UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

TRABAJO DE TITULACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE:
INGENIERO ELECTRÓNICO

PROYECTO TÉCNICO:
DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO Y ADQUISICIÓN DE DATOS DE LA PLANTA DIDÁCTICA MPS PA. COMPACT WORKSTATION DE FESTO UTILIZANDO HARDWARE LIBRE Y PROTOCOLOS DE COMUNICACIÓN SMTP Y GSM

AUTORES:
FRANCISCO ADRIÁN BANGUERA INTRIAGO
MANUEL ALFONSO DELGADO CEVALLOS

TUTOR:
ING. VÍCTOR LARCO TORRES MSc.

GUAYAQUIL - ECUADOR
2019

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA

Nosotros, Banguera Intriago Francisco Adrián con cédula de identidad N°.0930377783 y Delgado Cevallos Manuel Alfonso con cédula de identidad N°.0931005268, declaramos que este trabajo de titulación "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO Y ADQUISICIÓN DE DATOS DE LA PLANTA DIDÁCTICA MPS PA. COMPACT WORKSTATION DE FESTO UTILIZANDO HARDWARE LIBRE Y PROTOCOLOS DE COMUNICACIÓN SMTP Y GSM" ha sido implementado bajo los conceptos, análisis y conclusiones considerando los métodos de investigación, así como también el respeto a los derechos intelectuales a terceros, son de exclusiva responsabilidad de los autores; y la propiedad intelectual de la UNIVERSIDAD POLITÉCNICA SALESIANA.

Guayaquil, marzo del 2019

_____	_____
Francisco Banguera Intriago	Manuel Delgado Cevallos
CI: 0930377783	CI: 0931005268

CERTIFICADO DE CESIÓN DE DERECHO

Nosotros, Banguera Intriago Francisco Adrián con cédula de identidad N°.0930377783 y Delgado Cevallos Manuel Alfonso con cédula de identidad N°.0931005268, manifestamos nuestra voluntad de ceder la titularidad sobre los derechos patrimoniales de este trabajo de titulación "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO Y ADQUISICIÓN DE DATOS DE LA PLANTA DIDÁCTICA MPS PA. COMPACT WORKSTATION DE FESTO UTILIZANDO HARDWARE LIBRE Y PROTOCOLOS DE COMUNICACIÓN SMTP Y GSM" a la UNIVERSIDAD POLITÉCNICA SALESIANA según lo establecido por la ley de la propiedad intelectual y por la normativa institucional vigente.

Guayaquil, marzo del 2019

_____	_____
Francisco Banguera Intriago	Manuel Delgado Cevallos
CI: 0930377783	CI: 0931005268

CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN

Por medio de la presente declaro que bajo mi dirección y asesoría fue desarrollado este trabajo de titulación “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO Y ADQUISICIÓN DE DATOS DE LA PLANTA DIDÁCTICA MPS PA. COMPACT WORKSTATION DE FESTO UTILIZANDO HARDWARE LIBRE Y PROTOCOLOS DE COMUNICACIÓN SMTP Y GSM” realizado por los estudiantes FRANCISCO ADRIÁN BANGUERA INTRIAGO con cédula de identidad 0930377783 y MANUEL ALFONSO DELGADO CEVALLOS con cédula de identidad 0931005268, el mismo que cumple con los objetivos del diseño de aprobación y todos los requisitos pertinentes.

Guayaquil, marzo del 2019

_____

Ing. Víctor David Larco Torres, MSc.

Tutor

DEDICATORIA

Dedico este trabajo en primer lugar a Dios, quien nos regala la vida y nos llena de sabiduría para transitar en el camino de la vida. Segundo a mi familia, mis padres, mi hermana, y abuela quienes siempre me han brindado su apoyo y al igual que yo hemos esperado tanto este momento.

Por último, pero no menos importante a mi amigo y hermano que partió prematuramente de este mundo, sin embargo, fue quien me inspiró en muchos aspectos de mi vida y desde donde quiera que esté, seguirá haciéndolo.

Francisco Banguera Intriago

Este proyecto está dedicado especialmente a Dios por darme paciencia y fortaleza que necesité durante toda la carrera universitaria. A mi familia por estar siempre a mi lado aconsejándome para seguir sin rendirme a pesar de las dificultades que se me presentaban. A mi madre por ser el principal pilar en mi vida siendo padre y madre ella estuvo presente en los buenos y malos momentos ayudándome y aconsejándome durante todo este tiempo sin descanso alguno.

Manuel Delgado Cevallos

AGRADECIMIENTO

Agradezco a mi familia por siempre acompañarme en cada etapa de mi vida y en cada logro, a mis padres Aracely Intriago y Francisco Banguera; por haberme brindado la vida, educación, e inculcarme siempre valores.

A mi abuelita María que considero como una madre y siempre ha cuidado de mí, y también a mi primo que es como un hermano Abraham.

A mi hermana que cada día crece y está muy pendiente de mí y sobre todo en este proceso de titulación.

A mis amigos que son parte de mi familia, ya que siempre me han brindado unas palabras de aliento y ánimos para seguir creciendo, especialmente a Miguel, Enrique, Angelo, Jeshua y sobre todo a Manuel quien ha sido también mi compañero de clases y de trabajo de titulación al cual le hemos puesto todo el empeño y esfuerzo.

Francisco Banguera Intriago

Agradezco a Dios que me dio fuerzas necesarias para culminar la carrera universitaria. Agradezco a mi madre Mireya Delgado quien estuvo conmigo en los mejores momentos, en los logros que superé durante toda mi carrera, ella quien estuvo presente en todo momento brindándome todo su apoyo y aconsejándome, sé que estará muy orgullosa de mi por completar este ciclo de mi vida, todo un profesional. Agradezco a mis tíos quienes confiaron y aportaron con su granito de arena para culminar la carrera y verme finalmente graduado. Agradezco a mi compañero de tesis Francisco Banguera quien durante este proceso estuvo pendiente y detrás de la preparación de nuestro proyecto sin rendirse en ningún momento.

Manuel Delgado Cevallos

RESUMEN

AÑO	ALUMNOS	TUTOR DEL PROYECTO	TEMA DEL PROYECTO
2019	Banguera Intriago, Francisco Adrián Delgado Cevallos, Manuel Alfonso	Ing. Víctor David Larco Torres MSc.	“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO Y ADQUISICIÓN DE DATOS DE LA PLANTA DIDÁCTICA MPS PA. COMPACT WORKSTATION DE FESTO UTILIZANDO HARDWARE LIBRE Y PROTOCOLOS DE COMUNICACIÓN SMTP Y GSM”.

Actualmente los sistemas de comunicación nos facilitan tareas y nos ayudan a solucionar problemas en un menor tiempo, ya que han ido evolucionando pasando de ser solo llamadas telefónicas a un sin número de aplicaciones de mensajería instantánea.

Este proyecto utiliza los mensajes de texto y los correos electrónicos como un medio de comunicación entre la Planta Didáctica y el usuario encargado de su mantenimiento, ya que le permitirá recibir alertas por dichos medios, cuando exista un parámetro que exceda el rango de trabajo adecuado.

Los parámetros o variables de la planta que se van a monitorear serán: Nivel, temperatura, presión y flujo.

Además de los protocolos GSM y SMTP antes mencionados, se almacenará en una base de datos, cuando los parámetros superen el umbral de funcionamiento y se podrá visualizar en una página web, todo esto programado en LINUX con lenguaje de programación C en el software PYTHON. Teniendo así un menor impacto en posibles fallas que afecten el funcionamiento de la planta y un menor en el tiempo de mantenimiento correctivo cuando sea necesario.

ABSTRACT

YEAR	STUDENTS	PROJECT TUTOR	PROJECT THEME
2019	Banguera Intriago, Francisco Adrián. Delgado Cevallos, Manuel Alfonso	Ing. Víctor David Larco Torres MSc.	"DESIGN AND IMPLEMENTATION OF A SYSTEM OF MONITORING AND ACQUISITION OF DATA FROM THE MPS PA EDUCATIONAL PLANT. COMPACT WORKSTATION OF FESTO USING FREE HARDWARE AND SMTP AND GSM COMMUNICATION PROTOCOLS"

Currently communication systems facilitate tasks and help us solve problems in a shorter time, since they have evolved from being just phone calls to several instant messaging applications.

This Project uses text messages and emails as a means of communication between the Teaching Plant and the user in charge of its maintenance, since it will allow it to receive alerts by said means, when there is a parameter that exceeds the adequate range of work.

The parameters or variables of the plant that will be monitored will be: Level, temperature, pressure and flow.

In addition to the GSM and SMTP protocols, it will be stored in a database, when the parameters exceed the operating threshold and can be displayed on a web page, all programmed in LINUX with programming language C in the PYTHON software. Thus, having a lower impact on possible failures that affect the operation of the plant and a minor corrective maintenance time when necessary.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El siguiente proyecto consiste en el diseño e implementación de un sistema de monitoreo y adquisición de datos de una planta didáctica de FESTO mediante el envío de SMS con la ayuda de la tarjeta Arduino y correo electrónico utilizando la Raspberry, almacenando registros en una base de datos creada en un servidor.

Este proyecto está enfocado principalmente en el mantenimiento preventivo y correctivo de la planta didáctica cuya finalidad ésta se encuentre en un buen estado para el uso de los estudiantes de Automatización.

En la primera parte se definen los hechos preliminares tales como el planteamiento del problema, metodologías, técnicas, impacto del proyecto para los distintos usos de la planta didáctica de FESTO, entre otros puntos importantes en el desarrollo del proyecto.

Luego se desarrolla el marco teórico donde se detallan los conceptos generales, descripciones y características de todos los elementos que se utilizaron en el desarrollo del proyecto.

Posteriormente se describe el análisis y diseño del proyecto detallando el paso a paso de la implementación de la adquisición de datos, el envío de correo y SMS.

Finalmente se presentan las pruebas realizadas del proyecto, las conclusiones y recomendaciones de este.

1. EL PROBLEMA

1.1. Planteamiento del problema

En la Planta Didáctica Industrial MPS PA. Compact Workstation FESTO no era posible saber el estatus de los sensores y elementos, y si se requiere hacer un mantenimiento llevaría más tiempo saber que elemento específicamente presenta fallas por la carencia de dicha información y sin la existencia de reportes periódicos, que ahorraría tiempos de paro de la planta y garantizaría el menor tiempo posible en un futuro mantenimiento, evitando pérdidas de horas de clases demostrativas con la planta.

1.2. Antecedentes

A través de los años la tecnología GSM se ha vuelto muy conocida por el mundo y es la que ha servido como pilar para llevar a cabo las distintas tecnologías de libre acceso que existen hoy en día, actualmente en el mundo el 90% de las operadoras de telefonía celular cuentan con el servicio de GSM.

Del mismo modo el uso de correo electrónico se ha hecho muy frecuente en las empresas y prestadores de servicios, incluso utilizados por los bancos para el envío de notificaciones a sus usuarios.

1.3. Importancia y alcance

El empleo de las nuevas tecnologías, entre ellas la comunicación inalámbrica dentro de los procesos industriales ha beneficiado el desarrollo de nuevas aplicaciones en el ámbito educacional, basado en el diseño, implementación y monitoreo de redes inalámbricas, que conllevan a los futuros estudiantes de Ingeniería Electrónica a poner en práctica estos conocimientos y se reflejen los resultados en su vida profesional.

Las redes inalámbricas han captado mucho la atención en los procesos de control, por lo que la industria de la automatización se está esforzando en el desarrollo de nuevos protocolos. En los sistemas de control en red, por ejemplo, es grande el interés en el crecimiento de la tecnología inalámbrica como un sustituto potencial para la actual generación de redes cableadas industriales (Monsalve Posada, Arias Londoño, & Mejía Arango, 2015).

1.4. Delimitación del Problema

1.4.1. Delimitación espacial

Este sistema diseñado para el monitoreo y adquisición de datos de la Planta Didáctica Industrial MPS PA. Compact Workstation FESTO ha sido demostrado mediante pruebas en el laboratorio de Automatización ubicado en el cuarto piso del bloque E de la Universidad Politécnica Salesiana sede Guayaquil campus Centenario, en las calles Chambers #277 entre Laura Vicuña y Robles.

1.4.2. Delimitación temporal

El proyecto fue diseñado e implementado en un periodo de 1 año y medio entre diciembre 2017 y marzo 2019.

1.4.3. Delimitación académica

Se diseñó e implementó un sistema de monitoreo y adquisición de datos de la Planta Didáctica Compact Workstation de Festo, usando hardware libre (Raspberry y Arduino), programado bajo la plataforma del Sistema Operativo Linux y creación del servidor de mensajería instantánea con ayuda del módulo GPRS-GSM y un servidor de correo electrónico. Se usará el software LABVIEW para la creación de interfaces gráficas y se adquirirá equipamiento adicional necesario.

Además de los softwares, Python en el sistema operativo Linux y el IDE de Arduino.

1.5. Objetivos

1.5.1. Objetivo general

Diseñar e implementar un sistema de monitoreo y adquisición de datos de la planta didáctica MPS PA. Compact Workstation de FESTO utilizando hardware libre y protocolos de comunicación SMTP y GSM.

1.5.2. Objetivos específicos

- Adquirir los datos de presión, temperatura, nivel y flujo de la planta didáctica Compact Workstation de Festo.
- Realizar un enlace de comunicación entre las tarjetas DAQ Arduino y Raspberry, y el módulo GSM.
- Diseñar la interfaz gráfica de las variables adquiridas.
- Crear una base de datos para almacenar la información con la finalidad de tener un registro de los eventos.
- Generar un reporte con los datos adquiridos de la planta.
- Diseñar un sistema de comunicación que pueda enviar mensajes de texto SMS y correo electrónico mediante un servidor SMTP.

2. MARCO TEÓRICO

2.1. Planta Didáctica Industrial MPS PA Compact Workstation

El MPS PA. Compact Workstation es un sistema para la automatización de procesos y tecnología diseñado para facilitar la formación profesional en el área de instrumentación industrial y procesos de control orientado al sector, además los componentes industriales como sensores, transductores y actuadores son didácticamente apropiados, pudiendo con ellos establecer lazos de control según las variables que se manejen: flujo, nivel, presión o temperatura. (Festo Didactic, 2008)

2.1.1. Detalles Generales

Se deben tomar en cuenta las propiedades y medidas de funcionamiento que rige la planta y cada uno de los componentes para poder realizar cualquier acción de control o uso del MPS.

Eléctrico: Utilizar un voltaje por debajo de 24VDC para alimentar los actuadores y sensores. El calentador trabaja con tensión de 110VAC a 230VAC.

Neumático: No debería exceder la presión de 8 bar. Debe instalar y asegurar las conexiones de tuberías antes de encender el aire comprimido.

Mecánico: Coloque los componentes fijamente en la estructura mecánica. Solo se deben maniobrar los elementos cuando la planta no esté funcionando. (Festo Didactic, 2008)

2.1.2. Diseño y funcionamiento

El MPS PA. Compact Workstation combina 4 lazos de control con sensores y actuadores digitales y analógicos. Con la ayuda de un PLC o controlador puede utilizarse individualmente o en cascada.

- Sistemas de control de nivel.
- Sistemas de control de flujo.
- Sistemas de control de presión.
- Sistemas de control de temperatura.

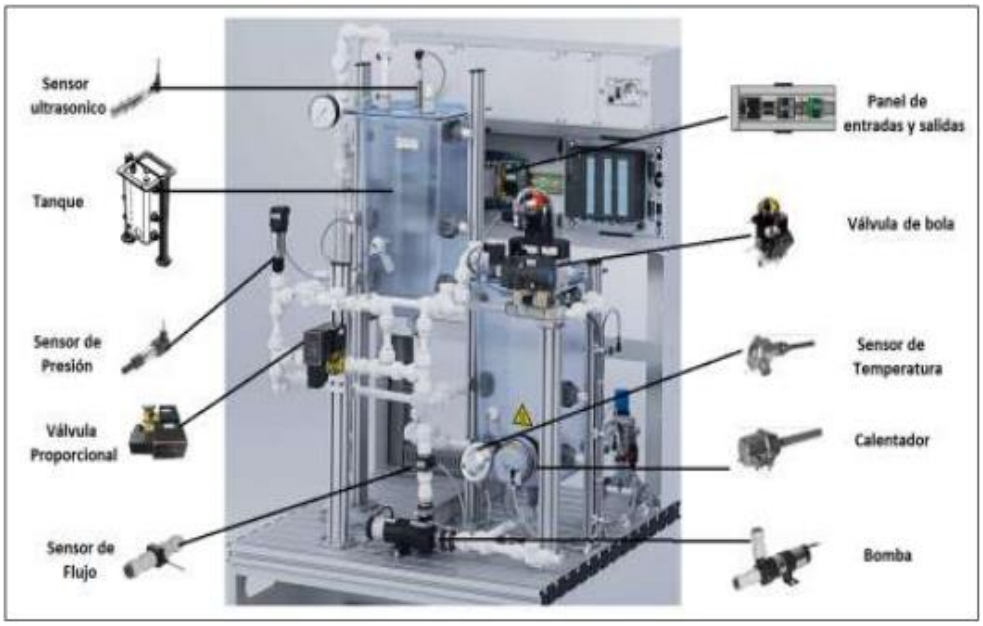

Figura 1. MPS PA. Compact Workstation. *(Idrovo & Peña, 2014)*

A continuación, se describen los componentes básicos de MPS PA. Compact Workstation observados en la Figura 2:

BINN 101, BINN 102: Pertenecen a los tanques B101 Y B102 que se usan para almacenar el líquido y muestran una escala de medida.

VSSL103: Tanque de aire a presión.

PUMP 10: Bobina centrífuga P101 que suministra de fluido al sistema.

E104: Elemento calefactor que eleva la temperatura del líquido en el tanque B101.

V101, V103, V104, V105, V107, V108, V109, V110, V112: Válvulas manuales que se abren o cierran permitiendo el paso del fluido por las tuberías.

V102: Válvula neumática de bola controlada por un actuador giratorio. La velocidad rotacional se representa con la letra S encerrada en un rectángulo en el P&ID.

V106: Válvula proporcional 2/2 para control de flujo.

FIC B102: Controlador Indicador de flujo

FIC B103: Controlador Indicador de presión.

TIC B104: Controlador Indicador de temperatura.

LIC B101: Controlador Indicador de nivel de líquido.

PI 105: Indicador de presión.

LSL B113 y LSL S117: Interruptores de líquido de nivel bajo.

LSH B113 y LSH B114: Interruptores de líquido nivel alto.

LSH S111: Interruptor flotador para nivel de líquido alto. (Idrovo & Peña, 2014)

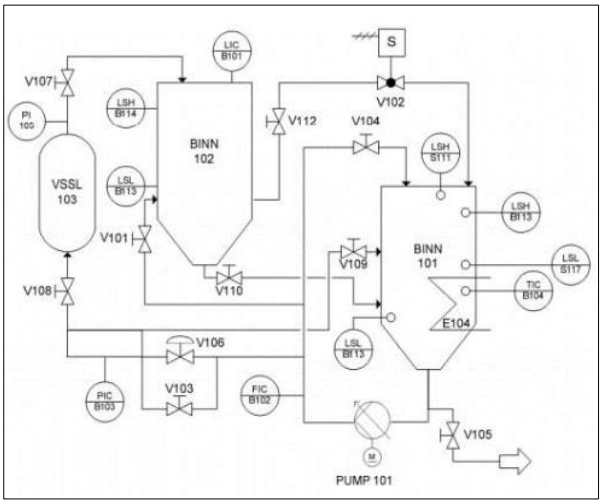

Figura 2. *P&ID del MPS PA. Compact Workstation. (Festo Didactic, 2008)*

2.2. Sistemas embebidos. Adquisición de datos analógicos y digitales.

2.2.1. Arduino

Arduino está compuesto por 3 partes:

- **Una tarjeta electrónica** con un microcontrolador reprogramable interno que posee pines, los cuales representan las entradas y salidas E/S, que permiten conectar actuadores y sensores, para futuros proyectos, como se visualiza en la Figura 3. (Torrente, 2013)

- **Un software** (o un “entorno de desarrollo”) gratis, libre y multiplataforma (ya que funciona en Linux, MacOS y Windows) que se instala en nuestro ordenador y que nos permite escribir, verificar y cargar en la memoria del microcontrolador de la placa Arduino el conjunto de instrucciones que deseamos que este empiece a ejecutar. (Torrente, 2013)

- **Un lenguaje de programación libre.** Por “lenguaje de programación” se entiende cualquier idioma artificial diseñado para expresar instrucciones con determinadas reglas sintácticas que pueden ser llevadas a cabo por máquinas. (Torrente, 2013)

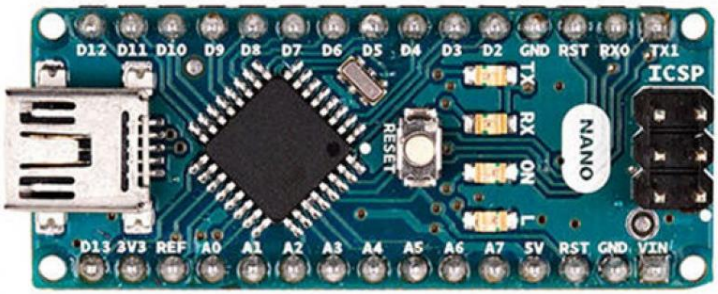

Figura 3. Placa Arduino NANO*. (Arduino, 2018)*

Como indica Arduino (2018) Arduino Nano es un tablero pequeño de microcontroladores basado en el ATmega328P, completo y fácil de usar, ofrece la misma conectividad y especificaciones de la placa UNO en una proporción más pequeña. Tiene 22 pines digitales de entrada / salida (de los cuales 6 se pueden usar como salidas PWM), 8 entradas analógicas, un

oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP, y un botón de reinicio. Contiene todo lo necesario para soportar el microcontrolador; simplemente conéctelo a una computadora con un cable USB o con un adaptador de CA a CC o batería para comenzar. Tensión de entrada de 7 – 12 voltios, consume 19mA, cada entrada/salida consume 40mA DC.

2.2.2. Raspberry Pi

Raspberry Pi es un procesador multimedia “Broadcom BCM2835”, esto quiere decir que la mayor parte de los componentes del sistema, la unidad central de procesamiento y la de gráficos junto con el audio y comunicaciones, que se encuentran integrados en el centro de la placa, con una memoria de 256MB. A diferencia de otras placas electrónicas, es que Raspberry utiliza una jerarquía de comandos denominada ARM, mostrada en la Figura 4.

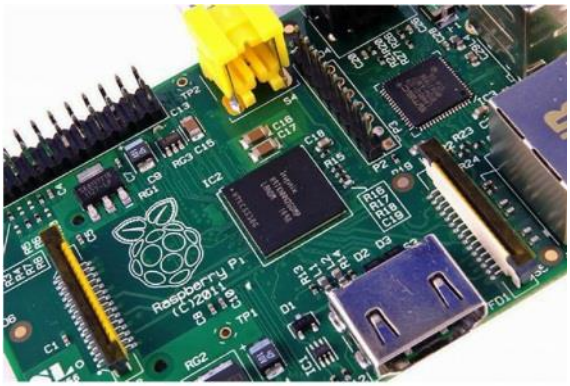

Figura 4. SoC BGM2835 ubicado en el centro de la placa. *(Upton & Halfacree, 2013)*

El conjunto de instrucciones ARM son el secreto de cómo la Raspberry es capaz de funcionar con una alimentación de 5V a 1A suministrada por un puerto interno. Otra de las razones es que no se encontrará ningún disipador térmico en el dispositivo, puesto que no consume mucha energía ni produce calor interno por más compleja que sea la tarea por ejecutar.

Otra diferencia existente entre la Raspberry y su computadora de escritorio o laptop es el sistema operativo que utiliza la Raspberry se diseñó para

ejecutarse en un SO basado en LINUX, a diferencia del resto de sistemas operativos, éste es de código abierto, por su tamaño y costo, es de fácil manejo.

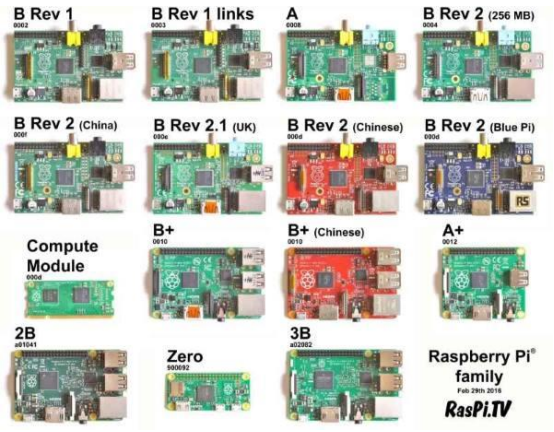

Figura 5. Modelos de Raspberry. (Upton & Halfacree, 2013)

2.3. Servidor de mensajería instantánea.

2.3.1. Arduino Shield GSM Sim900

Arduino Shield GSM es una placa que sirve para conectarse a internet, hacer y recibir llamadas de voz también enviar y recibir mensajes SMS. El Shield GSM utiliza un módem de radio M10 por Quectel. Es posible comunicarse con el tablero utilizando los comandos AT. La biblioteca GSM tiene un gran número de métodos para la comunicación con el escudo.

Esta placa utiliza pines digitales 2 y 3 para el software de comunicación serie con el M10. Pin 2 está conectado al pin TX del M10 y el pin 3 con el pin RX. Consulte estas notas para trabajar con un Arduino Mega, Mega ADK o Leonardo. Pin PWRKEY del módem está conectado al pin de Arduino 7.

El M10 es un cuatri-banda módem GSM / GPRS que funciona en las frecuencias GSM850MHz, GSM900MHz, DCS1800MHz y PCS1900MHz. Es compatible con los protocolos TCP / UDP y HTTP a través de una conexión GPRS. GPRS enlace descendente de datos y la transferencia de enlace ascendente de velocidad máxima es de 85.6 Kbps. Para interactuar con la red celular, la junta requiere una tarjeta SIM proporcionada por un operador de

red. La revisión más reciente de la junta utiliza el pinout 1.0 en el pin 3 de la placa Arduino Uno.

Antes de utilizar la placa Arduino Shield hay que considerar los siguientes requisitos:

- Requiere una placa Arduino.
- 5V Tensión de funcionamiento aislada de la placa Arduino Uno.
- Conexión con Arduino Uno de los pines 2, 3 (Serial Software)
- Una tarjeta SIM de cualquier operadora celular.
- Conexión de una antena transmisora y receptora. (Arduino, Escudo Shield Arduino, 2015)

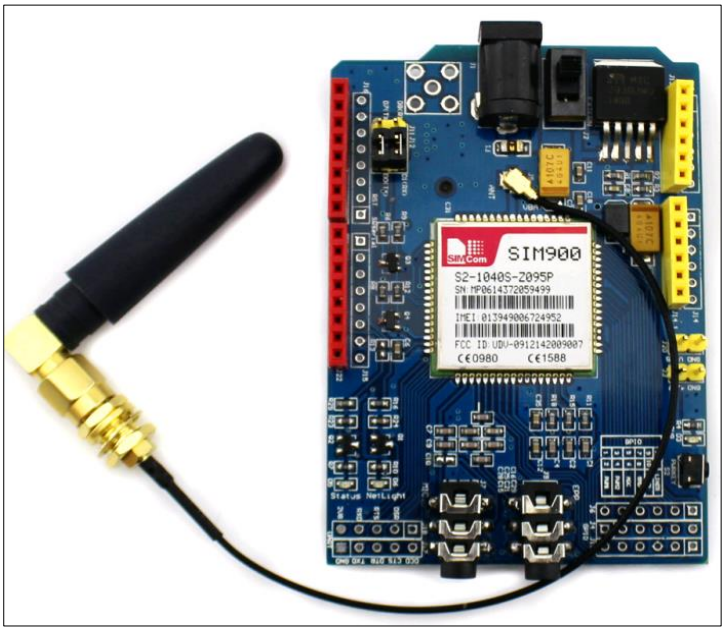

Figura 6. Tarjeta de comunicación Arduino Shield Sim900. *(Arduino, Escudo Shield Arduino, 2015)*

2.4. Creación de Servidor WEB

2.4.1. ¿Qué son los servidores web y por qué son necesarios?

Según Duplika (2010), la industria se hace muy compleja para la mayoría del web máster, en especial aquellos que no están familiarizados con términos de web hosting o lo que comúnmente llamamos “servidor”.

La posibilidad de alquilar un espacio en un servidor para almacenar los archivos de nuestro sitio se denomina como servidor web, por si queda duda visualicen varias computadoras encendidas las 24 horas, los 365 días del año.

Un servidor Web almacena archivos de un sitio donde la información es visualizada por un usuario a través de la Internet, es decir imaginen una gran computadora que envía y recibe información y cuando el usuario navega por internet se comunica con el servidor enviando y recibiendo datos que se visualizan en la pantalla. Es por ello, que los servidores Web según lo que solicite el usuario, transmite y almacena datos de un servidor a otro a través del Internet. (Duplika, 2010)

2.4.2. ¿Cómo funcionan los servidores?

Existe un código único que se le asigna a cada servidor y computadora que se conectan a la Internet, llamada dirección IP o protocolo de Internet como lo especifica sus siglas. Estas direcciones actúan como emisores y receptores de la información almacenada y solicitada, cuando visitas una página web, tu computador o móvil solicita un pedido desde la IP del dispositivo hacia la IP del servidor, de esta forma se establece una comunicación entre ellos, donde el servidor Web da su respuesta mostrando los datos solicitados a la dirección IP solicitante. (Duplika, 2010)

2.4.3. ¿Por qué son necesarios los servidores?

Si los servidores web no existieran, no habría información almacenada por ende no habría comunicación con las direcciones IP dentro de la telaraña

ciber náutica a la cual llamamos Internet, sin los servidores la industria del internet dejaría de funcionar y no habrían más avances tecnológico, puesto que la mayoría de servidores son comprados y utilizados para diferentes fines, se podría pensar que son una gran vitrina donde se publica una inmensa cantidad de información que necesita ser visualizada, tratada y transmitida hacia la red mundial. (Duplika, 2010)

3. MARCO METODOLÓGICO

En este apartado se describirá el diseño e implementación del sistema de monitoreo y adquisición de datos para la planta didáctica MPS PA Compact Workstation de Festo, donde se explica detalladamente la conexión de los equipos y la programación implementada en el software.

La implementación del sistema de monitoreo y adquisición está basado y regido de los siguientes esquemas como se muestra en las Figura 7 y Figura 8.

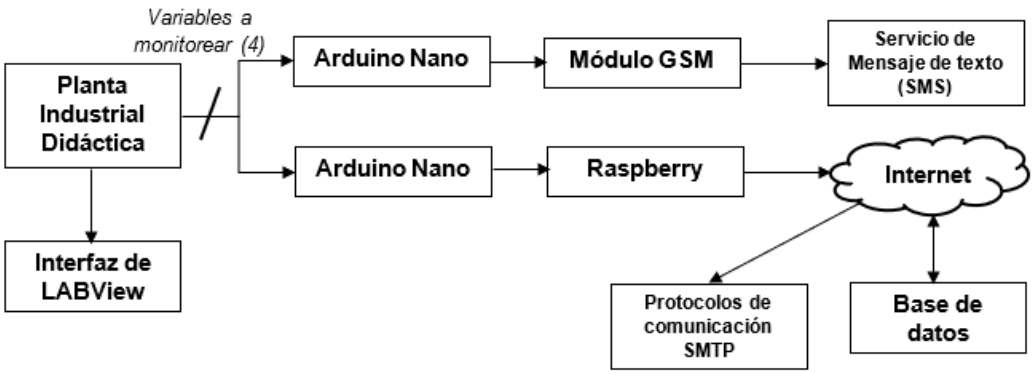

Figura 7: Diagrama de bloques

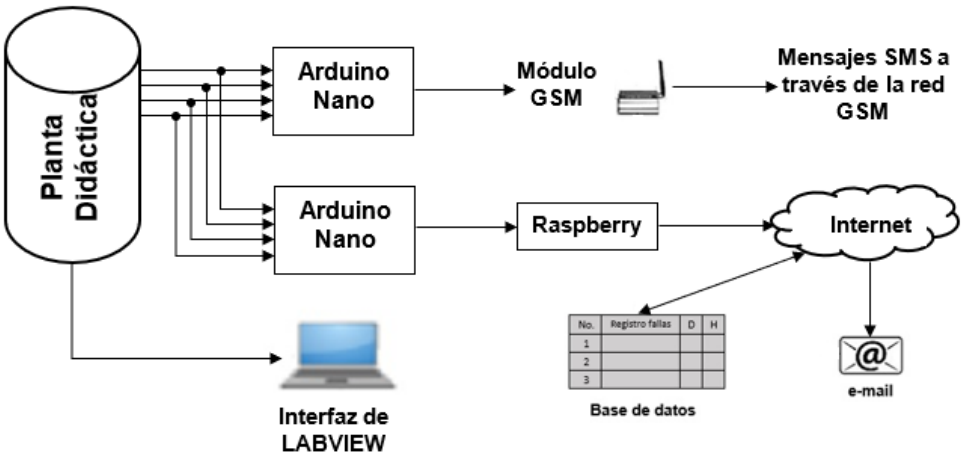

Figura 8: Diagrama físico

Con el avance de la tecnología existe una gran variedad de Hardware Libre que se utiliza para diferentes propósitos, como adquisidor de datos, para comunicación inalámbrica, almacenamiento de información, entre otros; que con la ayuda de software de programación los proyectos se vuelven más amigables y didácticos.

Para el proyecto propuesto se logrará una comunicación entre la planta de Festo y el usuario mediante el adquisidor de datos que será un Arduino Uno, para luego establecer una conexión con la Raspberry Pi donde transmitirán información simultáneamente, estos registros obtenidos a tiempo real se almacenarán en una base de datos creada, cuya información será enviada vía correo electrónico y mensajería instantánea SMS, todo esto programado en Python, bajo la plataforma de Linux. En la Figura 9 se muestra la Planta de Festo a utilizar.

Figura 9: Planta Didáctica MPS PA Compact Workstation

3.1. Diseño e Implementación del Sistema de monitoreo

3.1.1. Adquisición de datos de la planta MPS PA. Compact de Festo

Como primer paso para el desarrollo de nuestro sistema de monitoreo se deben adquirir los datos de los cuatro parámetros que se obtienen de la Planta didáctica, como lo son: presión, temperatura, nivel y flujo.

Figura 10: Visualización de los pines de salida de la tarjeta Easy Port de la planta de Festo.

Con ayuda del multímetro e investigando los manuales se identificaron los pines correspondientes a las cuatro variables mencionadas, del conector DB15 como se muestran en la Figura 11 y la respectiva configuración en la Figura 12.

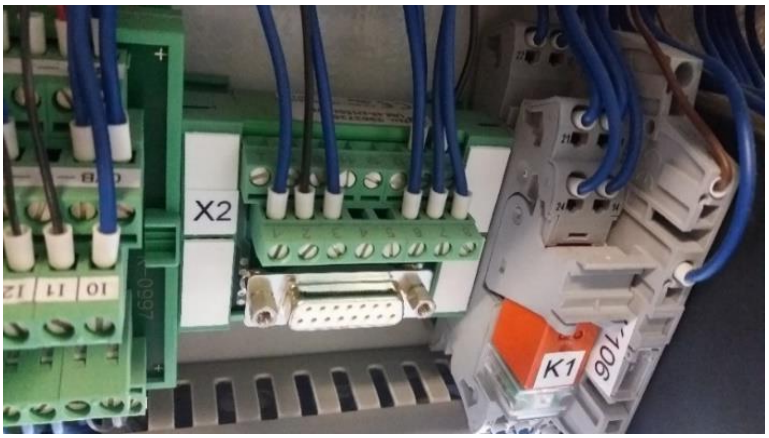

Figura 11: Identificación de los pines de los cuatro parámetros

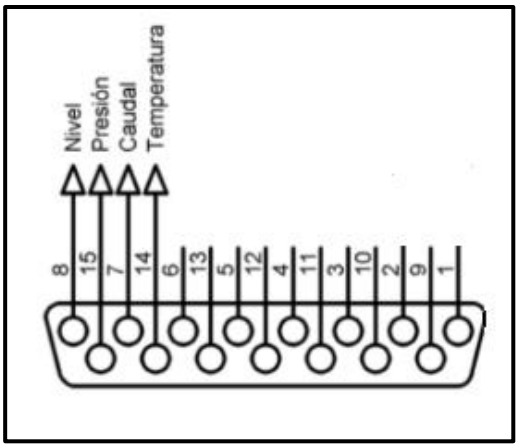

Figura 12: Configuración de los pines para el conector DB15.

3.1.2. Comunicación entre las tarjetas

Una vez identificados los pines, se procede a la adquisición de datos y se realizan las respectivas conexiones de las tarjetas embebidas, como lo son Arduino y Raspberry Pi, para ello se diseñó un conector DB15 con doble derivación o en “Y”, como se muestra en la Figura 13.

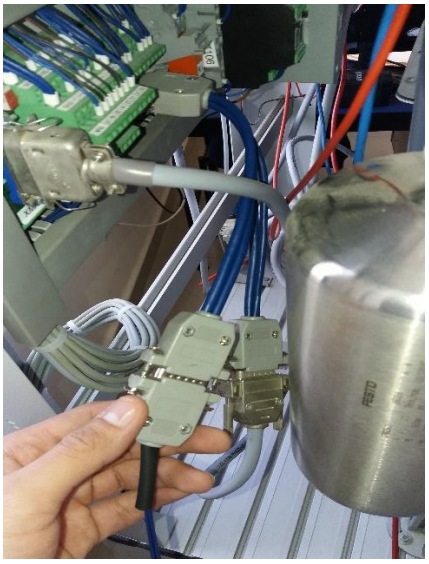

Figura 13: Conector DB15 con derivación en “Y”.

Esta configuración del conector DB15 facilita la comunicación entre el PLC de la Planta de Festo, la tarjeta Easy Port y el Arduino, para la lectura de información de las variables a tiempo real, sin alterar el diseño original de la Planta MPS PA.

En la Figura 14, se procede a realizar la comunicación entre el Arduino y la Raspberry Pi, para ello se necesitaron divisores de voltaje I2C, que por diseños de fabricante la ARDUINO es sensible a corrientes y voltajes elevados, ésta trabaja con rangos bajos de 0v a 5v, en cambio el Raspberry soporta hasta 24 voltios.

Posteriormente, el módulo GSM se conecta con el Arduino, para ello se necesitó agregar una tarjeta adicional para que existiera comunicación entre ellos, y un Display LCD para la visualización previa de las variables a tiempo real como se muestra en la Figura 15. Con la información obtenida se genera una base de datos desde la Raspberry y por consiguiente el envío de reportes vía correo electrónico y SMS, que se detallará en la sección 3.2.

Figura 14: Comunicación entre las tarjetas Arduino y Raspberry Pi.

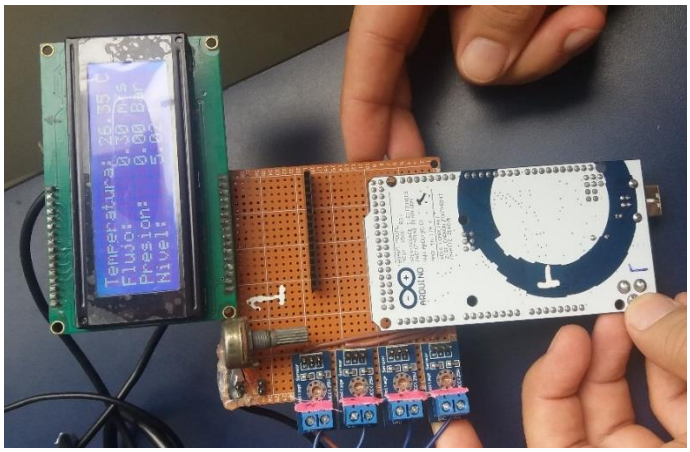

Figura 15: Interfaz de visualización de datos.

3.1.3. Interfaz Gráfica en Labview

Una vez establecida la comunicación entre las tarjetas, se procede a la interfaz gráfica para una visualización de la información que se almacena a tiempo real de la Planta de Festo a la base de datos. El PLC de la Planta MPS PA Compact Workstation utiliza lenguaje K o lenguaje gráfico por bloques y se programa en el software TIA PORTAL, para darle una mejor presentación y con un entorno amigable asociamos TIA PORTAL con el software NI LABVIEW, éste último puede comunicarse con cualquier controlador lógico (PLC). Para la comunicación de datos a tiempo real entre dispositivos de control de una planta y las interfaces HMI se utiliza el estándar o protocolo OPC Server, como se muestra en la Figura 16.

Para ello dentro del programa que se compila al PLC se deberá declarar variables como etiquetas HMI, que serán utilizadas para establecer una comunicación entre el PLC, el OPC Server y los VIs de LABVIEW, de esta forma se observará por medio de gráficos el funcionamiento de la planta a tiempo real, como se muestra en la Figura 17.

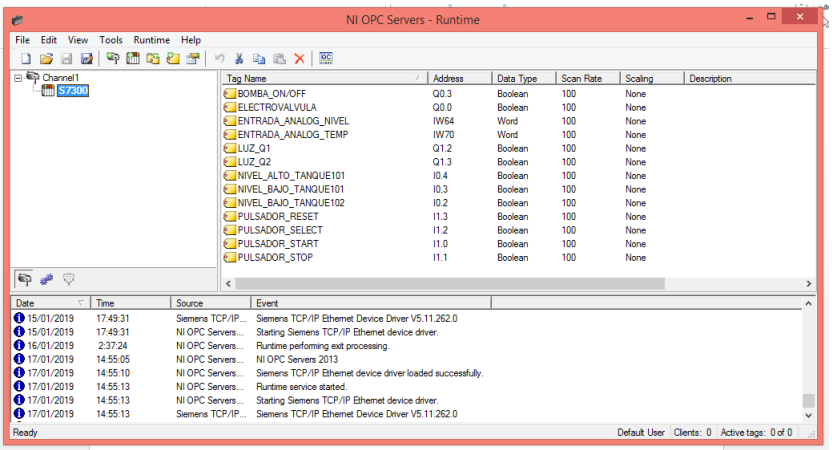

Figura 16: Comunicación a tiempo real de las variables del PLC por medio del OPC Server hacia la interfaz de LABVIEW

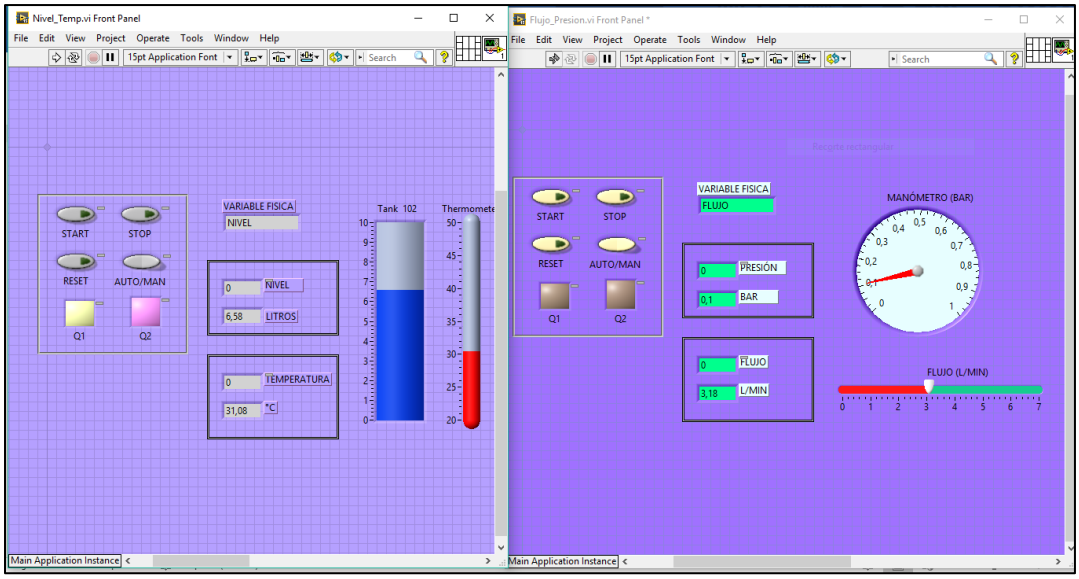

Figura 17. Interfaz gráfica de los datos obtenidos por la Planta MPS PA Compact Workstation en LABVIEW.

3.2. Implementación de los protocolos SMTP y GSM

3.2.1. Base de datos

Se realiza la programación en Python y se instalan las librerías para la creación de la base de datos en la plataforma Firebase de Google. Aquí se almacenarán los datos cuando alguno de los parámetros haya excedido su rango de trabajo adecuado con su respectiva fecha y hora de la novedad.

Para la creación de la base de datos tenemos que previamente tener una cuenta de correo de Gmail, en nuestro caso utilizamos:

monitoreo.mpspa@gmail.com

Una vez que nos autenticamos con la cuenta Google nos aparecerá la consola principal y seleccionamos nuestro proyecto que en este caso lo nombramos “Monitoreo”.

Figura 18: Consola principal de Firebase.

Dentro de nuestro proyecto nos aparecen los datos informativos de nuestra base de datos y para acceder a la base de datos escogemos la opción Database.

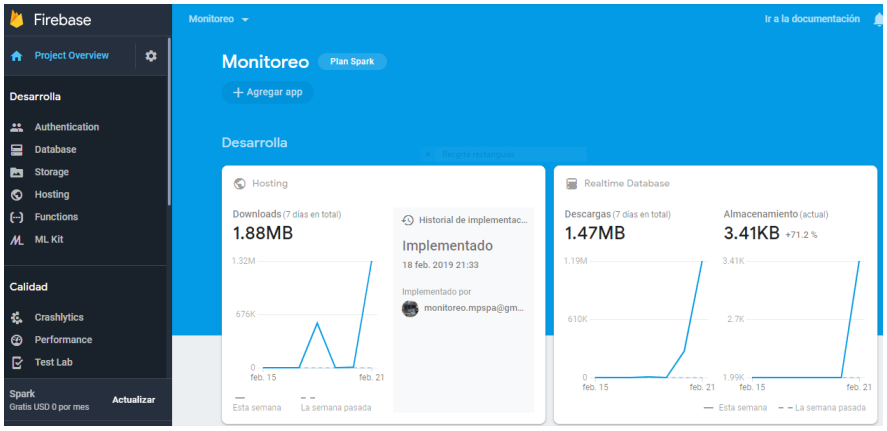

Figura 19: Página principal del proyecto.

En Database podemos observar nuestros datos almacenados, a cada alerta se le asigna un nombre por defecto y dentro de ellos se almacenan los parámetros de cada variable, la fecha y hora de registro.

Como se muestra en la figura 20 los datos están almacenados, dentro de la base de datos.

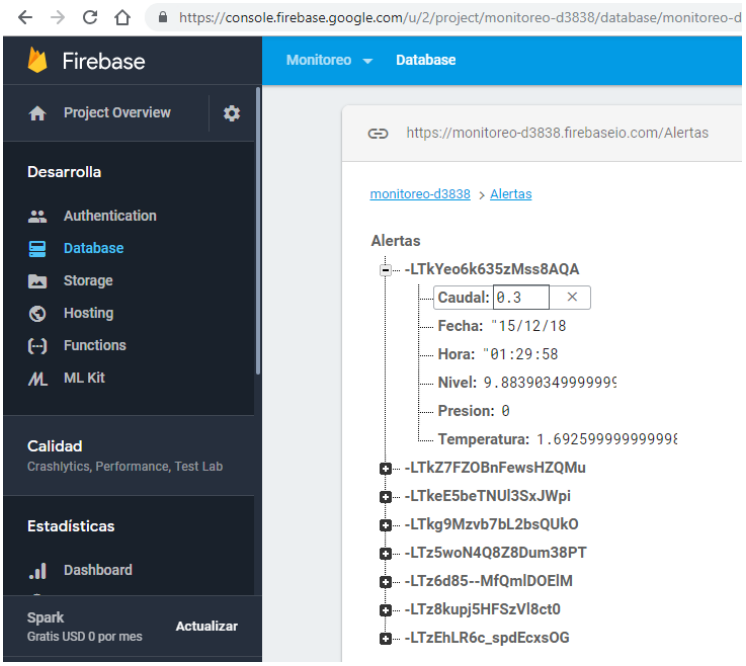

Figura 20: Base de datos.

3.2.2. Reporte de datos

El reporte de los datos se genera a manera de informe, y este es enviado mediante un correo electrónico a las direcciones que estén programadas, el contenido del reporte es el siguiente:

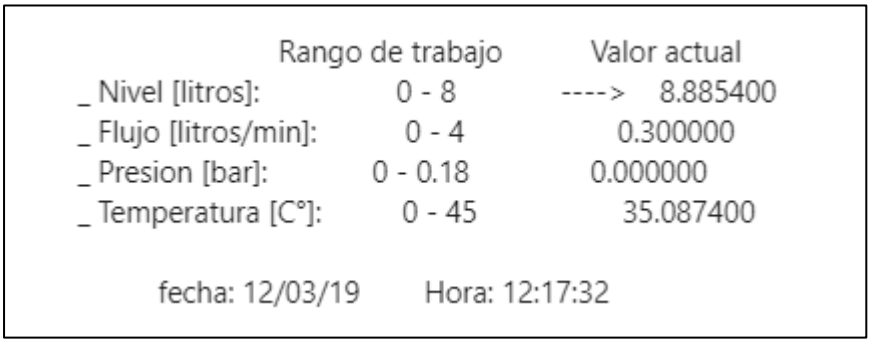

Figura 21. Reporte de fallas.

3.2.3. Envío de correo y mensaje de texto

Una vez que el sistema detecta que se excedieron los niveles, el programa de Raspberry como de Arduino enviarán alertas tanto por correo, como por SMS.

Dentro de la programación de Python se indica el o las direcciones a los que se enviará el correo con los datos como se muestra en la figura 22, para su posterior verificación.

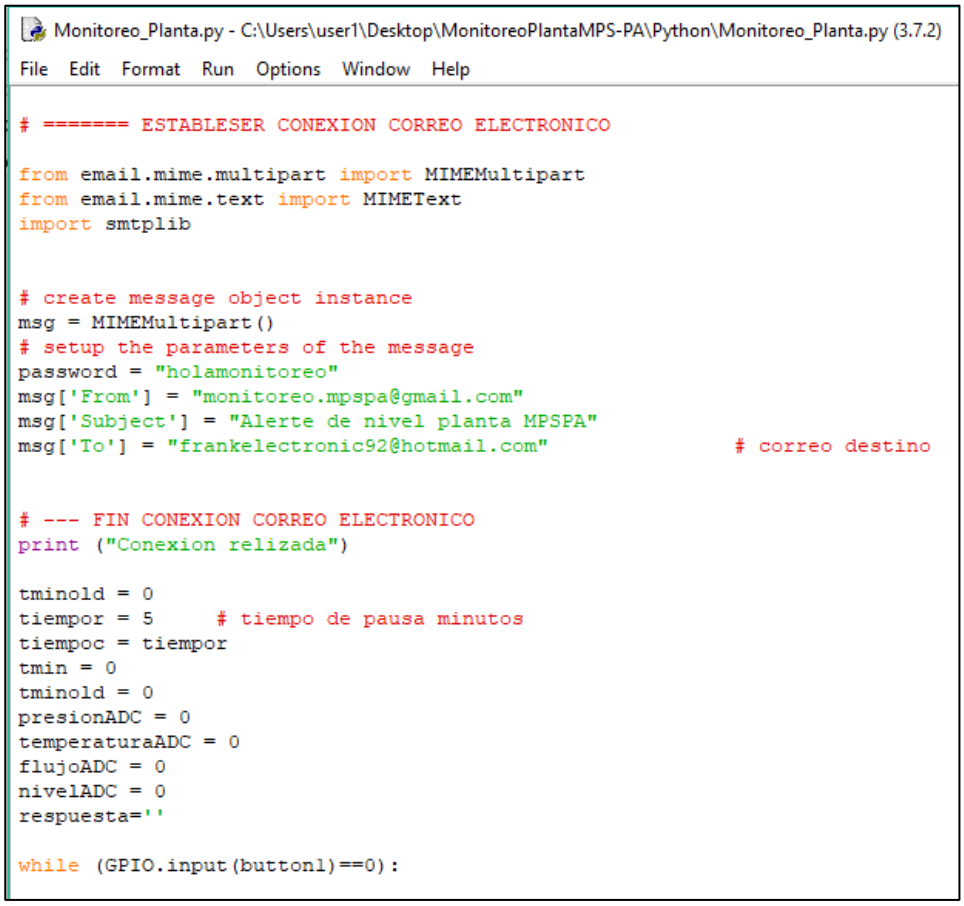

Figura 22. Configuración de envío de correo mediante las librerías SMTP en Python.

Mientras que el envío de mensajes de texto se realiza con Arduino y con el módulo Shield 900, dentro de la programación se puede colocar los números de teléfonos a los que queremos que lleguen los mensajes y el contenido de estos, en este caso de escribe un mensaje por cada variable monitoreada, como se muestra en las figuras 23 y 24.

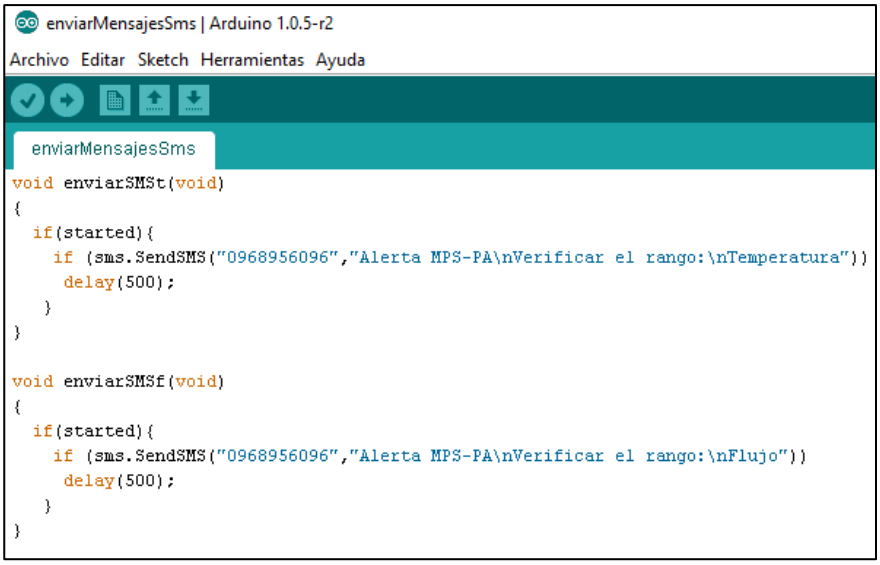

Figura 23. Programación de envío de mensajes, para las variables temperatura y flujo.

Figura 24. Programación de envío de mensajes, para las variables presión y nivel.

Para el desarrollo se han utilizado los siguientes métodos y técnicas.

Método Experimental: Se realizarán las pruebas respectivas con las tarjetas, Arduino y Raspberry, para comprobar su funcionamiento. Una vez realizado

esto se pondrá en marcha la implementación del diseño propuesto para establecer la comunicación con la planta y obtener los datos deseados.

Método Inductivo: El diseño es un prototipo en menor escala, pero en condiciones similares, y con los elementos necesarios y los parámetros que se requieran se lo puede aplicar de manera real en una Planta Industrial.

Experimental: Se utilizó esta técnica para lograr calibrar las variables con sus respectivas unidades, respecto a los voltajes de salida de 0 a 10v de cada sensor, y así lograr obtener un valor real y medible en cada unidad; el nivel en litros, temperatura en grados centígrados, presión en bar y flujo en litros por minutos.

4. RESULTADOS

4.1. IMPLEMENTACIÓN DEL SISTEMA

Una vez que se concluye el diseño de nuestro sistema se procede a su implementación, como se muestra en las figuras 25, 26 y 27.

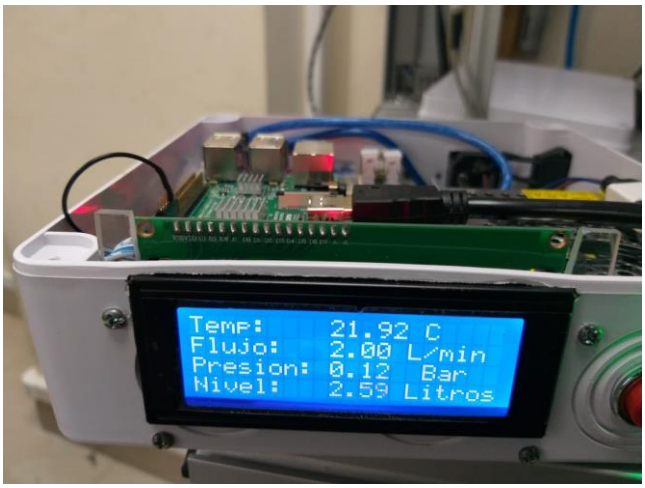

Figura 25. Vista frontal del sistema de monitoreo.

Figura 26. Vista trasera del sistema de monitoreo.

Figura 27. Vista superior del sistema de monitoreo.

El sistema empieza a correr inmediatamente después de ser conectado y encendido, paralelamente se activan las tarjetas Arduino, tanto la que nos sirve de DAQ para Raspberry y la que nos envía los mensajes de texto.

Cuando se encuentre listo el módulo Shield GSM el led que se encuentra en el sistema cambiará de Rojo a verde (figura 28) y enviará un mensaje de texto de inicialización GSM.

Figura 28. Led indicador que el módulo GSM se encuentra activo.

Mientras que la Raspberry ya se encuentra receptando los datos que le envía la otra tarjeta Arduino, los mismos que llegan en forma de tren de datos, teniendo cuatro dígitos para cada variable, esto se puede verificar desde una pantalla la ejecución del programa en Python y como se muestra en la figura 29, llegan 16 dígitos, cada uno con un intervalo de 0.3 segundos y posteriormente se concatenan para obtener el valor real de cada una de las cuatro variables.

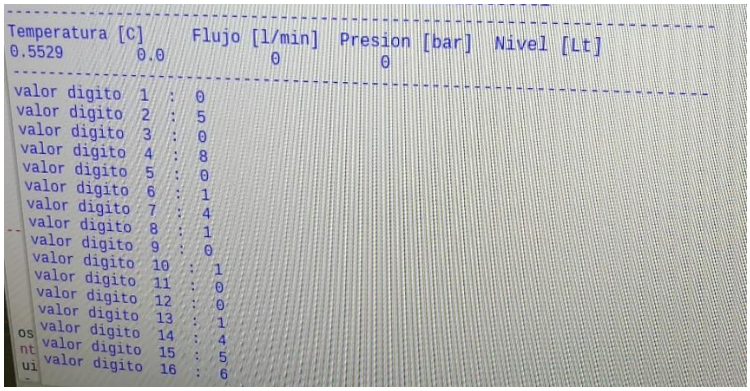

Figura 29. Dígitos de datos.

Los datos se seguirán enviando en tiempo real según el funcionamiento de la planta y el cambio que presente cada variable, hasta el momento en que sistema detecte que una o más variables excedieron en los rangos permitidos y nos enviará una alerta por mensaje indicando la variable y por correo un reporte detallado de los valores.

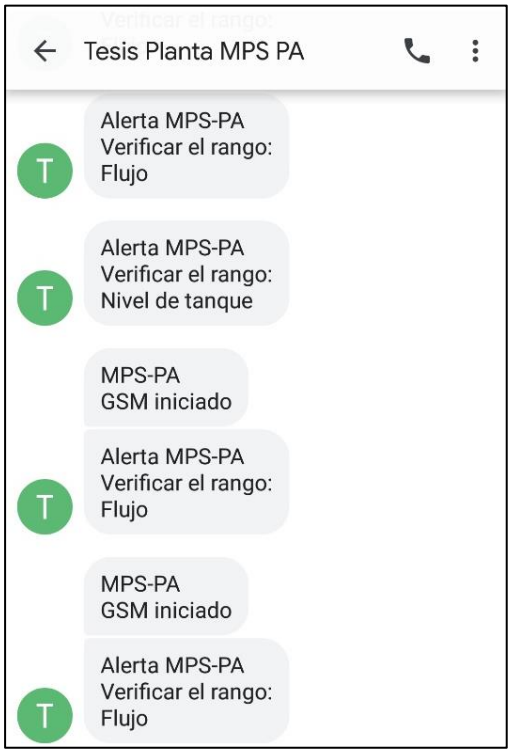

Figura 30. Mensajes de texto indicando las fallas.

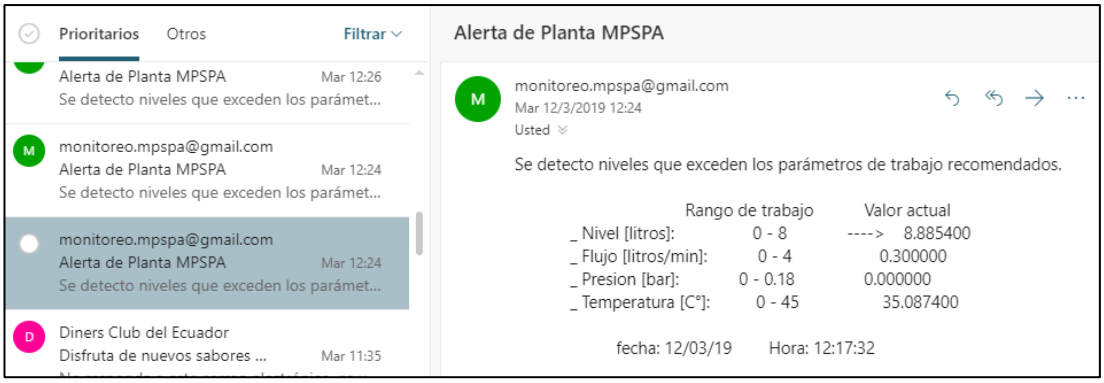

Figura 31. Correo electrónico indicando las fallas.

Al mismo tiempo que nos envía el reporte, éste se almacena en la base de datos de Firebase, donde se podrá acceder en cualquier momento para revisar datos históricos de las fallas y todos los registros almacenados, con las siguientes credenciales de acceso:

Usuario: monitoreo.mpspa@gmail.com

Contraseña: holamonitoreo

Figura 32. Registro de fallas en la base de datos.

5. ANALISIS DE RESULTADOS

5.1. ADQUISICIÓN DE DATOS

Como se había mencionado anteriormente para la obtención de datos se utilizó el cable con doble derivación DB15, una vez que se comprobaron mediante pruebas de continuidad y que se analizaron los voltajes de salida mismos que concuerdan con la especificación técnica del manual (Anexo 2) dónde indica que los voltajes de salida de los sensores analógicos tienen un rango de 0 a 10v.

Por lo que fue necesario escalar estos valores de manera que puedan ser receptados por las tarjetas Arduino, que solo soporta valores de 0 a 5v.

Para esto se utilizaron los módulos reguladores de voltaje y así lograr una salida con un voltaje menor.

En la gráfica se puede apreciar los 4 módulos reguladores AVR.

Figura 33. Módulos Reguladores de voltaje AVR.

Teniendo los voltajes ya reducidos en menor escala, podemos ingresarlos a las tarjetas Arduino, mismo que transformará los datos de Analógicos a Digital tanta para Raspberry como para el módulo GSM Shield Sim900.

5.2. ENLACE DE COMUNICACIÓN

Cuando la Raspberry y el módulo GSM reciben la información en forma digital se obtienen valores medibles de acuerdo a los bits de resolución, en este caso de 0 a 1023, es decir un valor ADC que no representa aún al valor real y medible.

Figura 34. Valores ADC recibidos por Raspberry

Cada valor ADC consta de cuatro dígitos que llegan a manera de tren de datos, hasta conformar un bloque concatenado.

5.3. INTERFAZ GRÁFICA

Para la interfaz gráfica se utilizó el software de Labview y además de adicionó una pantalla LCD para mostrar los datos reales, pero para la visualización se transformaron los valores en ADC, esto mediante mediciones de la variable real versus el voltaje que nos arrojaba en la salida reducida de cada módulo AVR.

Para el nivel se trabaja con centímetros gracias a la regla adaptada en el tanque 102, pero para una mejor visualización se decidió llevar estos valores a su unidad equivalente en volumen, es decir, Litros.

Para la variable de Temperatura se utiliza como unidades los grados centígrados, para el flujo litros por minutos y para la presión su unidad son los BAR.

Para obtener cada variable real se aplicaron las siguientes ecuaciones de las curvas características, cuya información puede ser vista a más detalle en el Anexo 3.

Nivel:

$$y = 0.047x - 0.0493 \quad (1)$$

Temperatura:

$$y = 0.0429x + 0.5529 \quad (2)$$

Flujo:

$$y = 0.053x \quad (3)$$

Presión:

$$y = 0.0002x - 0.0585 \quad (4)$$

5.4. BASE DE DATOS

La base de datos fue realizada con la ayuda de Firebase que se utiliza con la cuenta de Google, se creó un nuevo proyecto con todas las especificaciones y los datos que teníamos que almacenar, y que posteriormente se enlazaron con la programación de Python a través de las librerías SMTP.

Además, cabe recalcar que desde nuestra base de datos no sólo se visualizan los registros de las fallas, sino que también nos permite ver en tiempo real los valores de cada variable que recibe la raspberry.

Y almacenará los registros de fallas, cuando una de variables exceda los límites programados, con su respectiva hora y fecha del evento, a manera de registro histórico.

5.5. REPORTE DE FALLAS

El reporte se envía por correo electrónico; toma los registros de la base de datos y los envía de manera más clara y ordenada, indicando que variable excedió en el rango de trabajo, el rango de trabajo apropiado de cada una, la fecha y la hora en que se registró el evento.

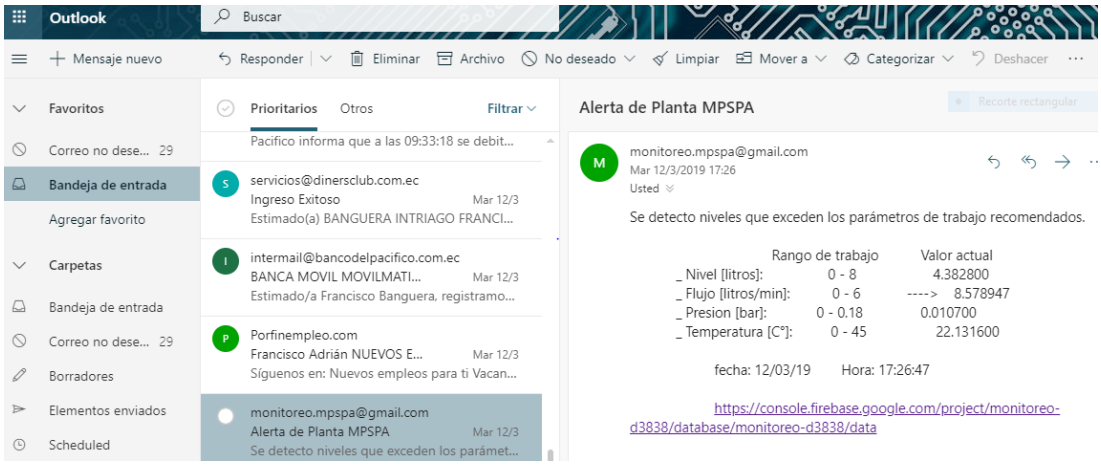

Figura 35. Bandeja de entrada de correo.

5.6. COMUNICACIÓN SMTP Y GSM

Mediante librerías de SMTP y JSON en Python se programa en la Raspberry el envío de correo electrónico automáticamente después de emitir una alerta de falla, a los correos que se registraron previamente.

De la misma forma en el software de Arduino mediante las librerías GSM se programa el envío de mensaje de texto indicando la alerta y cuál de las variables fue la que emitió la misma.

El tiempo de respuesta de cada mensaje de correo y de texto, se le da un tiempo de espera hasta volver a emitir una nueva alerta, esto sirve para poder revisar el motivo de la falla y tomar las acciones necesarias en caso de que se requiera.

CONCLUSIONES

- Al culminar nuestro proyecto de titulación se concluyó que gracias al uso frecuente de las tarjetas embebidas, como lo son Arduino Nano, Raspberry Pi y el módulo GSM se logró adquirir los datos a analizar (nivel, presión, temperatura y caudal) de la planta didáctica MPA PS Compact Workstation de Festo, para ello se estableció una conexión paralela con el PLC de la planta, se diseñó un cable DB15 con derivación en “Y" o ramificación en “Y" para obtener de forma directa los voltajes correspondientes a cada variable, así mismo con estos datos graficar su curva característica.

- Con la implementación del nuevo diseño del cable DB15 con derivación en “Y" logramos una comunicación directa del PLC de la planta hacia el Arduino Nano, que a tiempo real realizaba una lectura completa de la información de las variables en intervalos de tiempo. Luego de ello se conectó el Arduino a la Raspberry Pi, que bajo codificación en Python y conectando a otro Arduino Nano, se logró la comunicación con el módulo GSM.

- Gracias al software LabVIEW y a su lenguaje de programación gráfica logramos diseñar una interfaz de usuario para visualizar el comportamiento de las cuatro variables que se están analizando en el proyecto, y por ende para comprobar el correcto funcionamiento de la planta y calibración de las variables.

- Se concluyó que gracias a los servidores Web, se logra almacenar datos en la Internet, para nuestro proyecto fue necesario crear una base de datos en Firebase de Google, en donde se almacenan los reportes de fallas que se generaban durante el proceso de la planta industrial, los mismos que se enviaban de forma más detallada por correo electrónico e-mail y cómo mensajes de alerta por medio de mensajería instantánea SMS.

RECOMENDACIONES

Se recomienda vaciar los tanques y tuberías de la Planta cuando vaya a estar sin uso por un tiempo largo, ya que se pudo detectar que el filtro de la bomba estaba tapado con lama y esto impedía el flujo adecuado de agua en las tuberías y el tiempo de llenado era demasiado lento.

El sistema funciona indistintamente del uso o de las prácticas que los estudiantes estén desarrollando, por lo que podría incluso utilizarse para estudio de materias de telecomunicaciones, ya que se pueden desarrollar conceptos de protocolos de comunicación, adquisición de datos, envío de información mediante trenes de datos y redes GSM.

Para futuras investigaciones se podría crear una página web dónde se visualicen los registros almacenados en la base de datos de Firebase.

REFERENCIAS BIBLIOGRÁFICAS

Arduino. (2015). *Escudo Shield Arduino*. Recuperado el 19 de 09 de 2015, de productos: https://www.arduino.cc/en/Main/ArduinoGSMShield

Arduino. (19 de 09 de 2015). *Genuino Uno*. Recuperado el 19 de 09 de 2015, de Productos: https://www.arduino.cc/en/Main/ArduinoBoardUno

Arduino. (2018). Obtenido de https://store.arduino.cc/usa/arduino-mega-2560-rev3

Dueñas, F., & Chalacán, N. (2017). *Diseño, implementación de monitoreo de una red, inalámbrica entre dos plantas de didáctica industrial, usando antenas Ubiquiti Networks NanoStation5 y un scada bajo Labview.* Guayaquil.

Duplika. (20 de Septiembre de 2010). *Que son los servidores web y por son necesarios?* Obtenido de https://duplika.com/blog/que-son-los-servidores-web-y-por-que-son-necesarios/

Festo Didactic. (2008). *Compact Workstation Manual and Techinical Documentation.* Alemania.

Idrovo, A., & Peña, C. (2014). Construcción de una plataforma de instrumentación virtual sobre LabVIEW, monitoreo remoto e implementación de controladores PID para las variables de nivel, caudal, temperatura y presión de la planta MPS PA. Compact Workstation . Cuenca.

Martillo, D., & Peña, E. (2015). *Diseño de aplicaciones de sistemas embebidos basados en tecnología Raspberry-Pi y Odroid-U3 (trabajo de titulación).* Guayaquil: Universidad Politécnica Salesiana.

Monsalve Posada, J. F., Arias Londoño, A., & Mejía Arango, J. G. (2015). Desempeño de redes inalámbricas y redes industriales inalámbricas en procesos de control en tiempo real bajo ambientes industriales. *18(34)*, 87-99. Tecno Lógicas.

Programarfacil.com. (2018). Obtenido de https://programarfacil.com/blog/raspberry-pi/configurar-un-servidor-web-en-raspberry-pi-con-flask/

Torrente, O. (2013). ARDUINO Curso práctico de formación. En O. Torrente, *Curso práctico de formación* (págs. 63-86). Alfaomega Grupo Editor S.A de C.V.

Upton, E., & Halfacree, G. (2013). En *Raspberry Pi Guía de Usuario* (págs. 2-15,19). Anaya Multimedia. Obtenido de http://www.raspberryshop.es/guia-completa-raspberry-pi.php

ANEXOS

A1. PRESUPUESTO

Materiales	Cantidad	Precio Unitario	Precio Total
Tarjeta Arduino Nano	2	$ 15,00	$ 30,00
Modem GSM	1	$ 40,00	$ 40,00
Raspberry Pi 3	1	$ 100,00	$ 100,00
Caja de Raspberry Pi CASE	1	$ 10,00	$ 10,00
Mouse pequeño genérico	1	$ 15,00	$ 15,00
Teclado genérico	1	$ 25,00	$ 25,00
Cable USB a USB	2	$ 15,00	$ 30,00
Micro SD 32 GB	1	$ 40,00	$ 40,00
Monitor con HDMI	1	$ 250,00	$ 250,00
Cable HDMI	2	$ 15,00	$ 30,00
Conectores hembra y macho	30	$ 1,00	$ 30,00
Pantalla LCD	1	$ 20,00	$ 20,00
Estructura (banco de trabajo)	1	$ 100,00	$ 100,00
Gastos Varios	1	$ 300,00	$ 300,00
Materiales varios	1	$ 200,00	$ 200,00
Total*			**$ 1.220,00 ***

**Valor cubierto por los autores*

Tabla 1. Presupuesto.

A2. DATOS TÉCNICOS DE LA MPS PA. COMPACT WORKSTATION DE FESTO.

Parámetro	Valor
Máxima presión de funcionamiento para el sistema de tuberías	50 kPa (0.5 bar)
Suministro de energía para la estación	24V
Dimensiones	700 x 700 x 907 mm
Tasa de flujo de la bomba	5 l/min
Volumen máximo del tanque	10 l
Sistema de tubería flexible	DN10 (= 15mm)
Entradas digitales	7
Salidas digitales	5
Entradas analógicas	4
Salidas analógicas	2
Cantidad de tanques	3
Rango de control para la bomba	0. . . 10V
Rango de control para válvula proporcional 2/2w	0. . . 10V
Elemento calefactor de 230V	On/Off (Relé)
Rango de trabajo en lazo cerrado para control de nivel	0...10 l mm
Rango de medición del sensor de nivel	0. . . 9 l
Señal de salida para el sensor de nivel	Corriente de 4-20mA
Rango de trabajo en lazo cerrado para control de flujo	0...7 l/min
Rango de medición del sensor de flujo	0,3. . . 9 l/min
Señal del sensor de flujo	0. . . 1200Hz
Rango de trabajo en lazo cerrado para control de presión	0...30 kPa (0...300 mbar)
Rango de medición del sensor de presión	0. . . 10 kPa (0. . . 100 mbar)
Señal del sensor de presión	0. . . 10V
Rango de trabajo en lazo cerrado para control de temperatura	0...60° C
Rango de medición del sensor de temperatura	-50°C... 150°C
Señal del sensor de temperatura	Resistencia PT100

Tabla 2. Datos técnicos de MPS PA. Compact. *(Festo Didactic, 2008)*

A3. CURVAS CARACTERÍSTICAS PARA LA CALIBRACIÓN DE LAS VARIABLES.

Para calibrar las variables, respecto a los voltajes obtenidos en los terminales del conector DB15 se utilizó el método experimental al medir el voltaje que arrojaba vs el valor real que marcaba cada sensor y así se obtuvieron las curvas características y la ecuación para cada una, y así tener la misma información en físico, en la interfaz Labview y en la pantalla de nuestro sistema.

En la variable de Nivel por conveniencia y para una mujer visualización se decidió trabajar en función del volumen o capacidad del tanque, cuya unidad está dad en litros.

A continuación, se presentan las tablas de datos y las gráficas obtenidas.

Voltaje (mv)	Nivel (Litros)
0	0
89	0,5
210	1
330	1,5
449	2
549	2,5
657	3
777	3,5
890	4
1104	5
1309	6
1520	7
1715	8
1890	9

Tabla 3. Datos de Nivel.

Cm	Litros
1,4	1
4,8	2
8,4	3
11,66	4
15,1	5
18,3	6
21,4	7
24,2	8
27,3	9

Tabla 4. Relación cm vs litros.

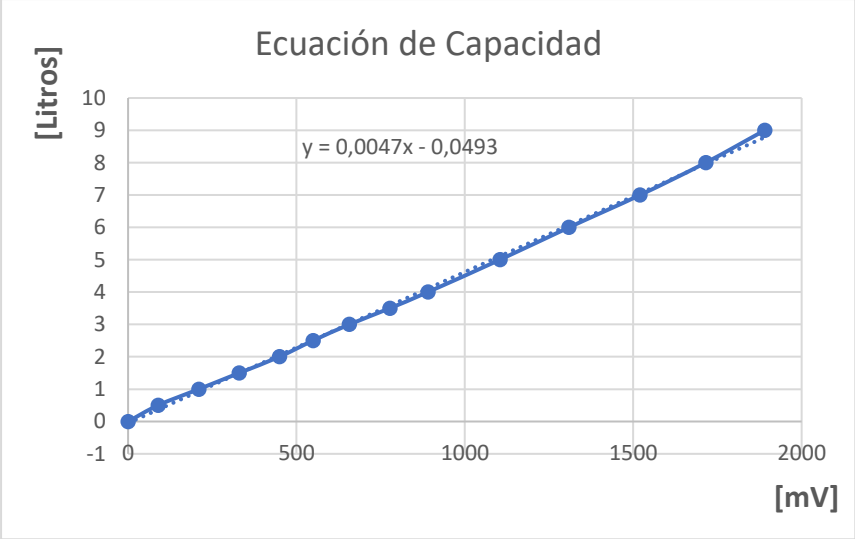

Figura 36. Ecuación de Capacidad.

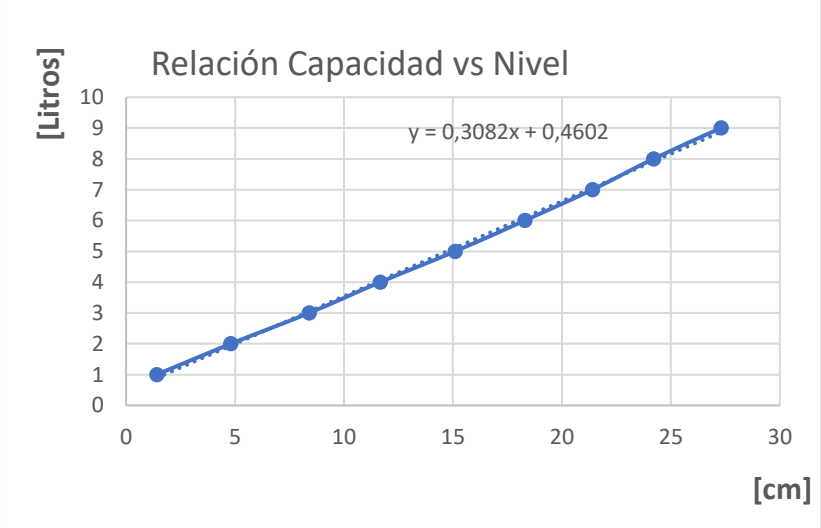

Figura 37. Ecuación de Nivel.

Voltaje (mv)	Temperatura (°C)
518	22,7
720	31,98
745	32,61
752	33,1
796	34,76
811	35
821	35,74

840	36,5
855	37,17
874	38
889	38,62
910	39,5
948	41,28
972	42
987	42,77
1031	45
1045	45,5
1085	47,12
1100	47,7
1114	48,43
1130	49,26

Tabla 5. Datos de Temperatura.

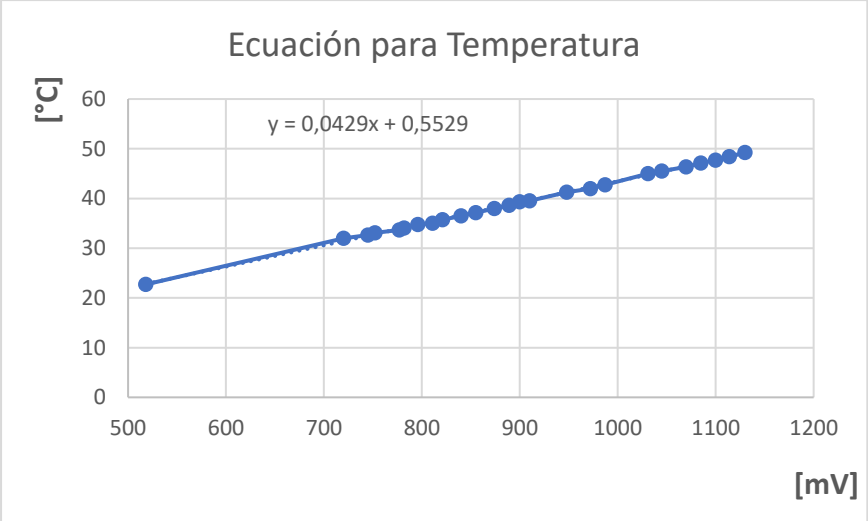

Figura 38. Ecuación de Temperatura.

Voltaje (mv)	Flujo (L/min)
637	3,35
601	3,16
563	2,96
520	2,74
500	2,63
478	2,52

486	2,56
468	2,46
436	2,29
398	2,09
369	1,94
312	1,64
295	1,55
250	1,32
198	1,04
147	0,77
101	0,53

Tabla 6. Datos de Flujo.

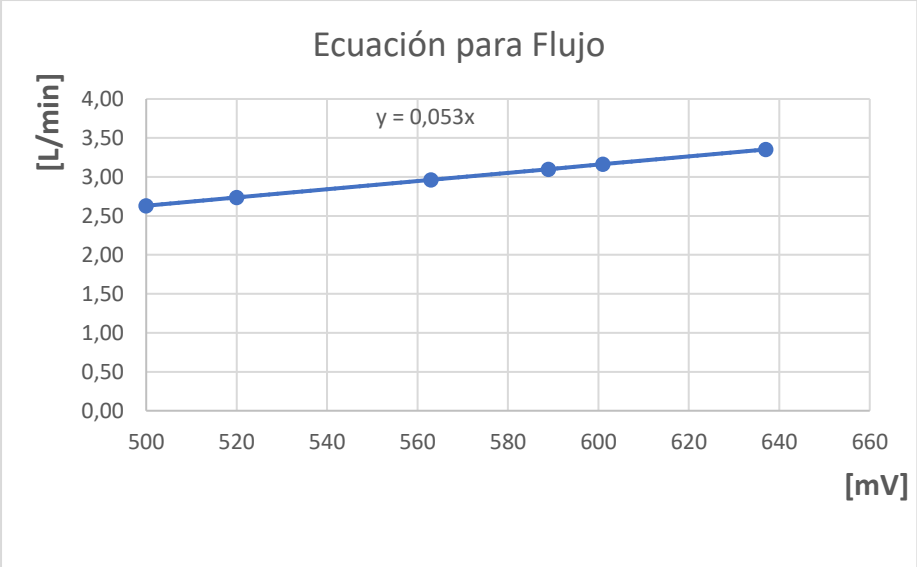

Figura 39. Ecuación de Flujo.

Voltaje (mv)	Presión (BAR)
1150	0,21
1100	0,2
998	0,18
921	0,16
821	0,14
733	0,12
637	0,1

548	0,075
458	0,05
378	0,025
260	0

Tabla 7. Datos de Presión.

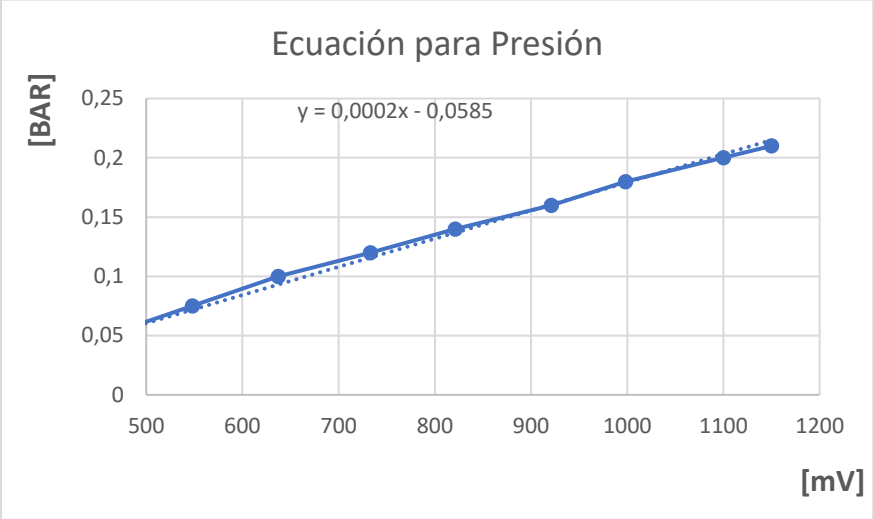

Figura 40. Ecuación de Presión.

A4. DISEÑO DE PLACA PARA MONTAJE DE LAS TARJETAS ARDUINO Y RASPBERRY.

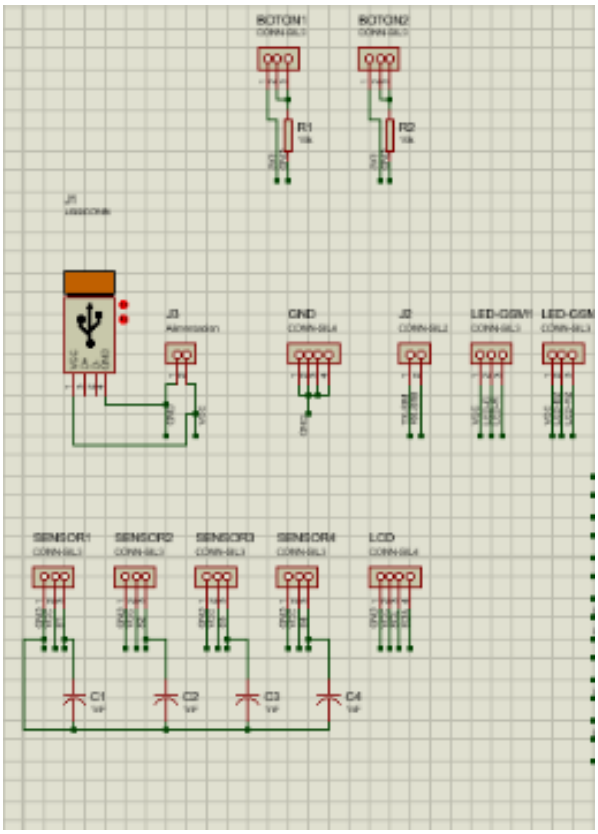

Figura 41. Diagrama esquemático en Proteus (1).

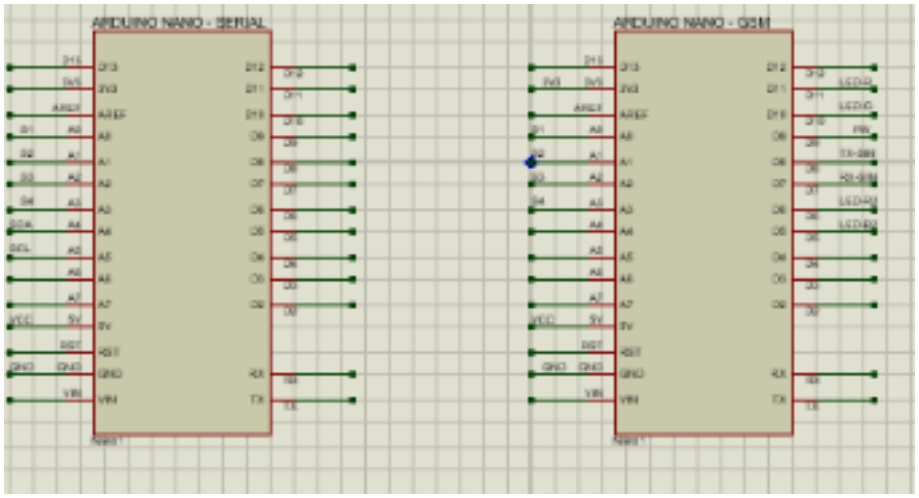

Figura 42. Diagrama esquemático en Proteus (2).

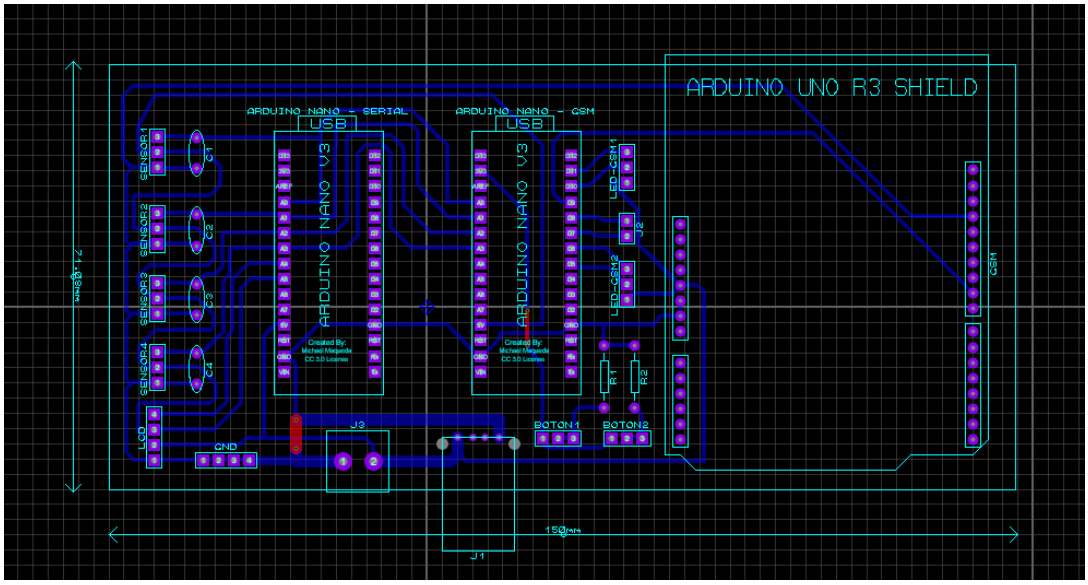

Figura 43. Diseño para el circuito impreso.

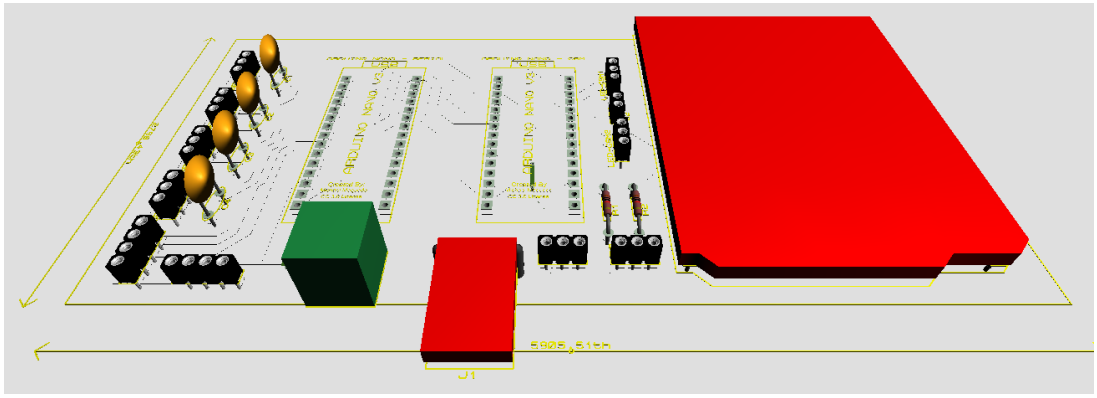

Figura 44. Diagrama en 3D con los elementos.

A5. CÓDIGO FUENTE EN PYTHON.

```
#=====> IMPORTAR LIBRERIAS <============
import os
import serial
import time
import firebase_admin
import RPi.GPIO as GPIO
from datetime import datetime
#--- fin importar librerias ----------

#=====> ESTABLECER CONFIGURACION GPIO <===
GPIO.setmode(GPIO.BOARD)
button1 = 7  # se dfine la variable en el pin 7
GPIO.setup(button1,GPIO.IN,pull_up_down=GPIO.PUD_UP) #Se define
button1 comoentrada y se activa la resistencia  pull up
pull_up_down=GPIO.PUD_UP

#=====> ESTABLECER CONEXION ARDUINO <====
print ("Conectado Arduino...")
#arduino.open()
arduino =  serial.Serial('/dev/ttyACM0', baudrate=115200)

#=====> ESTABLECER CONEXION BASE DE DATOS FIREBASE<====
from firebase_admin import credentials
from firebase_admin import db

print ("Conectado Base Datos...")
cred = credentials.Certificate(
   '/home/pi/MonitoreoPlantaMPS-PA/Python/monitoreo-d3838.json')
firebase_admin.initialize_app(cred,
                        {'databaseURL'
```

```
                    :'https://monitoreo-d3838.firebaseio.com/'})

monitoreodb = db.reference('Alertas')
datosdb = db.reference('Datos')
#---- FIN CONEXION DASE DE DATOS

# ======= ESTABLESER CONEXION CORREO ELECTRONICO

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib

# create message object instance
msg = MIMEMultipart()
# setup the parameters of the message
password = "holamonitoreo"
msg['From'] = "monitoreo.mpspa@gmail.com"
msg['Subject'] = "Alerta planta MPSPA"
msg['To'] = "frankelectronic92@hotmail.com"             # correo destino


# --- FIN CONEXION CORREO ELECTRÓNICO
print ("Conexion relizada")

tminold = 0
tiempor = 5     # tiempo de pausa minutos
tiempoc = tiempor
tmin = 0
tminold = 0
presionADC = 0
temperaturaADC = 0
```

```
flujoADC = 0
nivelADC = 0
respuesta=''

while (GPIO.input(button1)==0):

# ====> conexion leer sensores con arduino <======================

    # es neceario convertir los datos enviados a binario
    # para la comunicacion serial, por lo que se antepone
    # b --- op = b'T'



    arduino.write(b'T')
    time.sleep(0.3)
    d = 10
    valor = 0
    respuesta = 0
    temperaturaADC = 0
    presionADC = 0
    flujoADC = 0
    nivelADC = 0
    while arduino.inWaiting() > 0:
        valor = valor +1
        respuesta = int(arduino.read(1))
        if valor <5:
            temperaturaADC = temperaturaADC*d + respuesta
        if valor <9:
            presionADC = presionADC*d + respuesta
        if valor <13:
            flujoADC = flujoADC*d + respuesta
```

```
            if valor > 12:
                nivelADC = nivelADC*d + respuesta


            time.sleep(0.03)
        time.sleep(0.7)
# ------ fin leer sensores -------------------------
#======> calculo d variables <======

        temperatura = 0.0429*temperaturaADC - 0.5529
        flujo = 0.5326*flujoADC
        presion = 0.0002*presionADC - 0.0585
        nivel = 0.0047*nivelADC - 0.0493

        datosdb.set({'Temperatura':temperatura,'Flujo':
flujo,'Presion':presion,'Nivel':nivel,'Online':1})
#======> monitorear rango de funcionamiento <======

        sn = '   '
        sf = '   '
        sp = '   '
        st = '   '

        tiempo = datetime.now()
        tmin = tiempo.minute
        if (tmin != tminold):
                tminold = tmin
                tiempoc = tiempoc +1
                if(tiempoc > tiempor+1):
                        tiempoc = tiempor+1

        if(((temperatura > 40)or(flujo > 6)or(presion > 0.18)or(nivel >
8))and(tiempoc >= tiempor)):
```

```
        if(temperatura > 40):
            st = '-->'

        if(flujo > 6):
            sf = '-->'

        if(presion > 0.18):
            sp = '-->'

        if(nivel > 8):
            sn = '-->'

#---- crear registro en la base de datos
        hora = time.strftime("%H:%M:%S")
        fecha = time.strftime("%d/%m/%y")

monitoreodb.push({'Fecha':fecha,'Hora':hora,'Temperatura':temperatura,'Fluj
o': flujo,'Presion':presion,'Nivel':nivel})

#---- enviar correo de alerta de

        message = """Se detecto niveles que exceden los parámetros de
trabajo recomendados.

                        Rango de trabajo        Valor actual
        _ Nivel [litros]:           0 - 8          %s    %f
        _ Flujo [litros/min]:       0 - 6          %s    %f
        _ Presion [mbar]:           0 - 0.18       %s    %f
        _ Temperatura [C°]:         0 - 40         %s    %f

                fecha: %s       Hora: %s
```

```
https://monitoreo-d3838.firebaseapp.com
"""%(sn,nivel,sf,flujo,sp,presion,st,temperatura,fecha,hora)

# add in the message body
msg.attach(MIMEText(message, 'plain'))
#create server
server = smtplib.SMTP('smtp.gmail.com: 587')

server.starttls()

# Login Credentials for sending the mail
server.login(msg['From'], password)

# send the message via the server.

server.sendmail(msg['From'],'frankelectronic92@hotmail.com',
msg.as_string())
time.sleep(1);
server.sendmail(msg['From'],'madelce91@hotmail.com',
msg.as_string())
server.quit()



print('---------- Alerta de nivel detectado ----------')
print ('Fecha ','Hora ','Temperatura [C] ','Caudal [l/min]
','Presion [mbar] ','Nivel [Lt] ')
print (fecha,' ',hora,' ',temperatura,' ',flujo,' ',presion,'
',nivel)
print('')

tiempoc = 0
tiempo = datetime.now()
tminold = tiempo.minute
```

```
#------ fin monitoreo
        print ('Temperatura [C]    ','Caudal [l/min] ','Presion [mbar] ','Nivel [Lt] ')
        print (temperatura,'      ',flujo,'         ',presion,'        ',nivel)
        print ("--------------------------------------------------------------------")
datosdb.set({'Temperatura':0,'Flujo':0,'Presion':presion,'Nivel':0,'Online':0})
print ("Programa Terminado")
arduino.close()
#os.system("sudo halt") # comando de apagado del sistema operativo
```

A6. CÓDIGO FUENTE PARA ARDUINO.

Programa Principal

```
//====== Adquisicion de datos ========================
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // */
#include <SoftwareSerial.h>
#include "declaracionPines.h"
//==== S E T U P ==================================
void setup()
{
  definirPines();
  //lcd.begin(20, 4);
  Serial.begin(115200);

  lcd.init();
  lcd.backlight(); // */

  controlLCD = 0;
  lcd.clear();
}
```

```
//==== M A I N ====================================
void loop()
{
  entradaTemperatura();
  entradaPresion();
  entradaFlujo();
  entradaNivel();
  imprimirLCD();
  //recibirSerial();
  delay(300);
```

Lectura de Sensores

```
void entradaTemperatura()
{
  temperatura = analogRead(temperaturaADC);
  temperatura = map(temperatura,0,1023,0,5000);
  temperatura1 =0.0429*temperatura + 0.5529;
  if(temperatura1<0){temperatura1=0;}
}
void entradaPresion()
{
  presion = analogRead(presionADC);
  presion1 = 0.0002*presion-0.0175;
  if(presion1 < 0.1){presion1 = presion1-0.025;}
  if(presion1<0){presion1=0;}
}
void entradaFlujo()
{
  flujo = analogRead(flujoADC);
  flujo1 = map(flujo,0,453,0,7); // prueba funcionamiento potenciometro
  flujo1 = 0.052631*flujo;
  if(flujo1<0){flujo1=0;}
```

```
}
void entradaNivel()
{
  nivel = analogRead(nivelADC);
  nivel = map(nivel,0,1023,0,5000);
  nivel1 = 0.0047*nivel-0.0493;
  if(nivel1<0){nivel1=0;}
}
// ===============> LEER SENSORES <========================
void LeerSensores()
{
    entradaTemperatura();
    entradaPresion();
    entradaFlujo();
    entradaNivel();

    Serial.print("Temperatura: ");Serial.print(temperatura1);
    Serial.print(" Presion: ");Serial.print(presion1);
    Serial.print(" Flujo: ");Serial.print(flujo1);
    Serial.print(" Nivel: ");Serial.println(nivel1);
}
// ===============> CONTROL NIVELES <===================
void controlNivel(void)
{
  if((minSms >= tiempoSms))
  {
    if(temperatura1 > 40){minSms = 0;enviarSMSt();}
    else if(flujo1 > 3){minSms = 0;enviarSMSf();}
    else if(presion1 > 0.19){minSms = 0;enviarSMSp();}
    else if(nivel1 > 8){minSms = 0;enviarSMSn();}
  }
```

```
}
// Timer ------------------
void ControlTiempo(void)
{
    LeerSensores();

    mSms = mSms + 1;
    if(mSms >= 4)
    {
      mSms = 0; segSms = segSms + 1;
      //Serial.print("Tiempo: ");Serial.println(segSms);
    }
    if(segSms >= 60){segSms = 0; minSms = minSms + 1;}
    if(minSms >= tiempoSms){minSms = tiempoSms;}//Serial.println(minSms);
}
          }
```

Lectura Serial

```
void enviarDatos(int n)
{
  int d1 = 0; if(n > 999){ d1 = n/1000; n = n - d1*1000;}
  int d2 = 0; if(n > 99){ d2 = n/100; n = n - d2*100;}
  int d3 = 0; if(n > 9){ d3 = n/10; n = n - d3*10;}
  int d4 = 0; if(n > 0){ d4 = n;}
  Serial.print(d1);
  Serial.print(d2);
  Serial.print(d3);
  Serial.print(d4);
 }
void recibirSerial()
{
switch(c)
 {
```

```
    case 'D':   // envia los datos de los sensores a la raspberry
      enviarDatos(temperatura);
      enviarDatos(presion);
      enviarDatos(flujo);
      enviarDatos(nivel);
      lcd.backlight();
    break;
    case 'T':

    break;
    case 'F':

    break;
    case 'P':

    break;
    case 'N':

    break;
    case 'L':
      lcd.noBacklight(); // turn off backlight
    break;
  }
  c = ' ';
}
void serialEvent()
{
  if (Serial.available())
    {
      c = Serial.read();
        recibirSerial();
```

```
    }
}
```

Envío de Mensaje SMS

```
void powerSim900()
{
  digitalWrite(9,HIGH);
  delay(500);
  digitalWrite(9,LOW);
  delay(5000);
}
void controlSMS(void)
{
  //Serial.println("GSM Shield testing...");    //Start configuration of shield with baudrate.
  //powerSim900();                                  //For http uses is raccomanded to use 4800 or slower.
   started=false;
  if (gsm.begin(9600))
    {
       for(i=1;i<=20;i++)
         {
           sms.DeleteSMS(i);
         }
       //Serial.println("GSM status = OK");
       started=true;
    }
    else
    {
       //Serial.println("GSM status = FAIL \nrepeating GSM Shield testing...");
       powerSim900();
       if (gsm.begin(9600))
         {
```

```
            for(i=1;i<=20;i++)
              {sms.DeleteSMS(i);}
             //Serial.println("\nGSM status = OK");
             started=true;
          }
    }
 ///*
}
void enviarSMSt(void)
{
  if(started){
    if (sms.SendSMS("0968956096","Alerta MPS-PA\nVerificar el
rango:\nTemperatura"))
     delay(500);
   }
}


void enviarSMSf(void)
{
  if(started){
    if (sms.SendSMS("0968956096","Alerta MPS-PA\nVerificar el
rango:\nFlujo"))
     delay(500);
   }
}


void enviarSMSp(void)
{
  if(started){
    if (sms.SendSMS("0968956096","Alerta MPS-PA\nVerificar el
rango:\nPresion"))
     delay(500);
```

```
   }
}
void enviarSMSn(void)
{
  if(started){
    if (sms.SendSMS("0968956096","Alerta MPS-PA\nVerificar el
rango:\nNivel de tanque"))
      delay(500);
   }
}
```

A7. PROGRAMA EN LABVIEW PARA NIVEL Y TEMPERATURA.

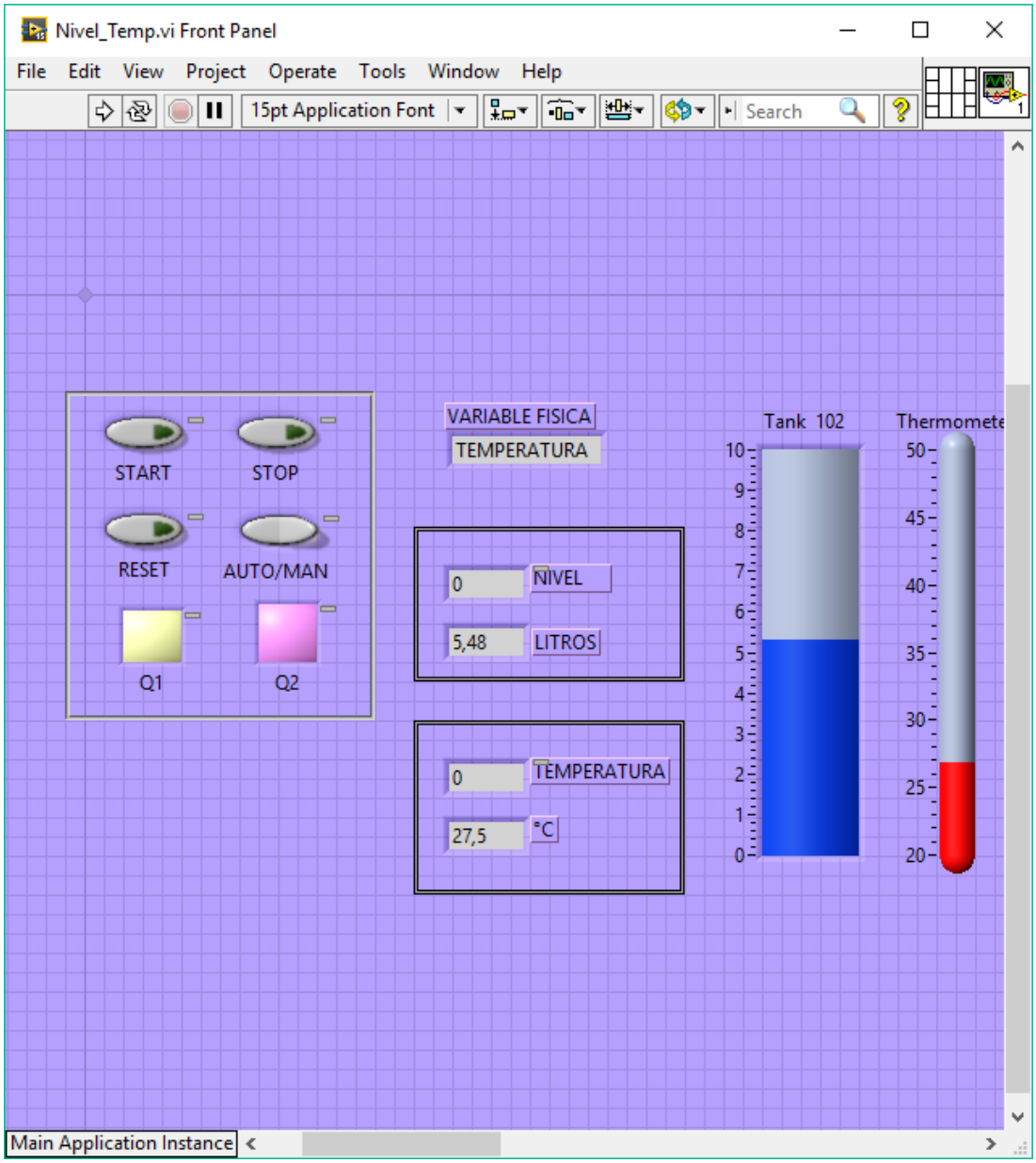

Figura 45. Panel Frontal en Labview para nivel y temperatura.

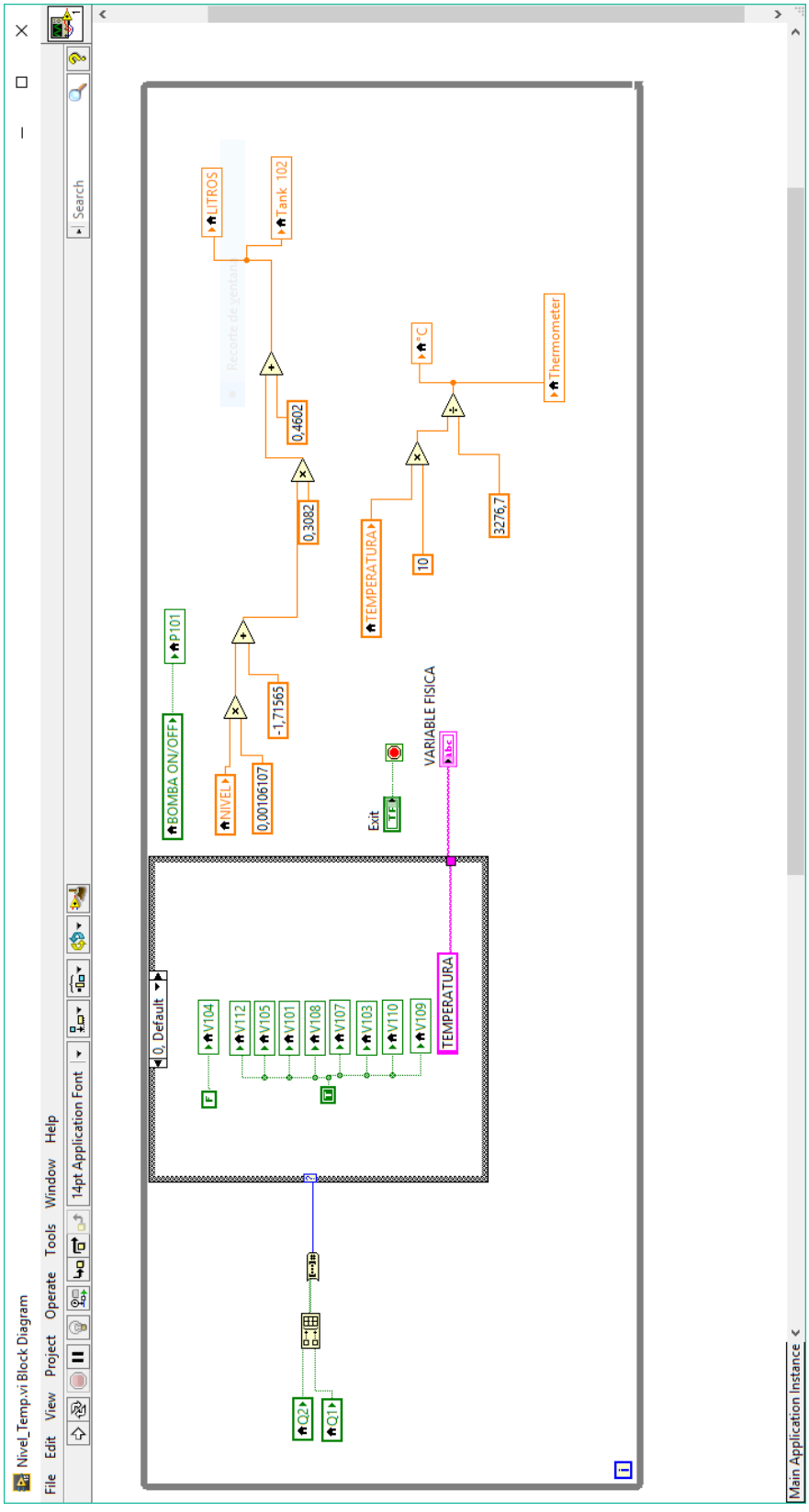

Figura 46. Diagrama de bloques en Labview para nivel y temperatura.

A8. PROGRAMA EN LABVIEW PARA PRESIÓN Y FLUJO

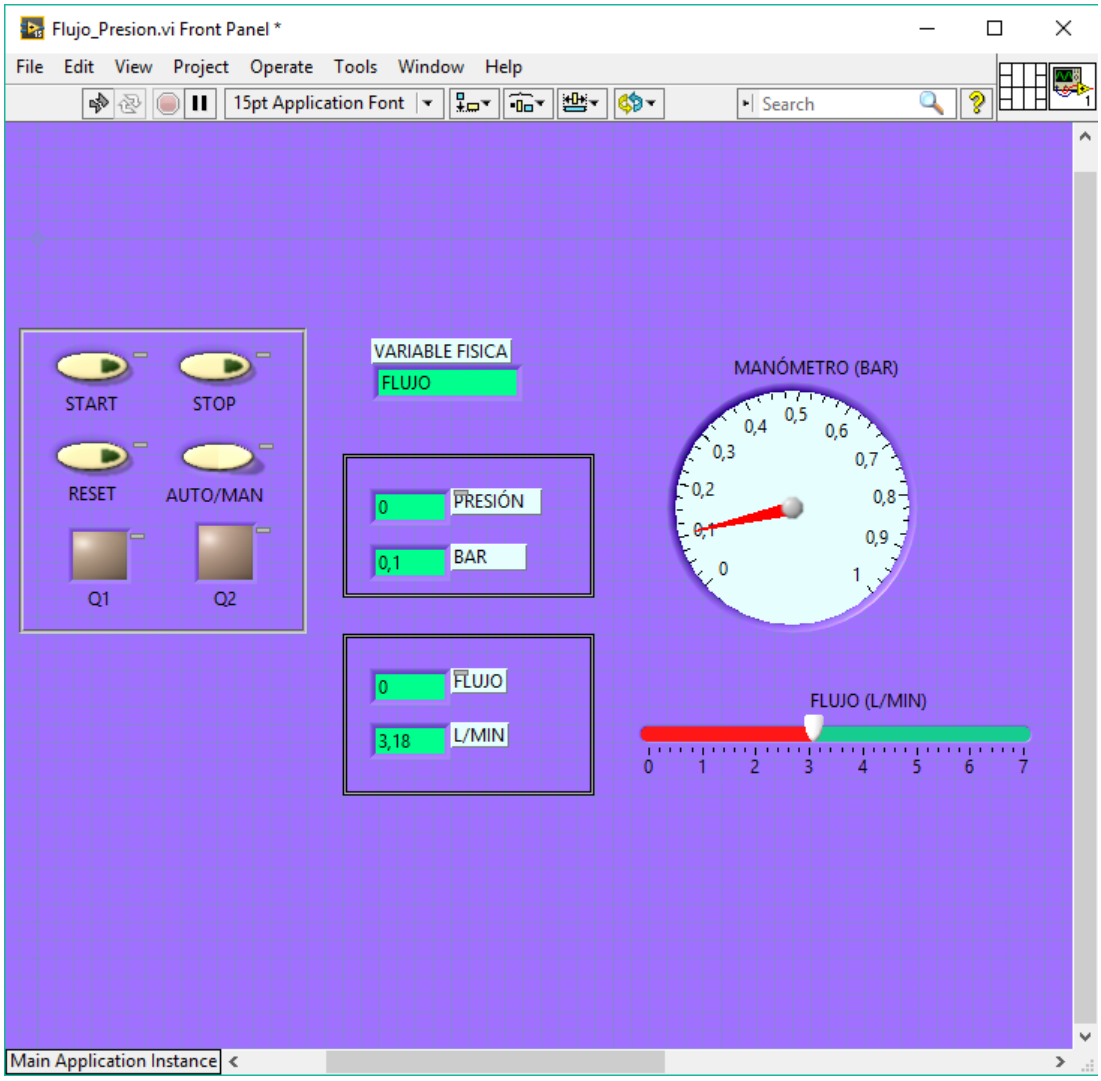

Figura 47. Panel Frontal en Labview para presión y flujo.

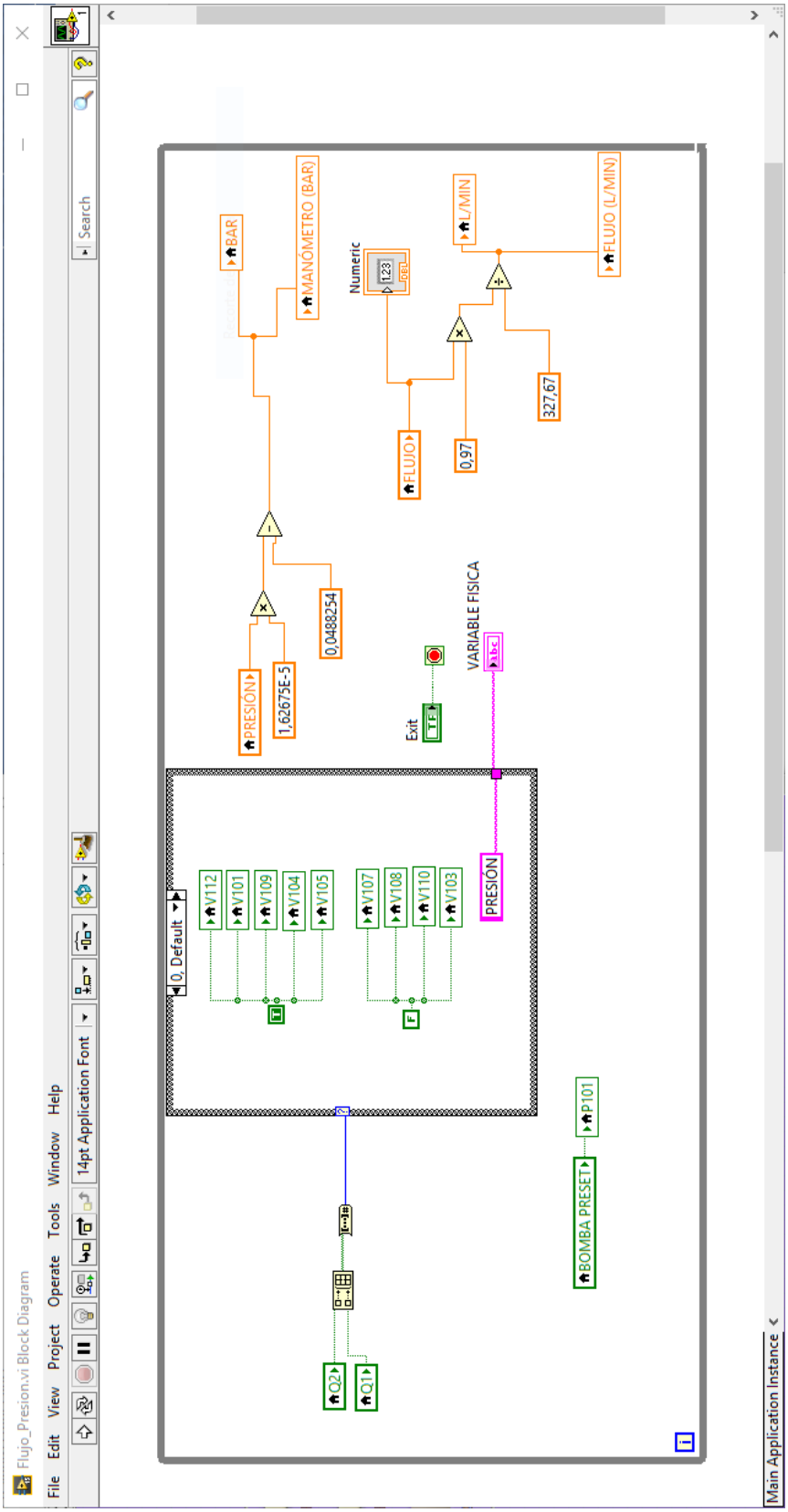

Figura 48. Diagrama de bloques en Labview para presión y flujo

A9. PROGRAMACIÓN EN TIA PORTAL.

Diagrama de Bloques Programa Principal

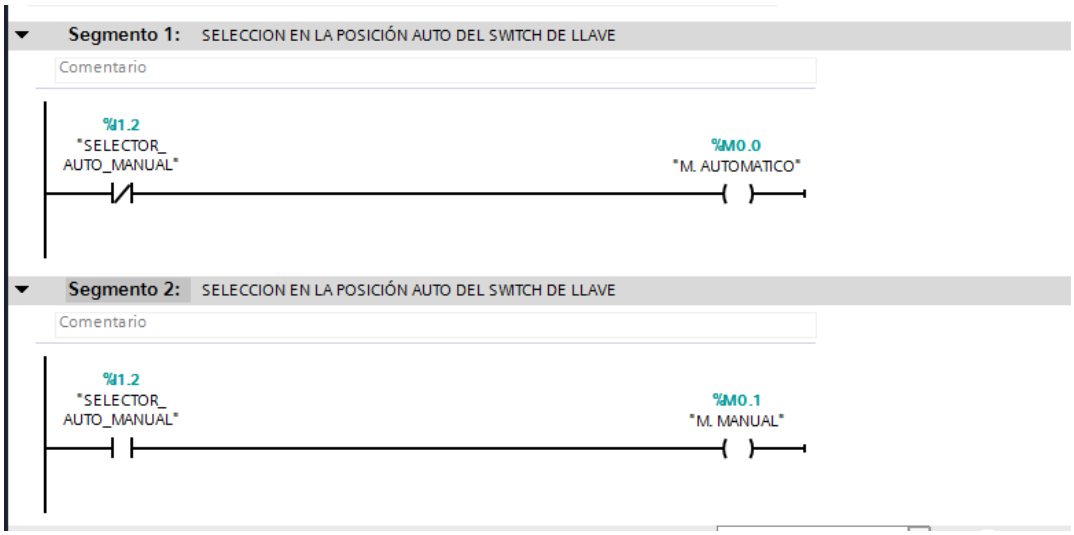

Figura 49. Programa principal (1) *(Dueñas & Chalacán, 2017)*

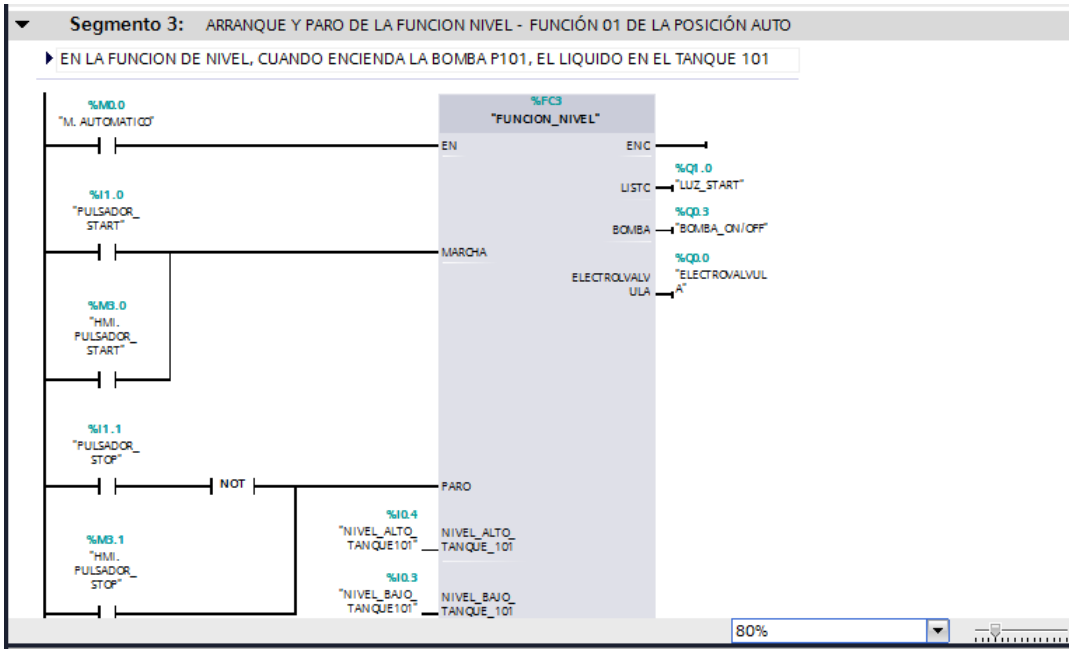

Figura 50. Programa principal (2) *(Dueñas & Chalacán, 2017)*

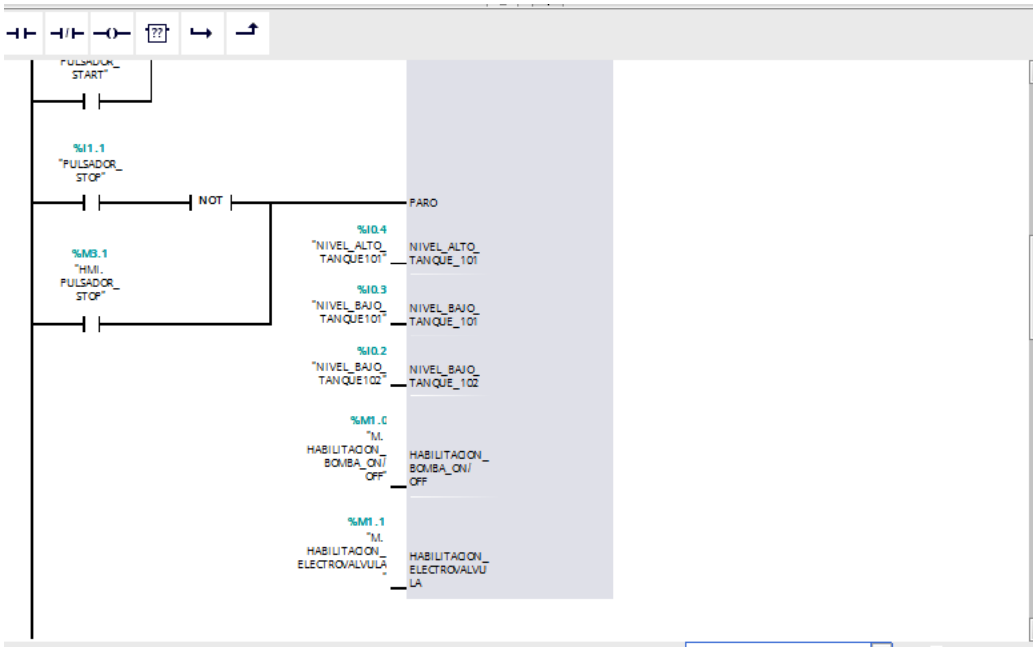

Figura 51. Programa principal (3) *(Dueñas & Chalacán, 2017)*

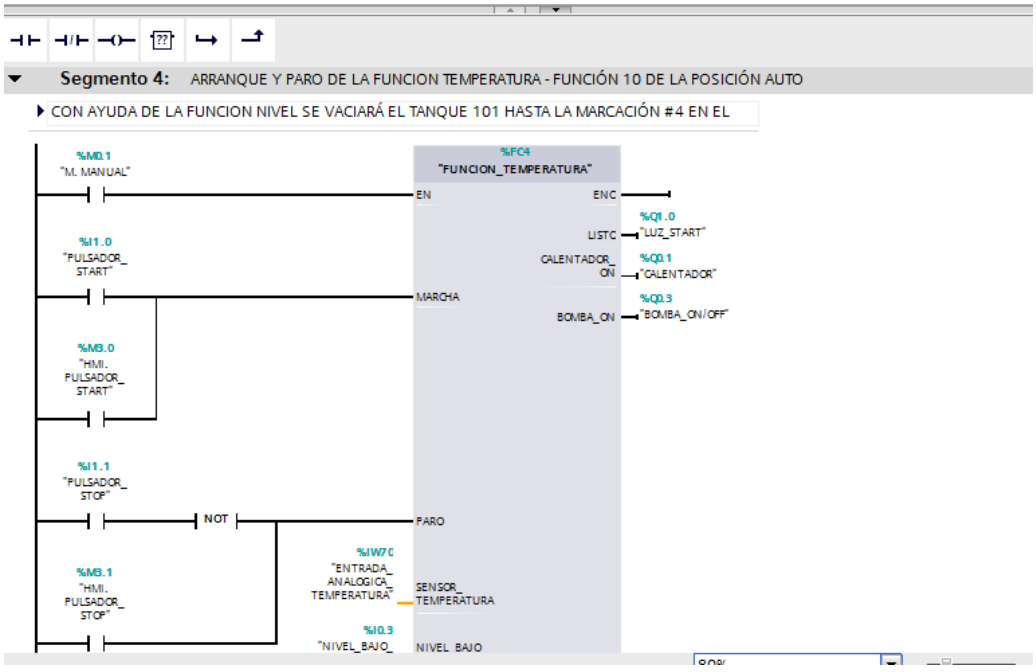

Figura 52. Programa principal (4) *(Dueñas & Chalacán, 2017)*

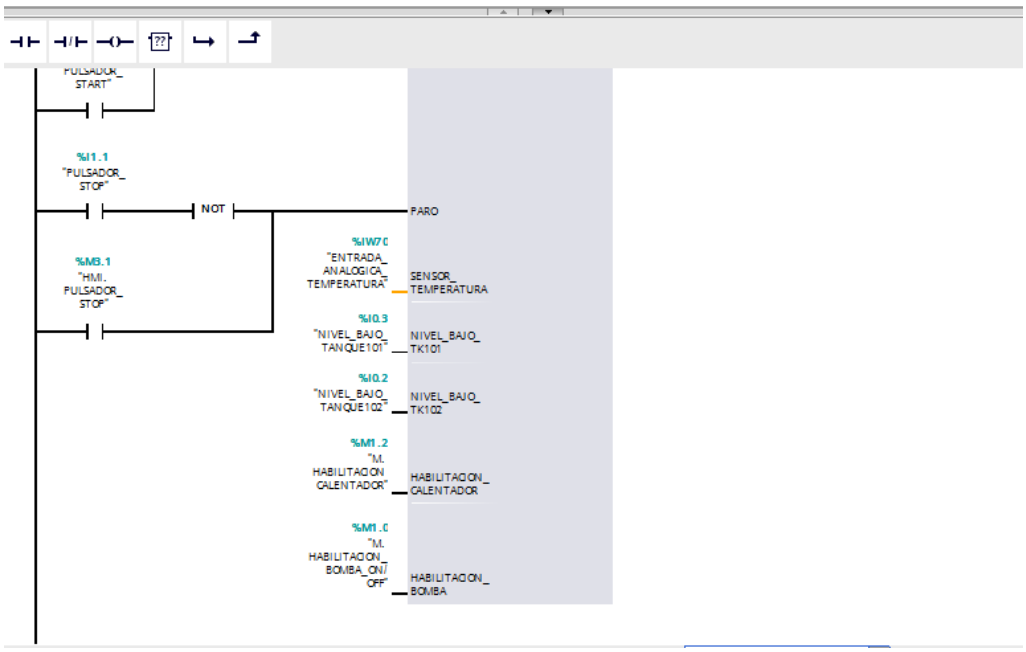

Figura 53. Programa principal (5) *(Dueñas & Chalacán, 2017)*

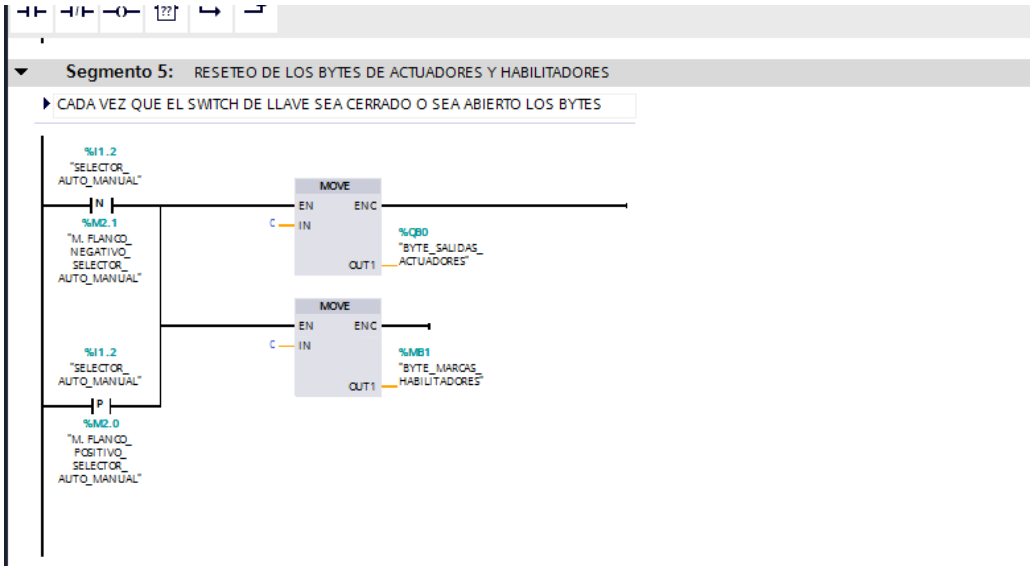

Figura 54. Programa principal (6) *(Dueñas & Chalacán, 2017)*

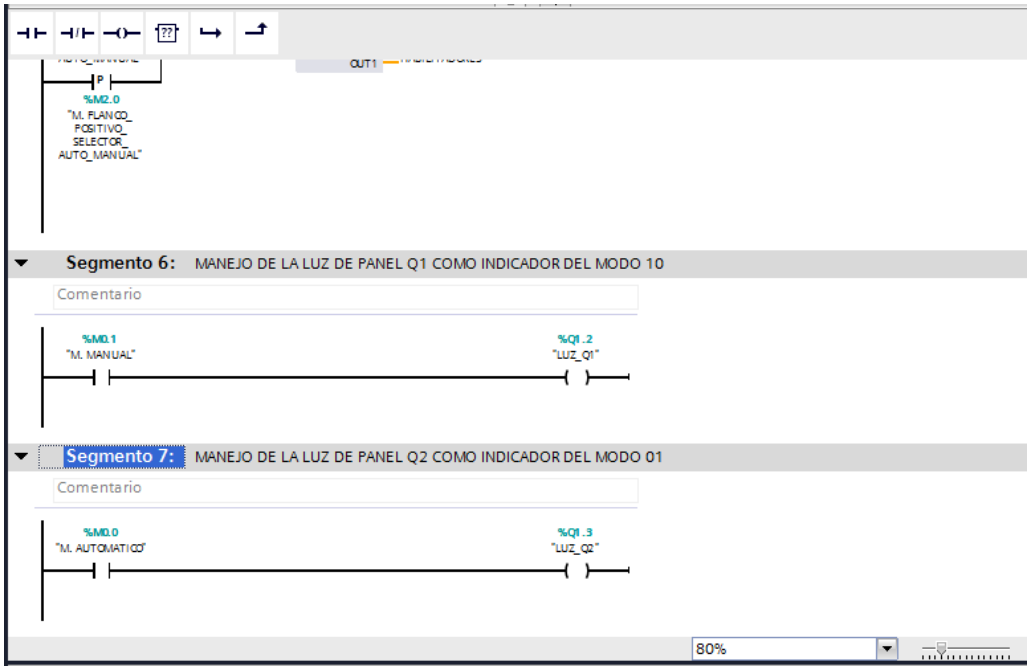

Figura 55. Programa principal (7) *(Dueñas & Chalacán, 2017)*

A10. Variables en NI OPC Servers

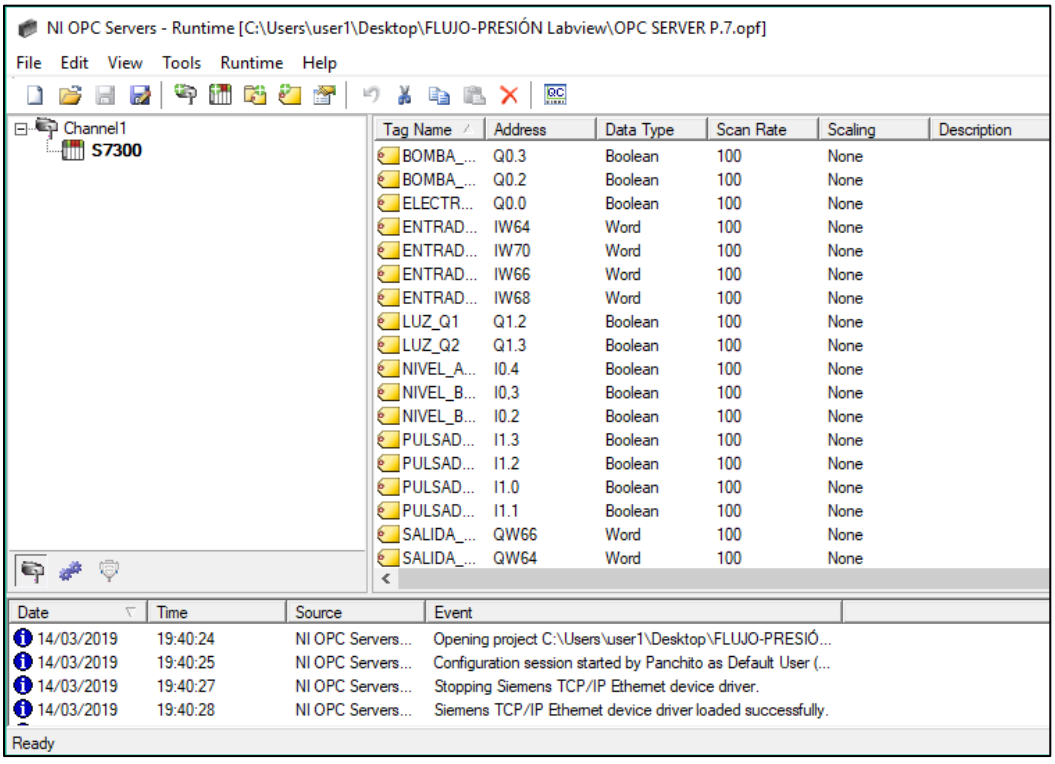

Figura 56. Variables en NI OPC Server *(Dueñas & Chalacán, 2017)*