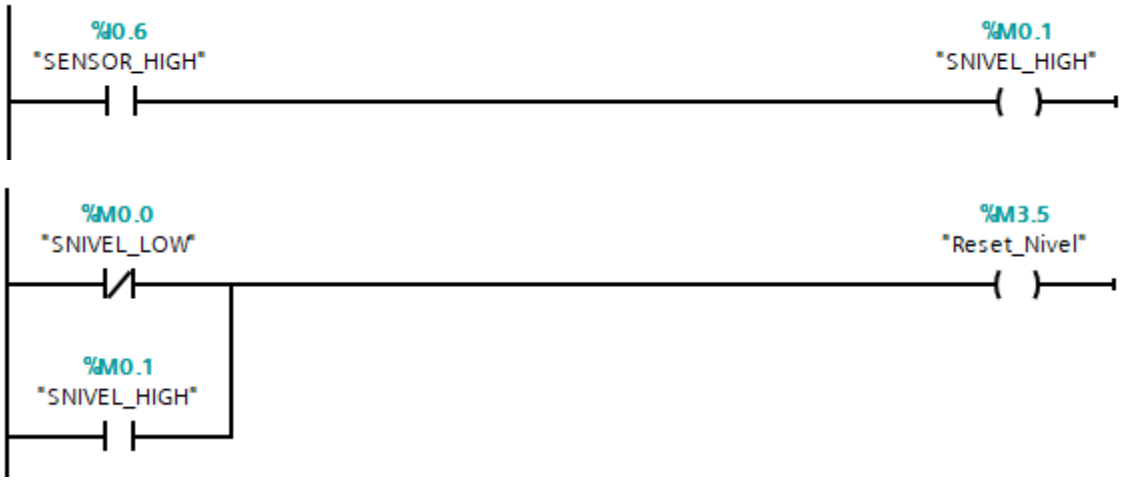


Segmento 6: Activar y desactivar solenoide 3 modo local



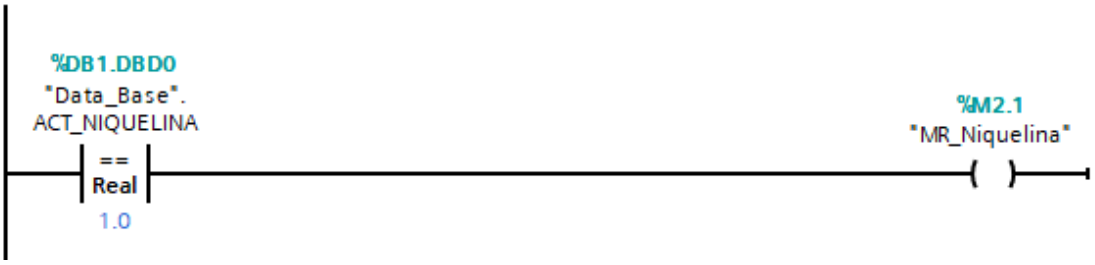


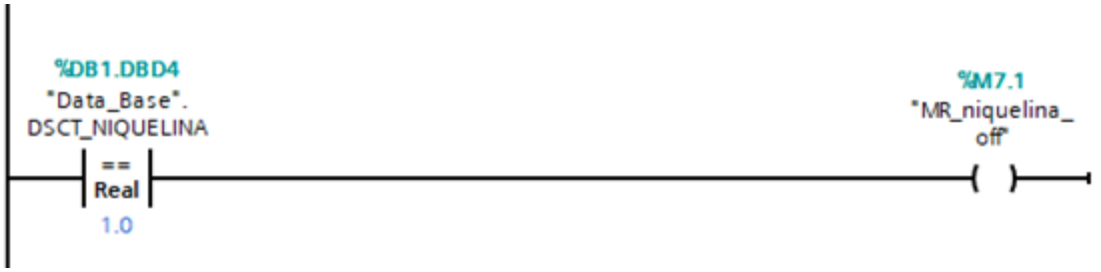
Segmento 7: Lectura de sensor de nivel de agua



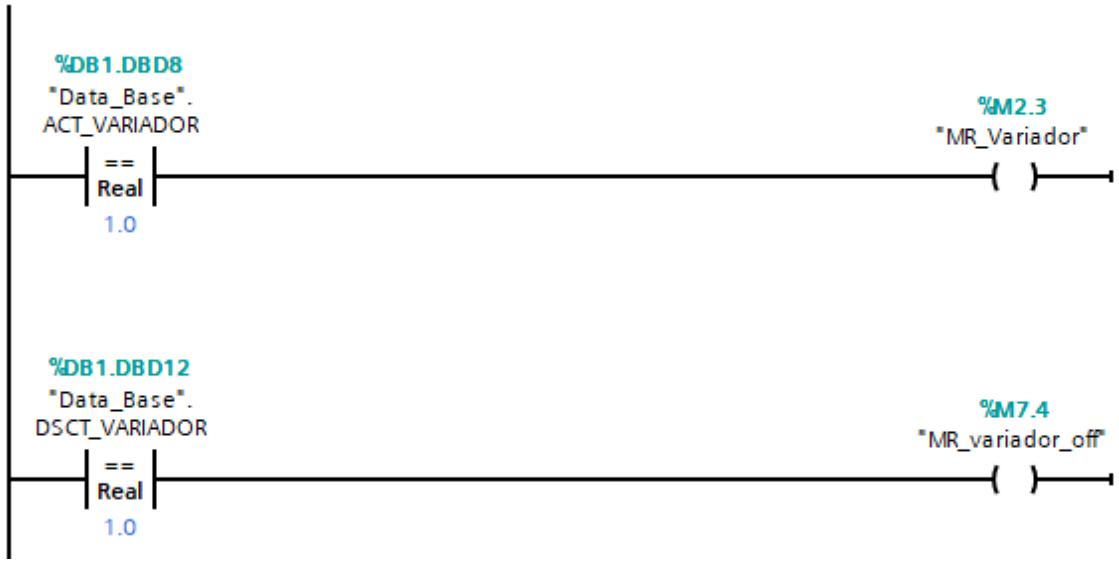
Bloques de programa: DISPOSITIVOS_REMOTO [FC3]

segmento 1: activar y desactivar niquelina HMI

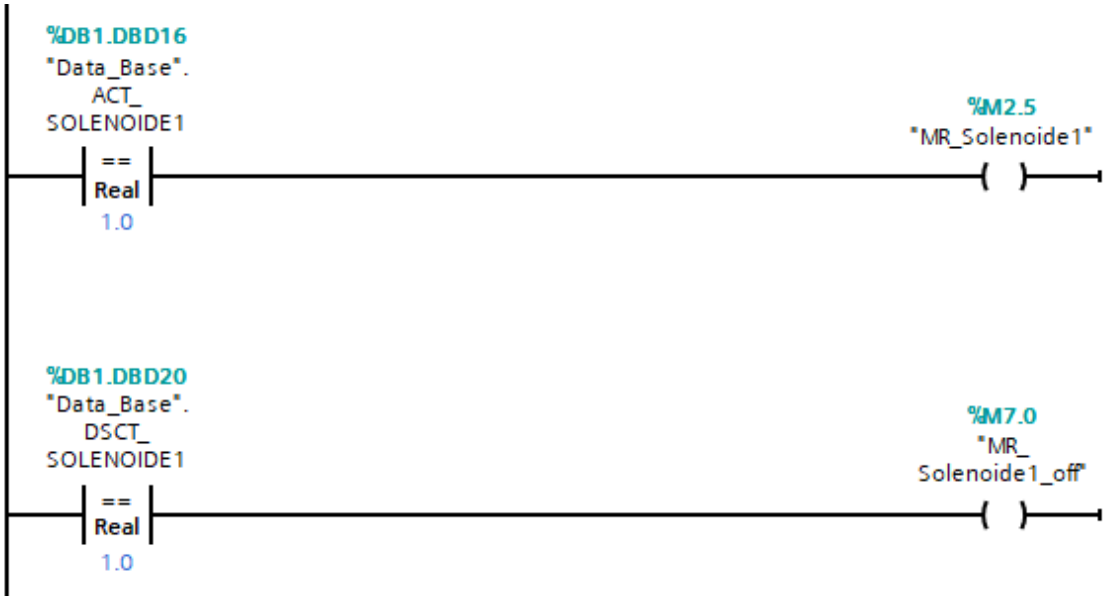




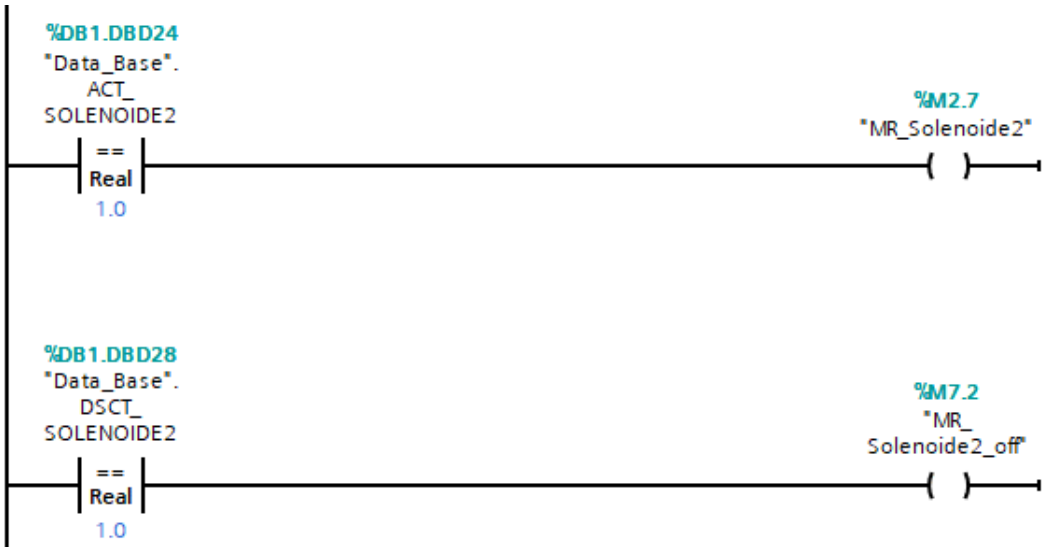
segmento 2: activar y desactivar variador de frecuencia desde HMI



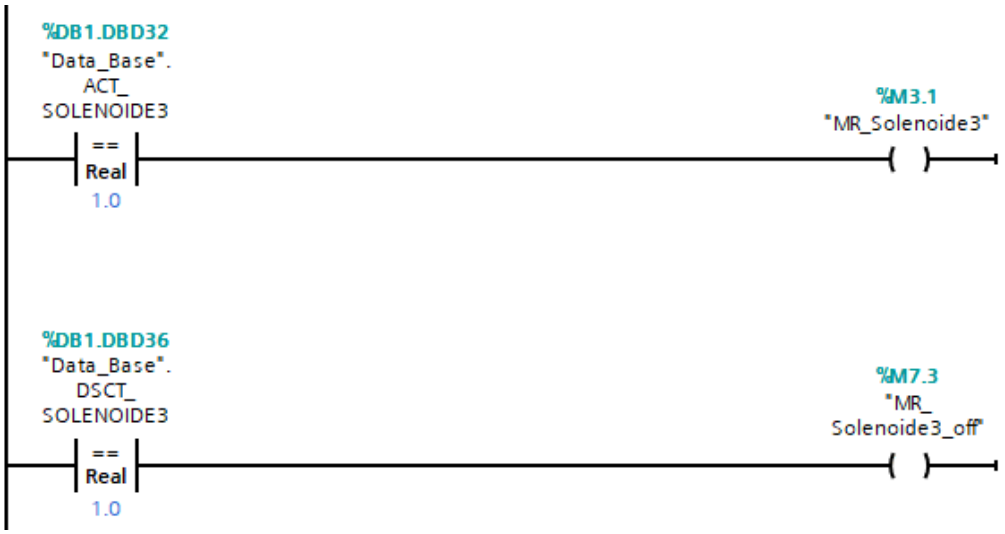
segmento 3: activar y desactivar solenoide 1 desde HMI



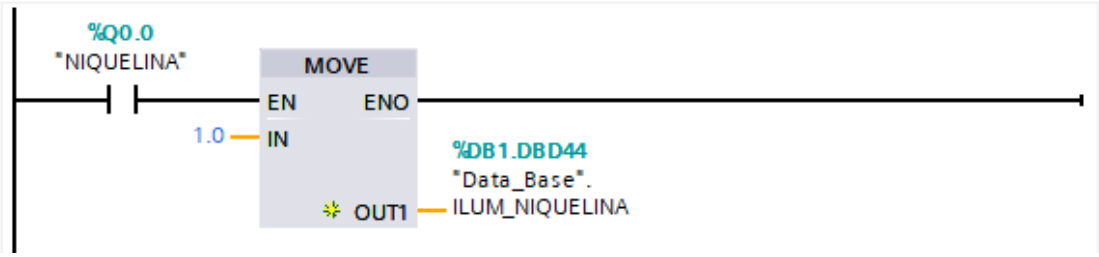
segmento 4: activar y desactivar solenoide 2 desde HMI

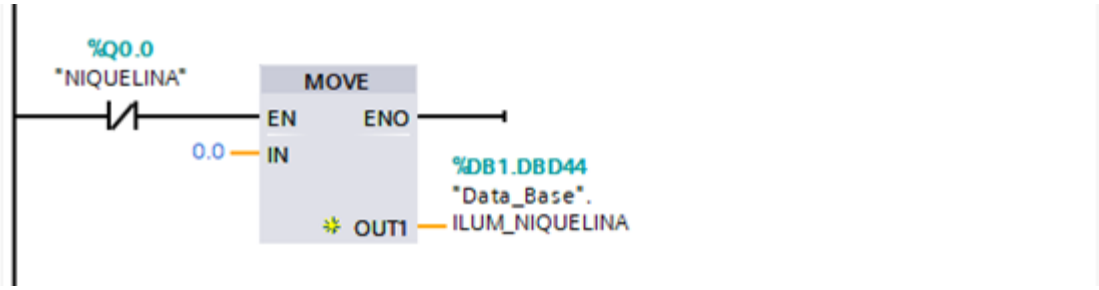


segmento 5: activar y desactivar solenoide 3 desde HMI

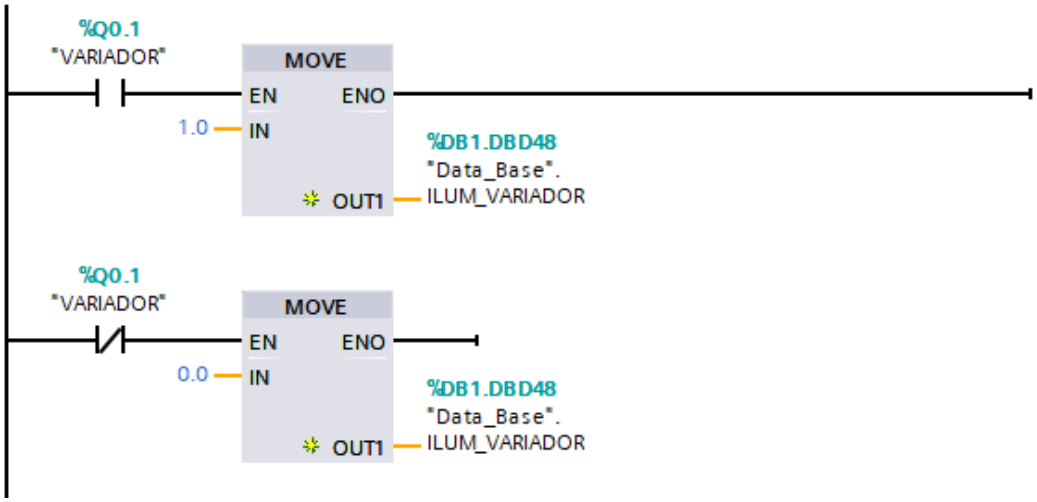


segmento 6: activar y desactivar luz piloto de niquelina en HMI

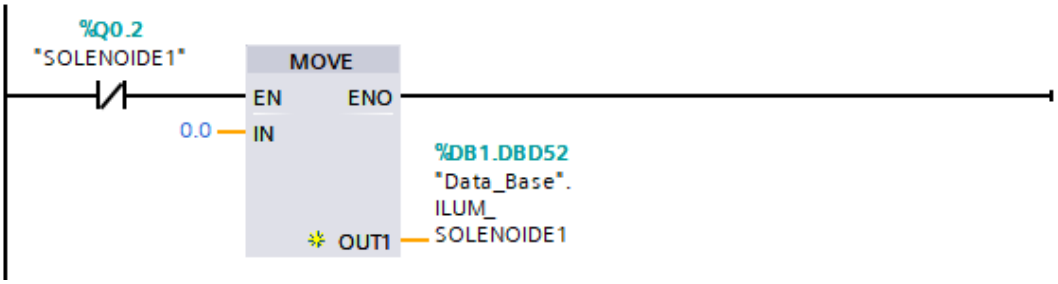


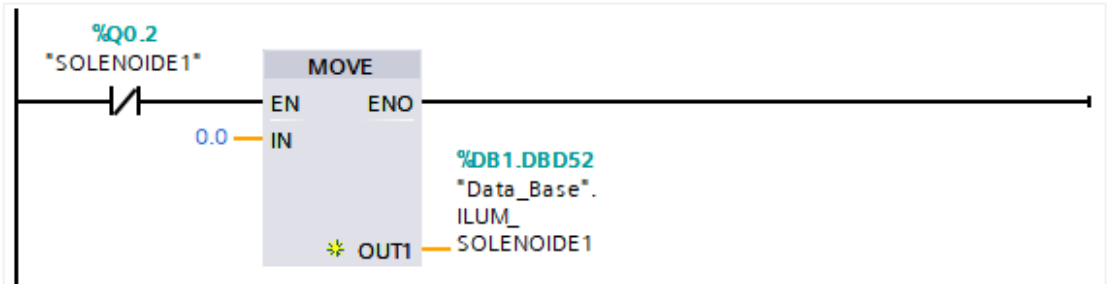


segmento 7: activar y desactivar luz piloto de variador en HMI

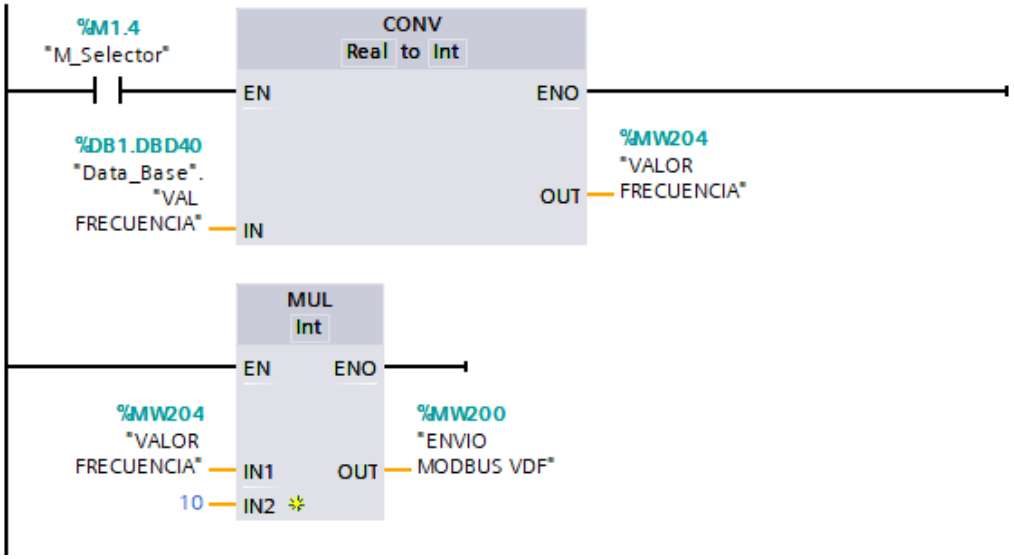


segmento 8: activar y desactivar luz piloto de solenoide 1 en HMI



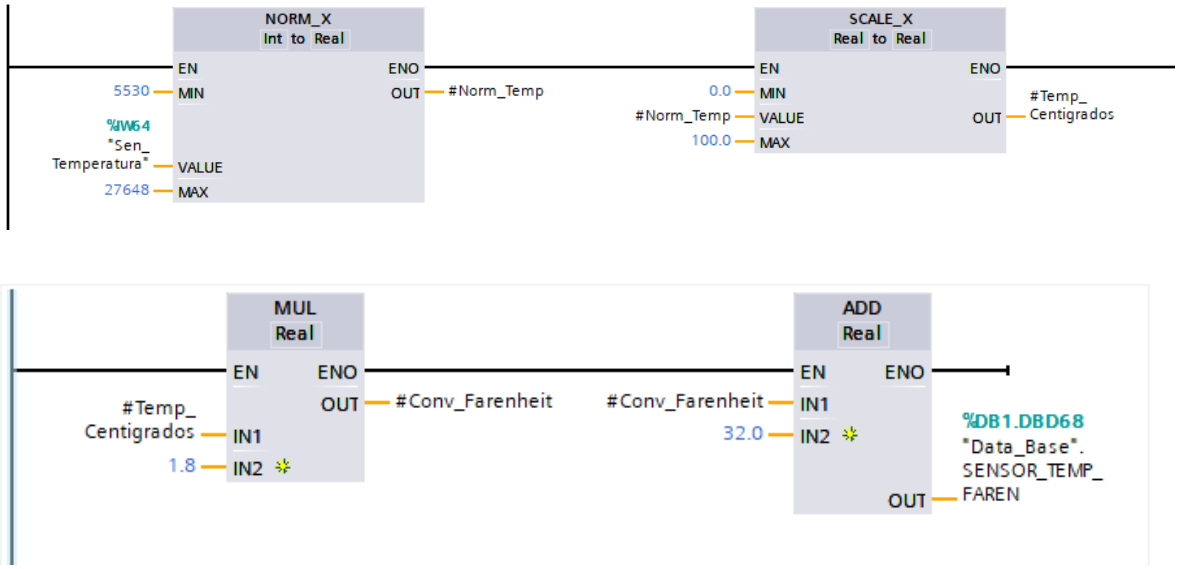


Conversión valor frecuencia y envío Modbus

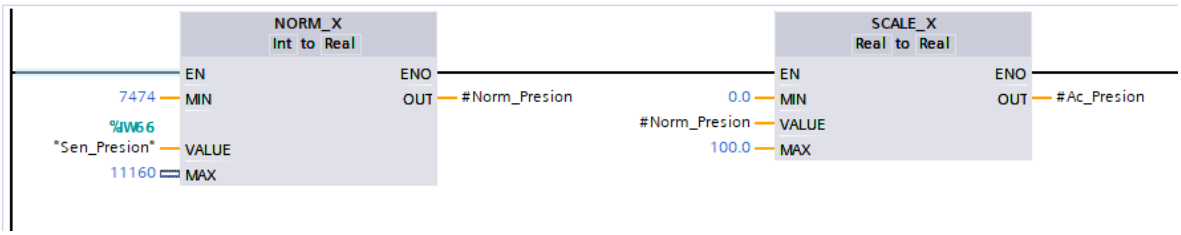


Bloques de programa: SENSOR_PRES_TEMP [FC5]

Segmento 1: Normalización y escalado de valores de temperatura



Segmento 2: Normalización y escalado de valores de presión



Bloques de programa: Data_Base [DB1]

Variables de lectura y escritura vía Modbus entre PLC y HMI

Data_Base

		Nombre	Tipo de datos	Offset	Valor de arranq...
1		Static			
2		ACT_NIQUELINA	Real	0.0	0.0
3		DSCT_NIQUELINA	Real	4.0	1.0
4		ACT_VARIADOR	Real	8.0	0.0
5		DSCT_VARIADOR	Real	12.0	1.0
6		ACT_SOLENOIDE1	Real	16.0	0.0
7		DSCT_SOLENOIDE1	Real	20.0	1.0
8		ACT_SOLENOIDE2	Real	24.0	0.0
9		DSCT_SOLENOIDE2	Real	28.0	1.0
10		ACT_SOLENOIDE3	Real	32.0	0.0
11		DSCT_SOLENOIDE3	Real	36.0	1.0
12		VAL FRECUENCIA	Real	40.0	0.0
13		ILUM_NIQUELINA	Real	44.0	0.0
14		ILUM_VARIADOR	Real	48.0	0.0
15		ILUM_SOLENOIDE1	Real	52.0	0.0
16		ILUM_SOLENOIDE2	Real	56.0	0.0
17		ILUM_SOLENOIDE3	Real	60.0	0.0
18		SEN_TEMP_CENTI	Real	64.0	0.0
19		SENSOR_TEMP_FAREN	Real	68.0	0.0
20		SENSOR_PRESION	Real	72.0	0.0
21		SENSOR_TANQUE_LOW	Real	76.0	0.0
22		SENSOR_TANQUE_HIGH	Real	80.0	0.0
23		SELECTOR	Real	84.0	0.0