2022

IMPLEMENTACIÓN DE DASHBOARD Y TELEGRAM BOT EN PLANTA DIDÁCTICA UBICADA EN LA UTFSM SEDE CONCEPCIÓN.

LAGOS LEIVA, SEBASTIAN ANDRES

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE CONCEPCION- REY BALDUINO DE BELGICA

IMPLEMENTACIÓN DE DASHBOARD Y TELEGRAM BOT EN PLANTA DIDÁCTICA UBICADA EN LA UTFSM SEDE CONCEPCIÓN.

Trabajo de Titulación para optar al Título Profesional de Ingeniero de Ejecución en Control e Instrumentación Industrial.

Alumno:
Sebastián Andrés Lagos Leiva.

Profesor Guía:
Nelson Artemio Vásquez Concha

2022

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

SIGLAS

ARPANET : Advanced Research Projects Agency Network.

SCADA : Supervisory control and data acquisition.

HMI : Human-Machine Interface.

ARPANET : Advanced Research Projects Agency Network.

PLC : Programmable Logic Controller.

AISI : American Iron and Steel Institute.

ELCB : Earth Leakage Circuit Breaker.

PID : Proportional-integral-derivative.

IEC : International Electrotechnical Commission.

API : Application programmming interfaces.

IBM : International Business Machines Corporation.

CPU : Central processing unit.

JSON : JavaScript Object Notation.

MQTT : Message Queuing Telemtry Transport.

IoT : Internet of things.

TCP/IP : Transmission Control Protocol/Internet Protocol.

TLC : Transport Layer Security.

SSL : Secure Socket Layers.

VM : Virtual Machine.

VPS : Servidor Virtual Privado.

M2M : Machine to machine.

KPI : Key performance indicator.

SIMBOLOGÍA

mm	: Milímetros.
mA	: Miliampere.
Kg	: Kilogramo.
Vac	: Voltaje alterno.
kVA	: Kilovolt-ampere.
Hz	: Hertz.
h	: Hora.

INTRODUCCIÓN

Actualmente, es común que las personas sepan interactuar con el internet; cito como consecuencia del gran avance tecnológico que se ha generado desde la primera red de computadoras "ARPANET". Lo que ha abierto el camino para que se desarrollen diferentes proyectos que relacionan máquinas e internet. Uno de los primeros programas, se llevó a cabo en 1982 en la Universidad Carnegie Mellon (CMU), donde programadores lograron conectar una máquina de bebidas a la red ARPANET; con el fin de visualizar a través de una computadora la cantidad de refrescos que se encontraban dentro y también la temperatura de estos.

Años después, John Romkey, fundador de FTP Software, logró controlar el encendido y apagado de una tostadora eléctrica a través de internet. Poco después los ingenieros de la universidad de Cambridge llevaron a cabo la monitorización de una máquina de café a través de una cámara conectada a internet, desarrollando así, la primera Webcam.

Son estos algunos de los proyectos que nos daban indicios de cómo sería el internet en un futuro, y el impacto que tendría en nuestras vidas.

Hoy en día, hay una extensa lista de dispositivos con la capacidad de conectarse a internet, incluso de manera inalámbrica. Esto ha permitido la creación de diferentes sistemas donde los estos pueden comunicarse entre sí, para realizar tareas, tomar decisiones, monitorizar, entre otras acciones; todo con la mínima o nula intervención de los usuarios.

Si nos enfocamos en el área industrial, el uso de internet no se ha utilizado principalmente en los procesos, sino para usos más bien administrativos. En este campo, para la obtención de datos tenemos diferentes protocolos de comunicación que realizan la transmisión y recepción de datos, para luego ser visualizados en softwares SCADA o dispositivos HMI.

Aun así, algunas empresas ya están realizando la monitorización de sus procesos a través de internet, ya que presenta la ventaja de poder visualizar los datos de sus procesos desde computadoras, tabletas y smartphones en diferentes partes del mundo.

Un ejemplo de esto es el proyecto realizado por ex alumnos de nuestra carrera quienes implementaron un sistema de extracción, programación y visualización de datos a través de internet para los aerogeneradores ubicados en el parque eólico Negrete Cuel.

Es por esto que, debido a la importancia futura de esta área y para que los alumnos de nuestra carrera puedan practicar e implementar estos sistemas en su desarrollo profesional, es que en el presente proyecto se realizará la extracción y visualización de datos a través de internet para la planta didáctica de nivel ubicada en el laboratorio de procesos de nuestra casa de estudios. Aquí los estudiantes podrán agregar o modificar sus funcionalidades y replicarlas en las diferentes estaciones de trabajo del laboratorio o implementarlas en su quehacer profesional.

OBJETIVO GENERAL

Implementar un sistema de visualización de datos a través de internet utilizando Node-RED para la planta didáctica de nivel ubicada en el laboratorio de procesos de nuestra universidad.

OBJETIVOS ESPECIFICOS

- Instalar Node-RED y software de programación correspondiente a PLC S7-200 en una computadora personal.
- Establecer comunicación entre Node-RED y PLC para la extracción de datos.
- Utilizar Node-RED para crear una programación que permita el envío y visualización de datos a través del protocolo MQTT.
- Utilizar Node-RED para crear una programación que permita la solicitud de datos a través de un BOT de Telegram.

ALCANCE

Este proyecto estará enfocado a la implementación de un sistema de visualización de datos a través de internet. Por ello, no se abordará en profundidad el funcionamiento del proceso escogido ni tampoco como está programado en el PLC.

CAPÍTULO 1. MARCO TEÓRICO

1.1 **PROCESO.**

Para el desarrollo práctico de nuestra carrera, la universidad dispone de un laboratorio en el que se encuentran diferentes maquetas que simulan procesos industriales.

Como se ha mencionado, el objetivo principal de este proyecto es desarrollar un sistema de monitoreo remoto a través de internet para un proceso industrial. Para esto, se ha escogido una de las maquetas que dispone el laboratorio de procesos. Esta maqueta corresponde a el Banco de control de nivel.

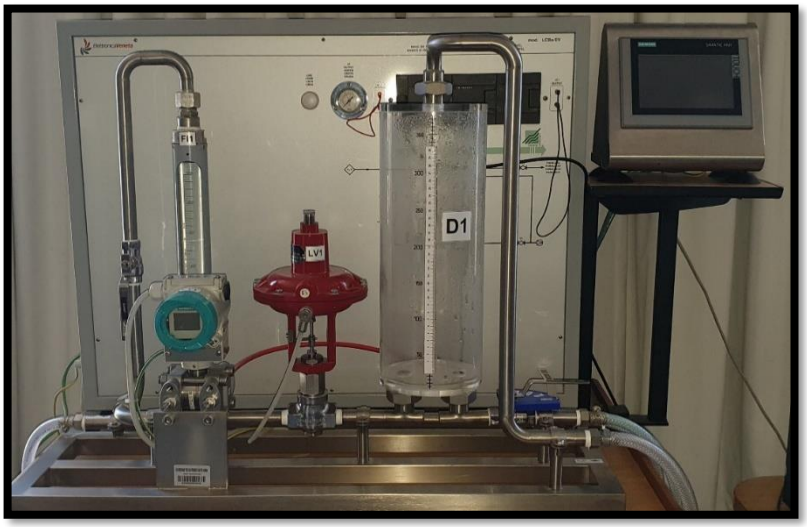

Figura 1-1. Proceso.

Esta maqueta fue desarrollada por la compañía italiana Elettronica Veneta, la cual se dedica a diseñar, fabricar e instalar equipos didácticos para la educación técnica; Esta unidad en particular permite estudiar el control de nivel utilizando una línea que incluye un depósito para el agua, una válvula de control, un transmisor de presión diferencial y un controlador lógico programable S7-200. El nivel se mide mediante el

transmisor de presión diferencial y se controla mediante una válvula neumática instalada en la entrada de agua del depósito.

Además, este proceso cuenta con una HMI que permite visualizar y controlar los datos del proceso.

Las especificaciones de este proceso son la siguientes:

- Estructura de acero inoxidable
- Válvula de control neumática de acero inoxidable AISI 316, Cv = 1,25
- Transmisor electrónico de nivel de presión diferencial, de acero inoxidable AISI 316 con rango de 0 a 500 mm H2O y 4 a 20 mA señal de salida
- Caudalímetro de área variable de acero inoxidable y vidrio con rango de 100 a 1000 l/h
- Convertidor electroneumático, 4 a 20 mA / 0,2 a 1 bar
- Depósito graduado de plexiglás con capacidad de 5 litros
- Manómetro para medir la señal de salida del convertidor I/P convertidor
- Cuadro de distribución de acero al carbono pintado con sinóptico, ELCB y terminales de medición para las señales de entrada y salida del controlador
- PLC industrial colocado en el cuadro eléctrico, incluyendo el bloque de control PID y módulo de comunicaciones Ethernet. Se incluye el software de programación estándar IEC 1131 61131 software
- Panel de operador, pantalla táctil TFT de 7", 16 millones de colores, 800 x 480 píxeles, con puertos RJ45 PROFINET; el software de supervisión industrial HMI.
- Dimensiones: 850 x 600 x 750 mm
- Peso neto: 50 kg
- Fuente de alimentación: 230 Vac 50 Hz monofásica - 0,5 kVA

La programación de este proceso está realizada a través del software V4.0 STEP 7 MicroWIN SP9 y se representa en las siguientes figuras.

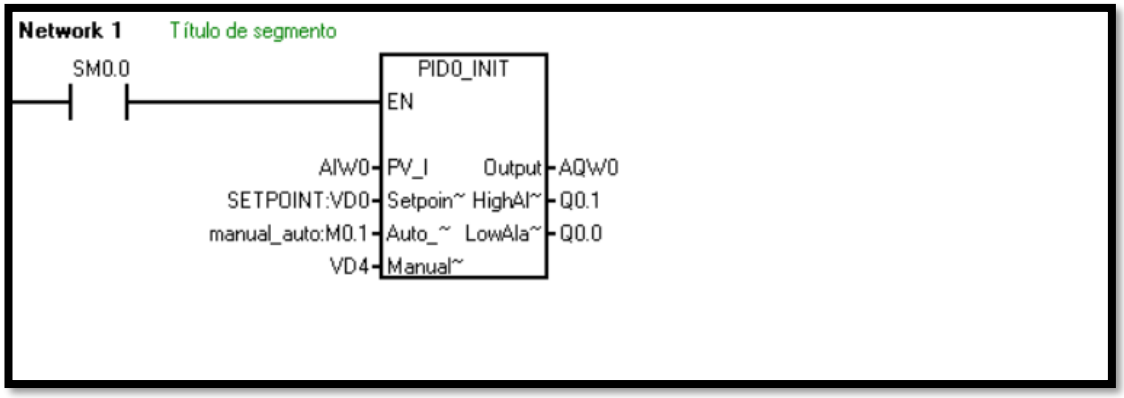

Figura 1-2. Programación PLC parte uno.

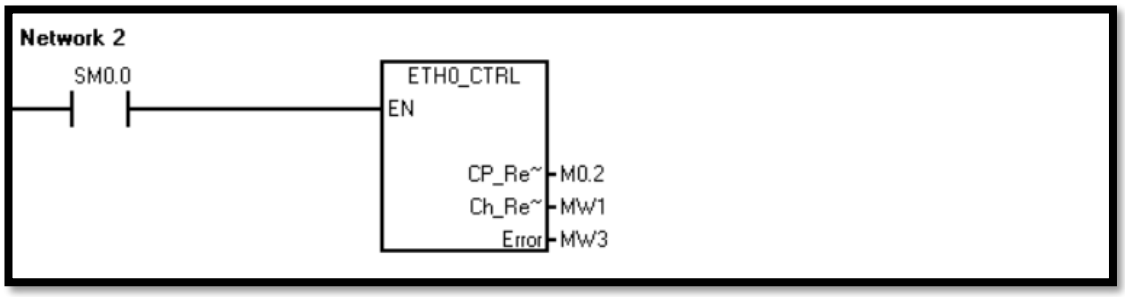

Figura 1-3. Programación PLC parte dos.

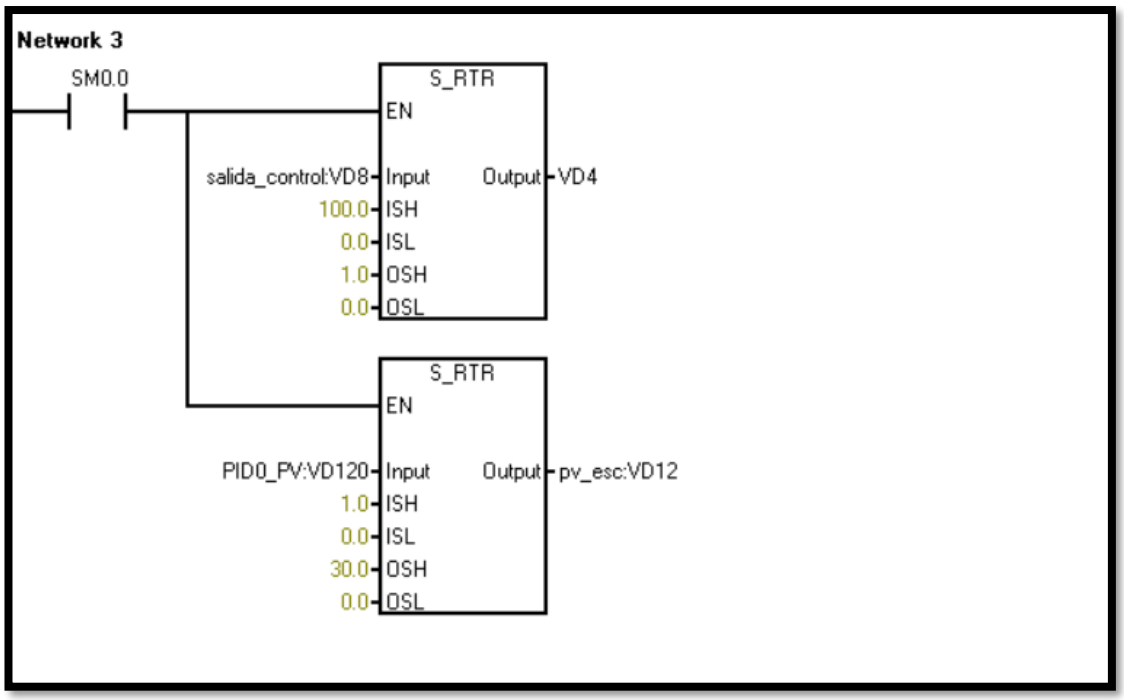

Figura 1-4. Programación PLC parte tres.

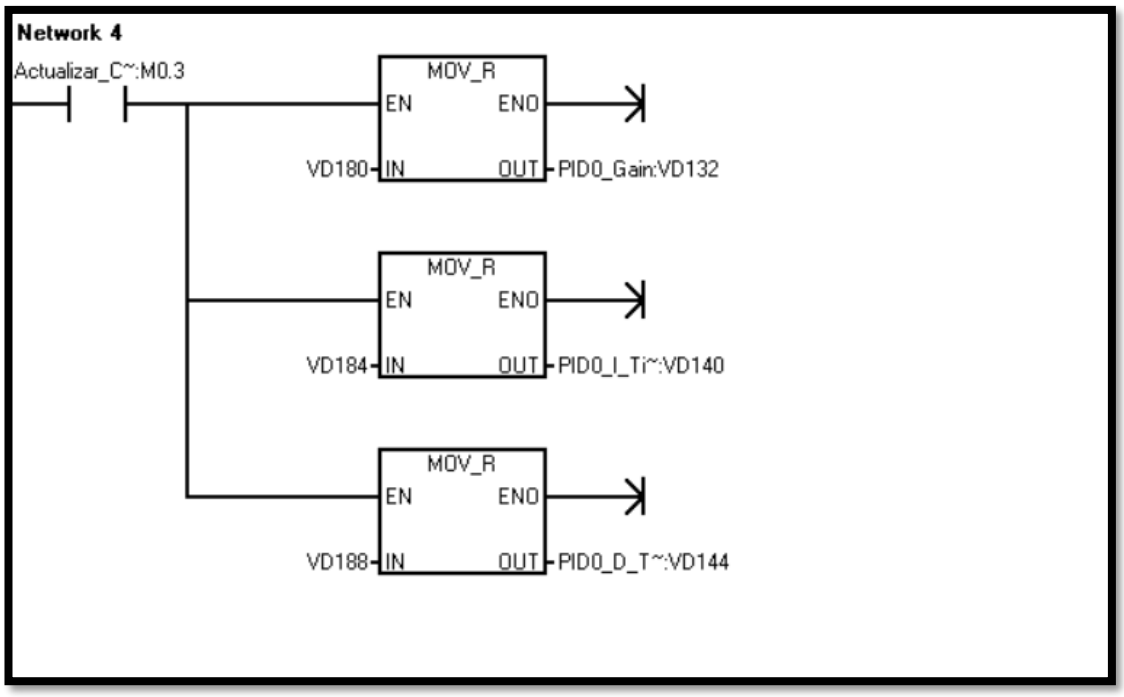

Figura 1-5. Programación PLC parte cuatro.

1.2 **NODE-RED**

1.2.1 ¿Qué es Node-RED?

Node-RED es una herramienta de programación de código abierto (JavaScript) basada en flujo que nos permite conectar hardware con APIs u otros servicios en línea.
Fue creada en 2013 por Nick O' Leary y Dave Conway, quienes formaban parte del grupo de servicios de tecnologías emergentes de IBM, y que tenían como objetivo ofrecer una solución que facilitara la conexión de hardware con otros servicios.
Al ser de código abierto, esto dio paso a la formación de JSFundation, la cual nace con la idea de unificar el lenguaje JavaScript y así establecer un estándar de programación con relación a este; Posteriormente, esta asociación se fusionó con Node.js iniciando OpenJs.

1.2.2 Programación basada en flujo.

Inventada por J. Paul Morrison en la década de 1970, la programación basada en flujo es una forma de describir el comportamiento de una aplicación como una red de "cajas negras" o "nodos", como se les llama en Node-RED. Cada nodo tiene un propósito definido; se le dan algunos datos, el nodo trabaja con estos y luego los transmite.

Esto ha permitido que más usuarios puedan usar esta herramienta, ya que es simple de entender y no se necesita saber cuál es la programación interna de cada nodo, sino del propósito que cumplen.

1.2.3 Tiempo de ejecución basado en Node.js

Node.js es un entorno de ejecución de un solo hilo y multiplataforma basado en el motor V8 de JavaScript de Google Chrome. Este software de código abierto permite construir aplicaciones de red escalables y en tiempo real. Este, funciona como un único proceso, lo que significa que no crea un nuevo "hilo" para cada petición (un hilo es un conjunto de instrucciones que debe realizar el servidor).

Node.js emplea operaciones de E/S no bloqueantes, es decir que cuando un cliente envía una solicitud al servidor web, el bucle de eventos de un solo hilo la recoge y la envía a un hilo trabajador para su procesamiento. Luego, en lugar de bloquear el hilo y desperdiciar recursos de la CPU mientras espera una respuesta, Node.js continúa trabajando en la tarea siguiente.

De esta manera, Node.js puede manejar una cantidad masiva de peticiones simultaneas, teniendo en consideración que las tareas no deben requerir un uso intensivo de la CPU, ya que podría impedir que el hilo principal maneje otras peticiones, provocando el bloqueo del sistema.

1.2.4 Edición de flujo basada en navegador.

La edición de flujo en Node RED es a través de un navegador, lo que permite realizar toda la programación dentro de este, ahorrando instalaciones de otros softwares para programar.

Luego de acceder al editor, se dispone de una amplia gama de "nodos" que se pueden utilizar en diversos proyectos. Además, la paleta de nodos se puede ampliar fácilmente instalando nuevos nodos creados por la comunidad.

Los flujos creados también podrán ser importados o exportados a través de archivos JSON.

En la siguiente figura se pueden ver los tres sectores más importantes del editor.

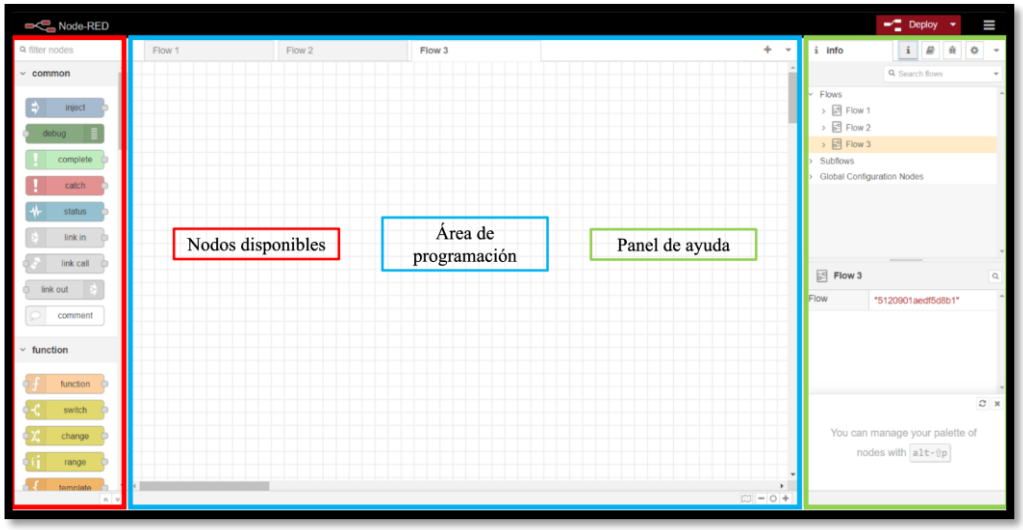

Figura 1-6. Ventana de programación Node-RED.

En el primer bloque, se ubican todos los nodos instalados en el editor, de los cuales se disponen desde la instalación de Node-RED en el PC. Además, gracias a que esta herramienta de programación trabaja en código abierto, múltiples usuarios han creado sus propios nodos, los cuales pueden ser descargados he instalados según se necesiten.
En el segundo bloque está ubicado el lugar de programación, donde se pueden colocar los nodos y realizar los flujos que se requieren en los proyectos.
En el tercer bloque existe una pestaña de ayuda, donde se tendrán como principales funciones:

1. Panel de debug: Herramienta que muestra errores y mensajes de nuestros programas.
2. Panel de información: Muestra los flujos y subflujos que tenemos desarrollados en nuestro editor.
3. Panel de ayuda: Entrega una pequeña reseña de los nodos disponibles.
4. Panel de configuración: Permite ver todos los nodos y los flujos que estemos usando, y también los que no.

Además, en este bloque se puede acceder a la configuración del dashboard, donde se ajustan los espacios que ocupará cada elemento del dashboard, los colores y títulos.

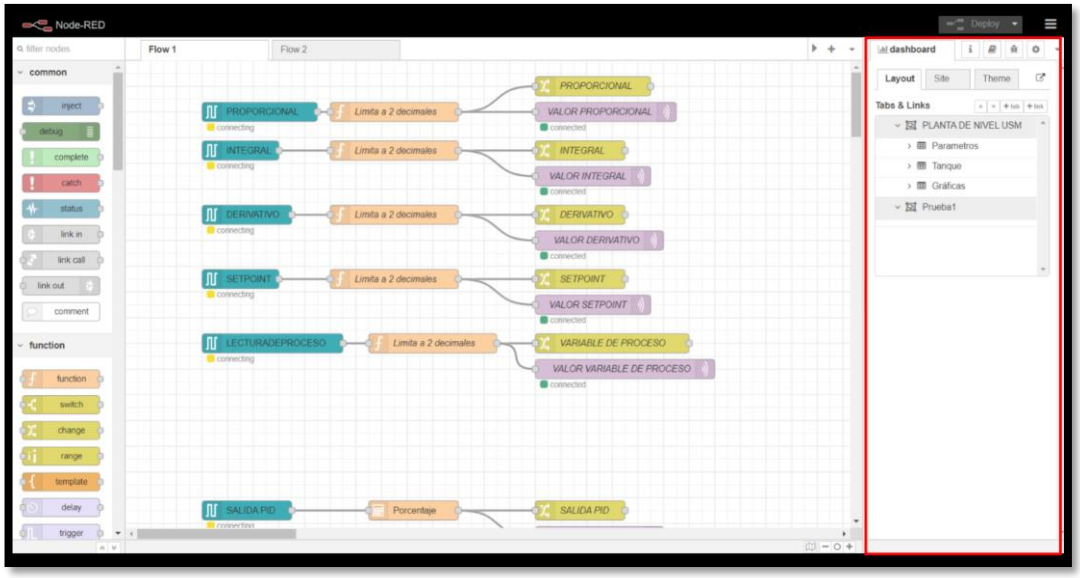

Figura 1-7. Pestaña de ajuste para dashboard.

También se dispone de un botón “deploy” que permite compilar el programa y de una pestaña de opciones básicas de edición, de las cuales destaca la función “manage palette”, que permite instalar y eliminar los nodos.

1.3 **MQTT.**

1.3.1 Protocolos de comunicación.

Antes de hablar sobre MQTT, se debe tener claro qué es un protocolo de comunicación, y cómo llegan a ser una pieza clave en los sistemas.

Un protocolo es un conjunto de instrucciones, normativas o reglas que permiten guiar acciones o que constituyen las bases al momento de realizar un procedimiento. Por otra parte, la comunicación es el proceso en el cual se intercambian opiniones, datos o información sobre un tema determinado y es vital para que exista un buen entendimiento entre las personas. Ahora, si unimos estos dos conceptos y lo enfocamos a la comunicación de máquinas y no de personas, se puede concluir que los protocolos de comunicación vendrían a ser el conjunto de instrucciones, normativas o reglas para la transmisión y recepción de datos o información. Dentro de esto, se define la sintaxis, la semántica y la sincronización de la comunicación.

Por lo tanto, cuando se necesita comunicar instrumentos, controladores, computadoras, u otros dispositivos, se debe tener en cuenta con cual protocolo se va a trabajar.

Algunos de los protocolos de comunicación más usados dentro de la industria son: HART, Profibus, MODBUS, Profinet, entre otros.

1.3.2 ¿Qué es MQTT?

MQTT (Message Queuing Telemtry Transport, o MQ Telemtry Transport) es un protocolo de transporte de mensajería que trabaja con la metodología de publicación/suscripción cliente-servidor. Es sencillo, ligero y diseñado para una fácil implementación, que busca minimizar el consumo de ancho de banda de la red y los requerimientos de los dispositivos, al mismo tiempo que, intenta garantizar la confiabilidad y cierto grado de garantía en la entrega de datos. Este protocolo se ejecuta sobre TCP/IP, o sobre otros protocolos de red que proporcionan conexiones ordenadas, sin pérdidas y bidireccionales.

Estas características lo hacen ideal para trabajar en comunicación M2M o en proyectos de IoT, donde los principales problemas se generan por limitación de potencia, consumo y ancho de banda.

1.3.3 ¿Cómo surge MQTT?

Este protocolo fue creado originalmente por el Dr. Andy Stanford-Clark y Arlen Nipper en 1999. Su propósito original fue permitir que los dispositivos de monitoreo utilizados en la industria del petróleo y el gas enviaran sus datos a servidores remotos.

En muchos casos, estos dispositivos de monitoreo se empleaban en ubicaciones muy lejanas, donde establecer cualquier tipo de línea fija, conexión por cable o enlace de transmisión de radio sería casi imposibles. En ese momento, la única opción para encarar estas situaciones eran las comunicaciones por satélite, que eran tremendamente costosas.

1.3.4 Arquitectura MQTT.

En la arquitectura MQTT, existen dos tipos de sistemas: los clientes y el bróker.

- Cliente: Estas entidades se encargarán de publicar o solicitar los mensajes al bróker. Estos pueden ser casi cualquier dispositivo "inteligente", incluidos teléfonos inteligentes, aplicaciones web, nodos de sensores, actuadores o incluso relojes inteligentes.
- Bróker: Este es el servidor que se encarga de realizar la transacción de mensajes, enviando y recibiendo datos de los clientes. Además, permite que los dispositivos (clientes) se comuniquen de forma desacoplada, por lo que esta arquitectura se puede escalar muy fácilmente sin siquiera afectar a los dispositivos cliente existentes. Gracias a que el bróker realiza todo el trabajo pesado, los clientes solo tendrán que hacer un procesamiento mínimo, ocupando un bajo ancho de banda.

a. Tipos de bróker:

   i. Bróker Gestionado: Los brókeres administrados no requieren que se configure nada en su servidor para habilitar la comunicación MQTT. Los servicios de agente administrado le permiten usar sus agentes alojados para su sistema. AWS IoT Core es un buen ejemplo de un agente MQTT administrado.

   ii. Bróker Auto hospedado: Como su nombre lo indica, los corredores MQTT auto hospedados requieren que instale el corredor en su propio VPS o servidor con una IP estática. El proceso de instalación no es difícil, pero administrar, asegurar y escalar los corredores requiere un conocimiento profundo del sistema. Hay varias implementaciones de código abierto de los corredores MQTT, incluyendo Mosquitto y Hivemq.

1.3.5 Metodología publicación/suscripción.

Como se ha mencionado, MQTT trabaja con la metodología de publicación/subscripción. Esto quiere decir que debe existir un cliente que publique los mensajes y otro que los solicite (suscriptor). Ambos clientes estarán conectados al bróker, el cual decidirá la recepción y envío de mensajes.

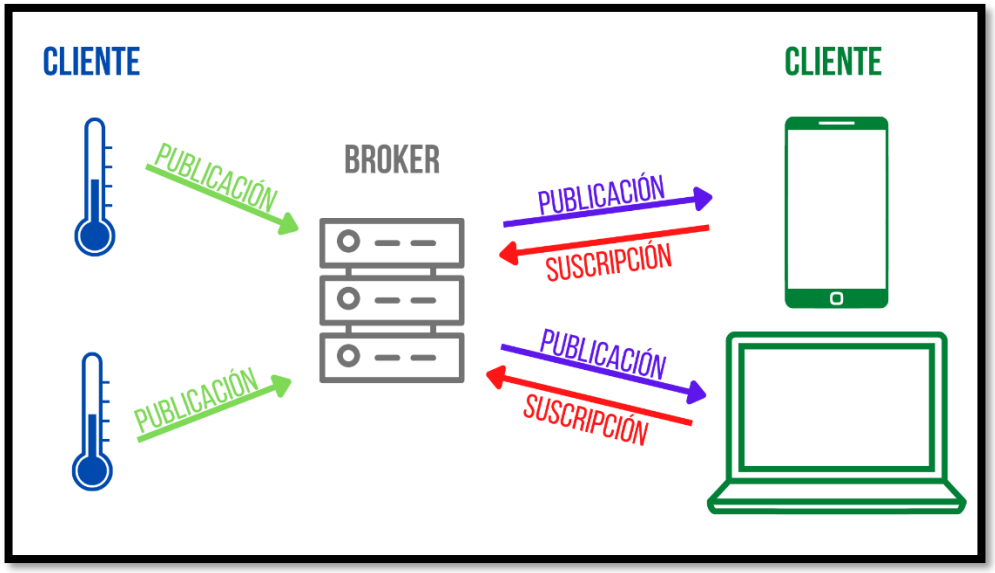

Figura 1-8. Arquitectura MQTT.

Fuente: Elaboración propia creada en https://www.canva.com

1.3.6 Tópicos.

Para filtrar los mensajes que son enviados a cada cliente, estos son publicados como temas o “tópicos” que podemos organizar jerárquicamente, por lo que, si un cliente publica un mensaje en un tema determinado, y se requiere leer ese mensaje, se deben suscribir a ese tema para que el bróker les haga llegar el mensaje.

Los tópicos están formados por uno o más "niveles" separados entre sí por una barra inclinada '/'. Cada nivel está formado por uno o más caracteres.

Como ejemplo, si se implementa un sistema de monitorización de temperatura y humedad dentro de un edificio a través de MQTT, se podría llegar a tener los siguientes tópicos:

Tópico	Función
“Edificio1/Piso1/Temperatura”	Este tópico estará realizando él envío o la recepción de la variable temperatura del piso 1 correspondiente al edificio 1.
“Edificio1/Piso3/Humedad”	Este tópico indica que estará realizando él envío o la recepción de la variable humedad del piso 3 correspondiente al edificio 1.
“Edificio1/Piso3/#”	En este caso, se usa un comodín que permite recibir todos los tópicos que empiecen con Edificio1/Piso3/. Hay que tener en cuenta que sólo se puede utilizar en el último nivel del tópico.
“Edificio1/+/Temperatura”	En este caso, se usa un comodín que permite recibir los datos de temperatura de “todos los pisos” del edificio 1. Este comodín permite reemplazar un único nivel en cualquier lugar del tópico.

Tabla 1-1. Ejemplo de posibles tópicos.

1.3.7 Arquitectura de mensajes.

Otra forma en que MQTT minimiza sus transmisiones, es con un tamaño de mensaje pequeño y bien definido. Cada mensaje consta de tres partes.

<table>
<tr><th colspan="2">1</th><th>2</th><th>3</th></tr>
<tr><th colspan="2">Fijo</th><th>Opcional</th><th>Opcional</th></tr>
<tr><td colspan="2">Cabecera</td><td rowspan="3">Cabecera</td><td rowspan="3">Contenido/Payload</td></tr>
<tr><td>Control</td><td>Paquete</td></tr>
<tr><td>Cabecera</td><td>Longitud</td></tr>
<tr><td>1 byte</td><td>1-4 bytes</td><td>0-Y bytes</td><td>0-256 Mbps</td></tr>
</table>

Tabla 1-2. Arquitectura de mensajes MQTT.

1. Cabecera fija. Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.
2. Cabecera variable. Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.
3. Contenido(payload). Es el contenido real del mensaje. Puede tener un máximo de 256 Mb.

1.3.8 Calidad de servicio (QoS).

MQTT dispone de tres niveles diferentes de calidad de servicio (QoS) que permiten a los diseñadores de redes elegir entre minimizar la transmisión de datos y maximizar la fiabilidad.

Estos niveles son:

- QoS 0: Ofrece la cantidad mínima de transmisión de datos. Con este nivel, cada mensaje se entrega a un suscriptor una vez, sin confirmación, por lo que no hay forma de saber si los suscriptores recibieron el mensaje. Este método a veces se denomina “lanzar y olvidar” o “una entrega como máximo”. Debido a que este nivel asume que la entrega está completa, los mensajes no se almacenan para entregarlos a los clientes desconectados que luego se vuelven a conectar.

- QoS 1: El bróker intenta entregar el mensaje y luego espera una respuesta de confirmación del suscriptor. Si no se recibe una confirmación dentro de un período de tiempo especificado, el mensaje se envía de nuevo. Usando este método, el suscriptor puede recibir el mensaje más de una vez si el bróker no recibe el acuse de recibo del suscriptor a tiempo. Esto se suele denominar "entregado al menos una vez".
- QoS 2: el cliente y el bróker utilizan un protocolo de enlace de cuatro pasos para garantizar no sólo que el mensaje se reciba, sino que lo haga una única vez. También se conoce como "entrega exactamente una vez".

1.3.9 Mensajes retenidos.

En MQTT, los mensajes recibidos no se almacenan en el bróker a menos que estén marcados con la bandera de retención. A esto se le llama mensaje retenido. Los usuarios que deseen almacenar los mensajes recibidos deberán almacenarlos en otro lugar fuera del protocolo MQTT. Sin embargo, existen excepciones.

Al ser un protocolo basado en eventos, es posible que un suscriptor reciba muy pocos mensajes sobre un tema determinado, incluso durante un largo período de tiempo. Por lo tanto, los nuevos suscriptores al tema podrían llegar a pensar que exista algún problema de comunicación al no recibir ningún mensaje. Entonces, al usar la opción de mensaje retenido, cada vez que un nuevo cliente se suscriba, se enviará el ultimo dato registrado por el tema.

1.3.10 Ultima voluntad y testamento.

Cuando las comunicaciones no son fiables, es posible que un editor se desconecte de la red sin previo aviso. En tales casos, un editor puede registrar un mensaje para enviarlo a los suscriptores por si se desconecta inesperadamente, en lo que se denomina última voluntad y testamento. Este mensaje se almacena en la caché del bróker y,

normalmente, incluye información que permite identificar al editor desconectado para que se puedan tomar las medidas adecuadas.

1.3.11 Tipos de mensajes MQTT.

Para mantener el protocolo al mínimo, sólo se pueden efectuar cuatro acciones en cualquier comunicación: publicar, suscribirse, cancelar suscripción o hacer ping.

- Publicar: envía un bloque de datos que contiene el mensaje que se va a enviar. Estos datos son específicos de cada implementación, pero pueden ser algo tan simple como una indicación de encendido/apagado o un valor de un determinado sensor, como temperatura, presión, etc. En el caso de que el tema que se está publicando no exista, éste se crea en el bróker.

- Suscribirse: convierte a un cliente en suscriptor de un tema. Se puede suscribir a uno o varios temas. Para suscribirse, un cliente envía un paquete SUBSCRIBE y recibe un paquete SUBACK a cambio. Si hay un mensaje retenido para el tema, el nuevo suscriptor también lo recibe.

- PING: un cliente puede hacer ping al bróker. El suscriptor envía un paquete PINGREQ y, como respuesta, se recibe un paquete PINGRESP. Se pueden utilizar pings para garantizar que la conexión siga funcionando y que la sesión TCP no haya sido cerrada inesperadamente por otro equipo de red, como un Router o una puerta de enlace.

- DESCONECTAR: un suscriptor o editor puede enviar un mensaje de DISCONNECT al bróker. Este mensaje informa al bróker de que ya no necesitará enviar o poner en cola mensajes para un suscriptor y que ya no recibirá datos de un editor. Este tipo de cierre permite al cliente volver a conectarse utilizando la misma identidad de cliente que en ocasiones anteriores. Cuando un cliente se desconecta sin enviar un mensaje de desconexión, se envía su última voluntad y testamento a los suscriptores.

1.3.12 Seguridad en MQTT.

Si bien la seguridad no fue una de las principales preocupaciones durante el diseño e implementación de MQTT, hay algunas opciones de seguridad disponibles a costa de una carga superior en la transmisión.

- Seguridad de la red: si la red en sí puede protegerse, la transmisión de datos inseguros en MQTT es irrelevante. En tal caso, los problemas de seguridad tendrían que producirse desde el interior de la propia red, quizás a través de un actor malicioso u otra forma de penetración en la red.
- Nombre de usuario y contraseña: MQTT permite nombres de usuario y contraseñas para que un cliente establezca una conexión con un bróker.
- SSL/TLS: al ejecutarse sobre TCP/IP, la solución obvia para proteger las transmisiones entre clientes y brókeres es la implementación de SSL/TLS. Lamentablemente, esto añade una sobrecarga sustancial a las comunicaciones.

1.3.13 ¿Dónde se usa MQTT?

Debido a que las implementaciones de MQTT son más ligeras y eficientes qué otras arquitecturas de comunicación (por ejemplo, una API RESTful basada en HTTP), MQTT es a menudo una buena opción para IoT o IIoT. Algunos de los casos de uso más comunes son:

1. Recopilar datos de los nodos del sensor y publicarlos en el servidor.
2. Publicación de datos de misión crítica directamente desde un nodo de sensor a un dispositivo de usuario.
3. Configuración remota de dispositivos IoT e IIoT.
4. Envío de datos de configuración desde una única plataforma web o aplicación de smartphone a todos los dispositivos a la vez (gracias a los temas MQTT que permiten comodines).

5. Enviar instantáneamente actualizaciones OTA (over-the-air) a todos los dispositivos de la red.

Además de los casos de uso de IoT e IIoT, MQTT también ha encontrado su camino en los productos de software tradicionales y los sistemas distribuidos a gran escala. En particular, Facebook utiliza MQTT para su aplicación Messenger. Del mismo modo, muchas aplicaciones de mensajería de código abierto, como Chat-App, utilizan MQTT. A medida que la comunidad MQTT continúa creciendo, se están lanzando más proyectos de código abierto y de código cerrado que utilizan este.

1.4 **TELEGRAM.**

1.4.1 ¿Qué es Telegram?

Telegram es una aplicación de mensajería en la nube para móviles y computadoras con foco en la seguridad y la velocidad. Este implementa un protocolo propio llamado MTProto, que transmite los mensajes de forma segura entre nuestro móvil y el servidor. Incluso permite crear chats seguros entre dos clientes, de tal forma que ni siquiera los servidores de Telegram pueden ver qué estás enviando.

También, esta aplicación dispone de diferentes APIs se pueden utilizar en los proyectos. En este caso, se utilizará la API Bot, que permite conectar bots a los sistemas. Estos son cuentas especiales que no requieren un número de teléfono adicional para configurarse y que sirven como interfaz para el código que se ejecuta en algún lugar del servidor.

La ventaja de incluir Telegram en este proyecto, es que se puede integrar la API de Bot en Node-RED, lo que permitirá tener interacción entre los dispositivos móviles y el proceso.

CAPÍTULO 2. PRESENTACIÓN DE PROYECTO.

2.1 **PROYECTO.**

En este proyecto, se utilizará una de las plantas didácticas que se encuentran en el laboratorio de procesos de nuestra universidad.

El proceso seleccionado se encarga de medir y controlar el nivel de un estanque cerrado. El control se realiza a través de un PLC S7-200, el cual interactúa con un transmisor de presión que le permite calcular el nivel del estanque. Dependiendo de este nivel y el Setpoint asignado, el PLC toma la decisión de abrir o cerrar la válvula que deja entrar el agua al estanque.

Originalmente, el PLC está conectado a una HMI que permite interactuar con el proceso. Este se encuentra conectado a través de un cable ethernet con la PC local, donde se encuentran los softwares de programación.

En el desarrollo de este trabajo, se realizará la implementación de un nuevo sistema de visualización de datos para el proceso a través de internet.

Para esto se utilizará una PC personal, que se encargará de comunicarse directo con el PLC y enviar los datos a internet.

Cabe destacar que el PC personal tiene un SO Windows 10 Home Single Language Versión 21H1.

2.2 **SOFTWARE DE PROGRAMACIÓN PLC-S7200.**

Para realizar la programación de este PLC, se hace uso del software V4.0 STEP 7 MicroWIN SP9.

Como ya se ha mencionado, la programación de este proceso ya está realizada, y se encuentra disponible en las PC locales que se encuentran en el laboratorio, por lo tanto, lo primero que se llevará a cabo, es la instalación del software en la PC personal.

Luego de tener instalado el software, se debe realizar la conexión entre el PLC y la PC. Para esto desconectamos el cable ethernet que comunica el PLC con la HMI y conectamos el PLC con la PC.

Como antes el proceso estaba comunicado con la PC local, se deben copiar los datos de red de esta y aplicarlas en la PC personal. La configuración de red es la siguiente:

	PLC	PC
Dirección IP	10.4.5.46	10.4.5.45
Mascara de subred	255.255.255.0	255.255.255.0
Puerta de enlace predeterminada	10.4.5.1	10.4.5.1

Tabla 2-1. IP utilizada en el proceso.

Esta configuración se debe cambiar en las propiedades del protocolo de comunicación TCP/IPv4.

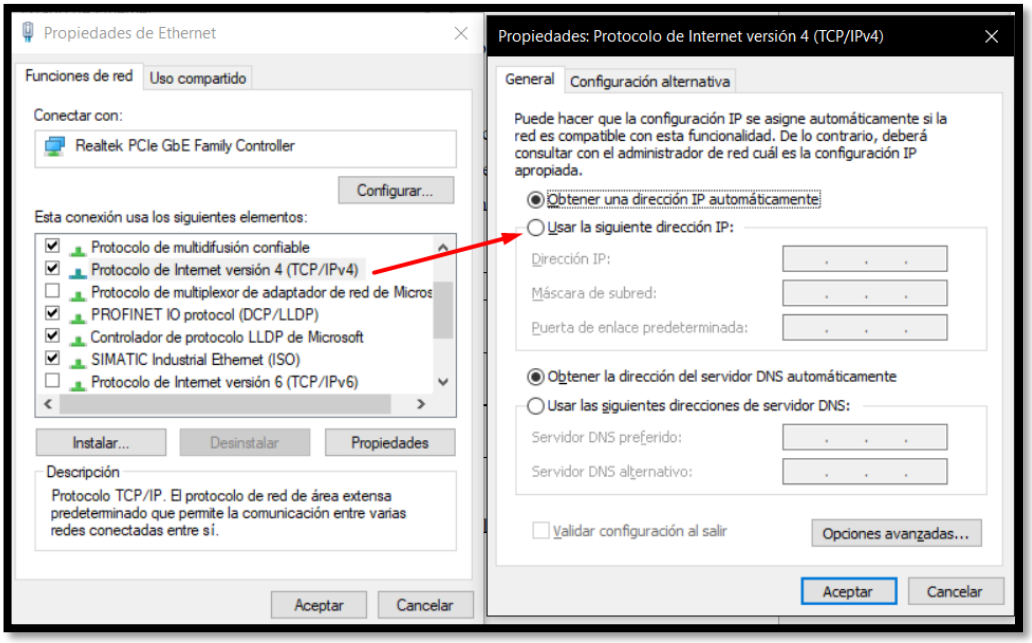

Figura 2-1. Cambio de IP TCP/IPv4.

Después de tener las IP cambiadas, se debe entrar al software de programación y abrir el archivo del programa LCB.

Dentro del programa, se selecciona la opción de Ajustar interfaz PG/PC para luego asignar la tarjeta de red que controla la conexión TCP/IP.

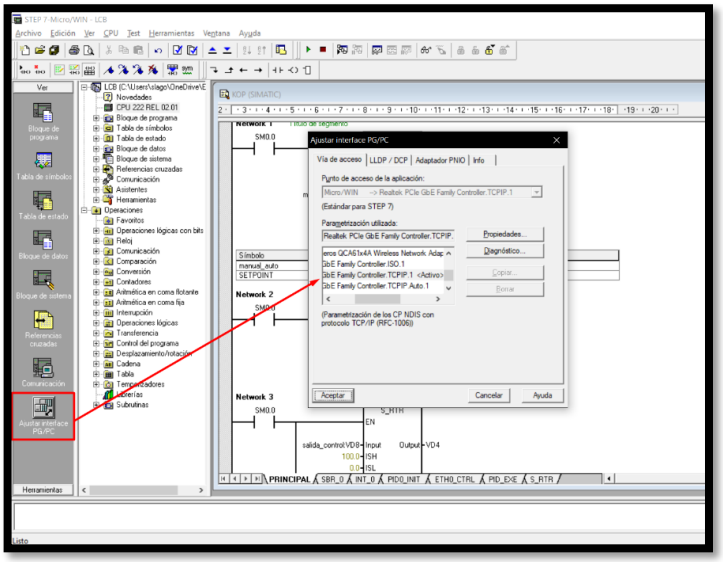

Figura 2-2. Ajuste de interfaz PG/PC dentro de MicroWin.

Posterior a esto, se selecciona la pestaña de Comunicación para identificar el PLC al cual se está conectado.

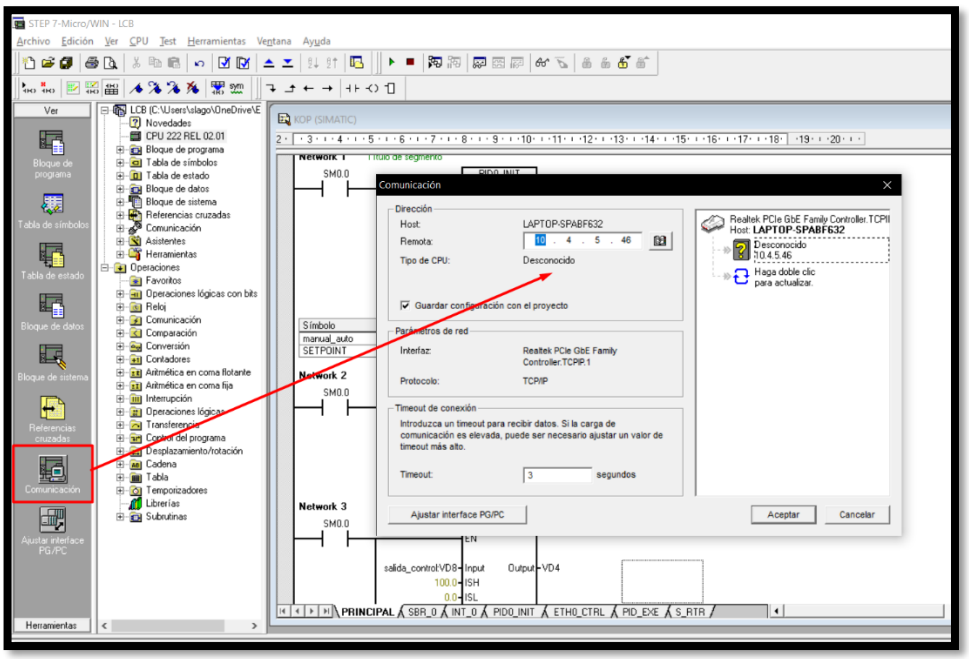

Figura 2-3. Configuración para la comunicación con PLC.

Ahora, si no se presenta ningún error, se podría interactuar con el PLC.

En la barra de herramientas de este software se utilizarán:

Herramienta	Descripción
Cargar en PG	Permite cargar en el PLC el programa que se disponga.
Run	Coloca en funcionamiento el PLC.
Stop	Esta opción detendrá el PLC.
Estado del programa	Esta opción muestra gráficamente como interactúa nuestro programa.
Estado de tabla	Aquí se pueden ver los valores actuales del proceso.

Tabla 2-2. Herramientas utilizadas en MicroWin.

También se debe tener en cuenta las variables de control y sus direcciones dentro del programa. En la siguiente tabla se presentan las direcciones más importantes:

Variables	Dirección
Variable de proceso	VD12
Setpoint	VD0
Kc	VD132
Ti	VD140
Td	VD144
Salida PID	VD128

Tabla 2-3. Direcciones variables de control.

2.3 **INSTALACIÓN NODEJS:**

Antes de la instalación de NodeJS, se debe tener en cuenta que nuestro sistema operativo soporte una versión de este que sea compatible con Node-RED. Para esto, la página oficial de Node-RED hace entrega de la siguiente información:

Versión NodeJS	Nivel de soporte Node-RED
<8.x	No soportado
8.x	Soportado
10.x	Soportado
12.x	Soportado
14.x	Recomendado
16.x	Soportado

Tabla 2-4. Versiones de NodeJS que soportan Node-RED.

Con esta información, nos dirigimos a la página de descarga de NodeJS y descargamos la versión necesaria. Para este proyecto se utilizará la versión recomendada 16.17.0 LTS.

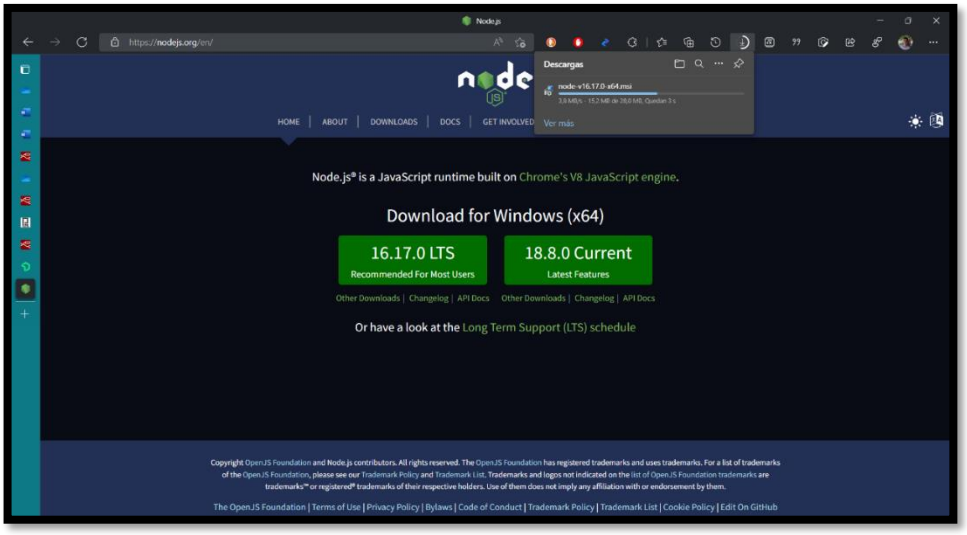

Figura 2-4. Página de descarga NodeJS.

En caso de necesitar otra versión, la página cuenta con otras versiones de NodeJS descargables.

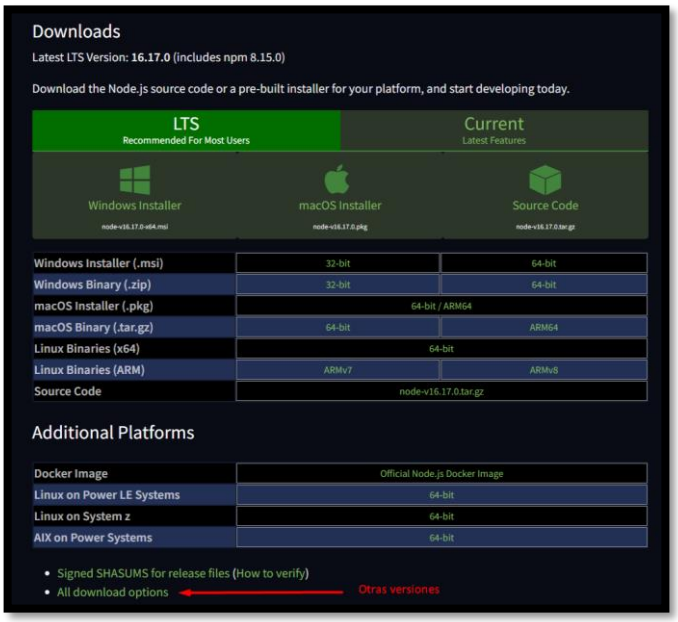

Figura 2-5. Página para descargar otras versiones NodeJS.

Luego de la descarga, se instala el ejecutable y ya se contará con NodeJS y el gestor de paquetes npm (Node Package Manager) dentro de la PC.

Se puede corroborar la instalación desde el CMD de Windows con los comandos node –v y npm –v.

2.4 INSTALACIÓN NODE-RED.

Para la instalación de Node-RED, se hará uso del gestor de paquetes npm. Por lo tanto, mediante el CMD se introduce el siguiente comando:

npm install -g --unsafe-perm node-red

Luego de esto, Node-RED estará listo para ser ejecutado desde el navegador web.

2.5 INSTALACIÓN BROKER EMQX.

Para este proyecto se ha seleccionado el bróker MQTT de código abierto EMQX. Este es uno de los más utilizados en los proyectos de IoT gracias a la gran escalabilidad que presenta.

Este bróker, será instalado en una VM (Virtual Machine) que estará alojada en la nube, con el fin de garantizar la disponibilidad del bróker en todo momento. Para realizar esto, dispondremos de los servicios que ofrece Google Cloud Plataform, que a pesar de que es un servicio de pago, nos ofrece un periodo de prueba de 3 meses y un presupuesto de 300$ que sería suficiente para el desarrollo del proyecto. Por lo tanto, lo primero que se debe hacer es registrarnos en la página de Google Cloud Plataform.

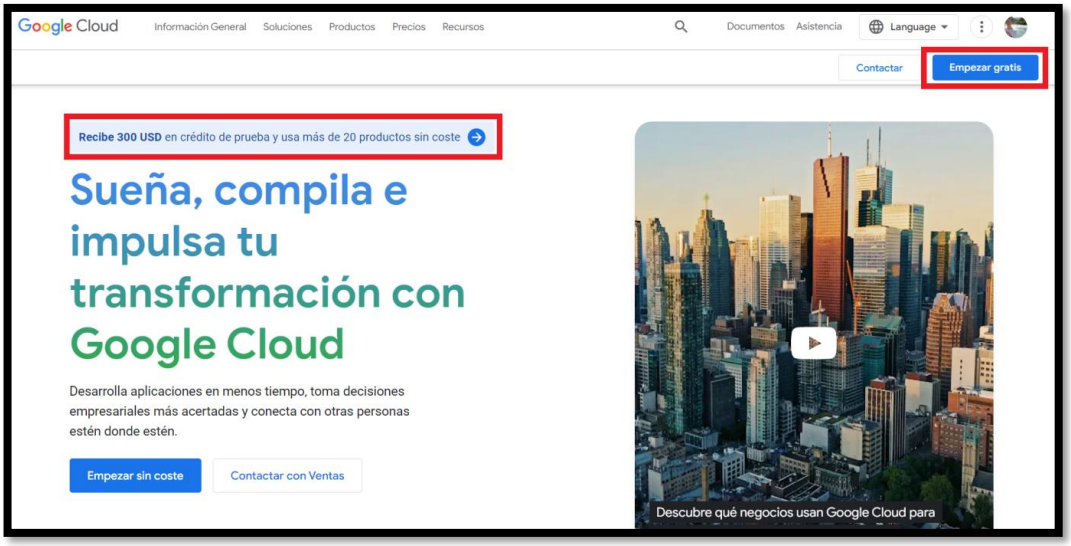

Figura 2-6. Página principal Google Cloud Plataform.

Se solicitará un número de teléfono y una tarjeta de crédito para confirmar la identidad del usuario. Luego de registrarse, se crea un nuevo proyecto.

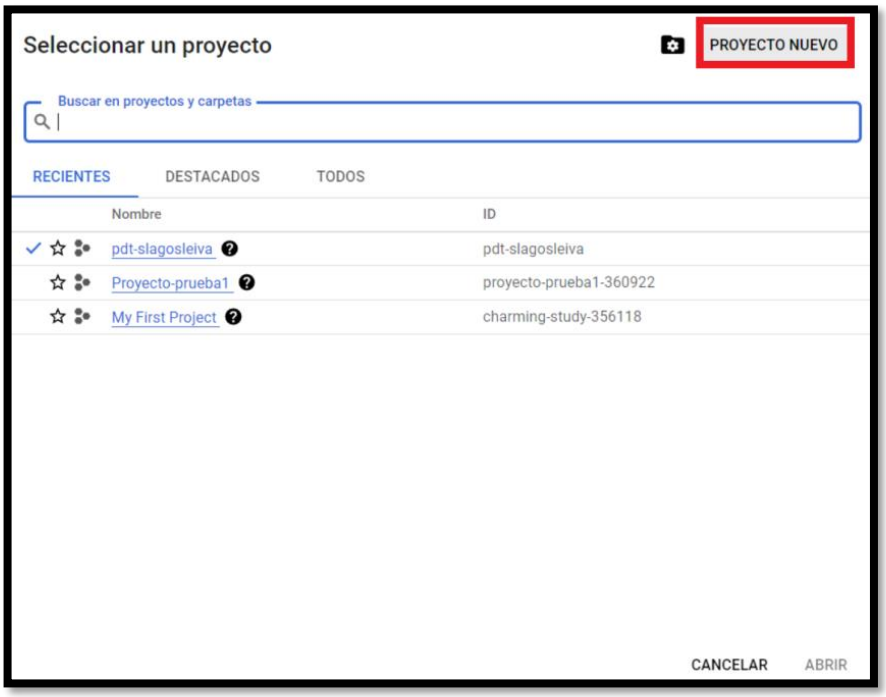

Figura 2-7. Ventana crear nuevo proyecto.

Dentro de este proyecto, se selecciona la opción de Compute Engine e Instancia de VM.

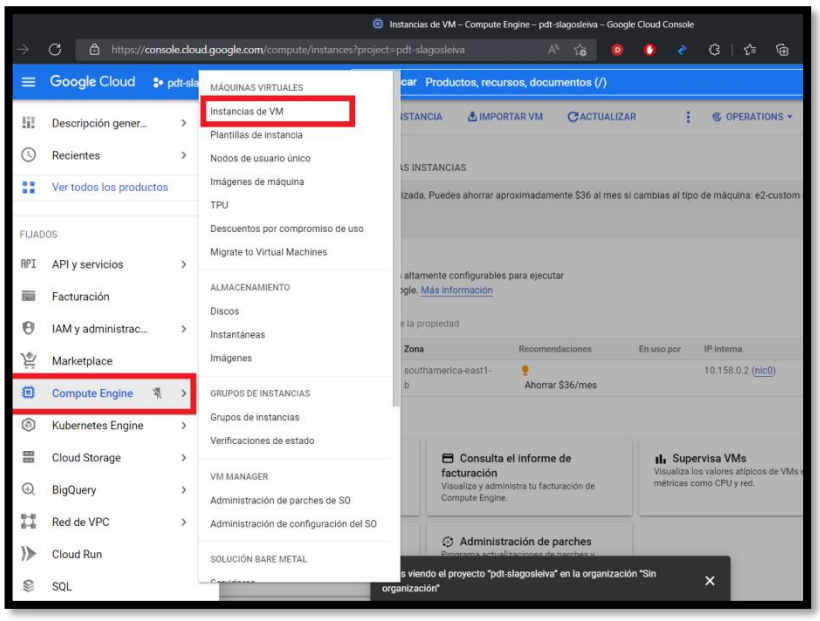

Figura 2-8. Creación VM parte uno.

En la ventana de VM se podrá especificar la región donde estará establecido el servidor, el tipo de máquina, sistema operativo, entre otras características. Dependiendo de estas, el servicio cobrará más o menos mensualmente.

Para este proyecto se utilizó una maquina e2-standard-2, ubicada en la zona southamerica-east1-b, y que cuenta con el sistema operativo Ubuntu 20.04 LTS.

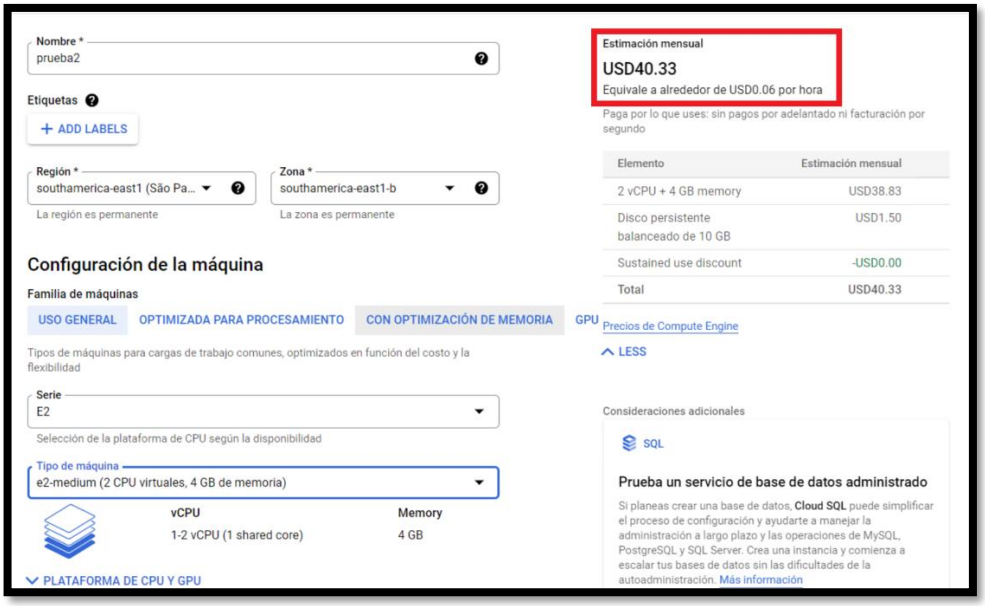

Figura 2-9. Creación VM parte dos.

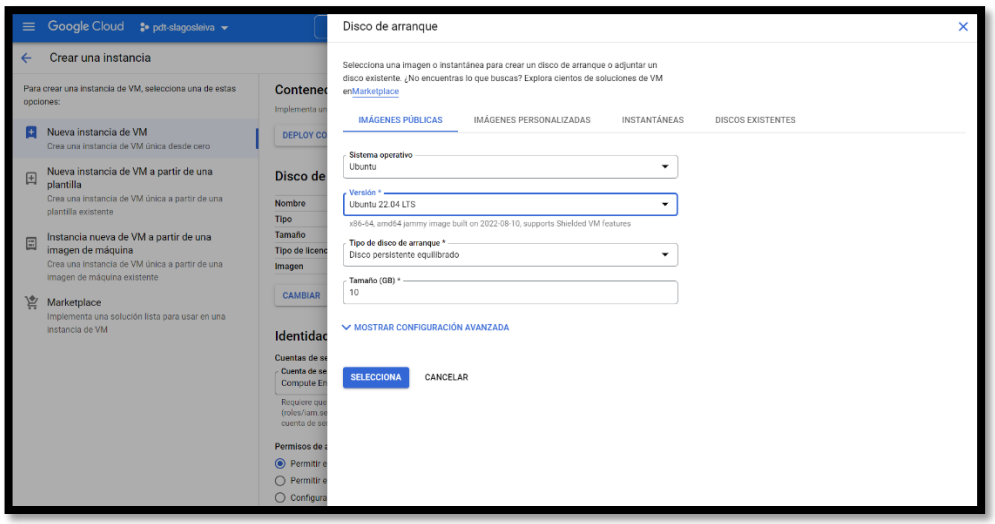

Figura 2-10. Creación VM parte tres.

Luego de creada la VM, Google Cloud hace entrega de la IP donde estará alojada la máquina virtual. Esta se debe reservar para que se mantenga fija durante todo el proyecto.

También se crean nuevas reglas de Firewall VPC, que controlarán el tráfico saliente o entrante de los datos con la instancia. Inicialmente se habilitan los siguientes puertos:

Puerto TCP	Definición
18083	Puerto del dashboard EMQX
1883	Puerto que utiliza el protocolo MQTT

Tabla 2-5. Puertos habilitados en Firewall VPC.

Terminado esto, se debe establecer la comunicación con el servidor a través del protocolo SSH. Esto permitirá que se abra la ventana de comandos de la VM.

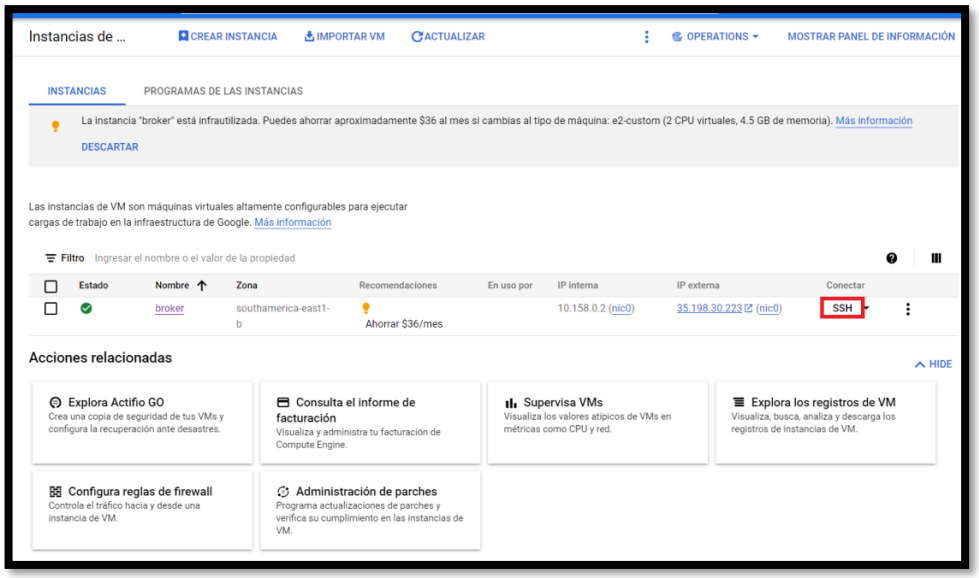

Figura 2-11. Página principal de VM.

A través de la ventana de comandos, se realiza la descarga e instalación del bróker EMQX.

Para esto se accedemos al siguiente enlace:

https://www.emqx.io/downloads?os=Ubuntu

Dentro de la página se encontrarán los comandos que se deben escribir dentro de la VM. Los comandos son los siguientes:

Función	Comando
Descarga del repositorio EMQX	curl -s https://assets.emqx.com/scripts/install-emqx-deb.sh | sudo bash
Instalación EMQX	sudo apt-get install emqx
Inicio de EMQX	sudo emqx start

Tabla 2-6. Comandos utilizados para instalar EMQX.

Luego de colocar en funcionamiento el bróker, se deben habilitar dentro de la VM los puertos mencionados anteriormente. Para esto se ingresan los siguientes comandos:

Función	Comando
Habilitar puerto 18083	sudo ufw allow 18083/tcp
Habilitar puerto 1883	ufw allow 1883/tcp

Tabla 2-7. Puertos habilitados dentro de la VM.

2.6 COMUNICACIÓN NODE-RED Y PLC S7-200.

Para realizar la comunicación entre Node-RED y el PLC, primero se debe iniciar Node-RED en la PC personal.

Para esto, se abre el CMD de la PC y se escribe el comando: “node-red”.

Luego en el navegador web se debe escribir la IP del PC seguido de: “:1880”. En este proyecto, la IP de trabajo sería:

http://127.0.0.1:1880/

Dentro de Node-RED, se busca e instala el nodo “node-red-contrib-s7”, el cual fue desarrollado por la comunidad para establecer la comunicación de esta aplicación con los PLC’s de siemens.

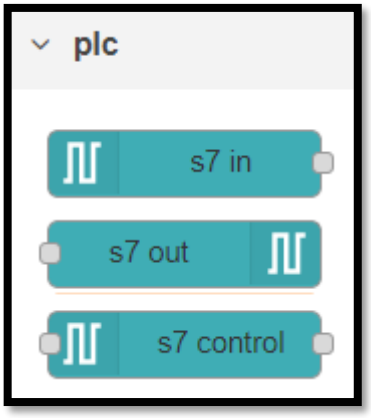

Figura 2-12. Nodo siemens.

De estos tres nuevos nodos, se utilizará el nodo “s7 in”, que permite realizar la lectura de las variables de proceso.

Al abrir la configuración de este nodo se debe registrar un nuevo PLC. Para esto se debe seleccionar la opción que se indica en la siguiente figura:

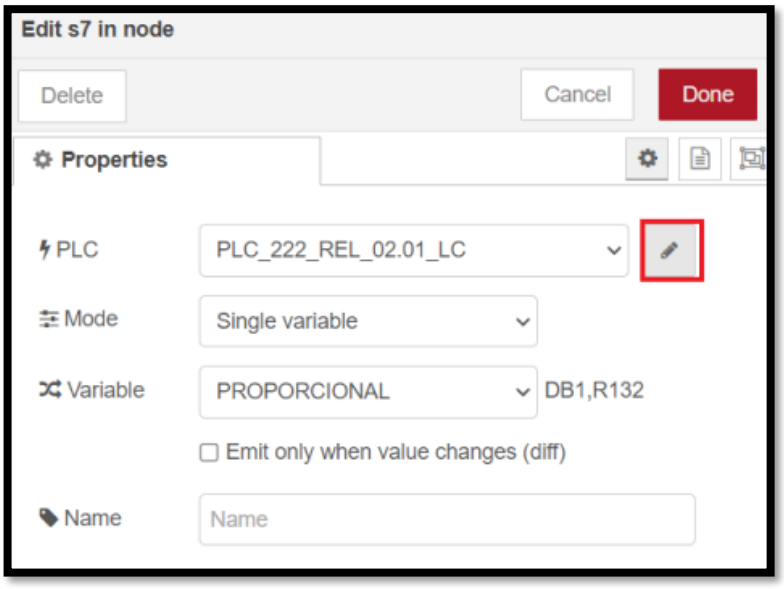

Figura 2-13. Configuración nodo siemens.

En la ventana de propiedades, se escribe la IP del PLC junto al nombre que se le asignará.

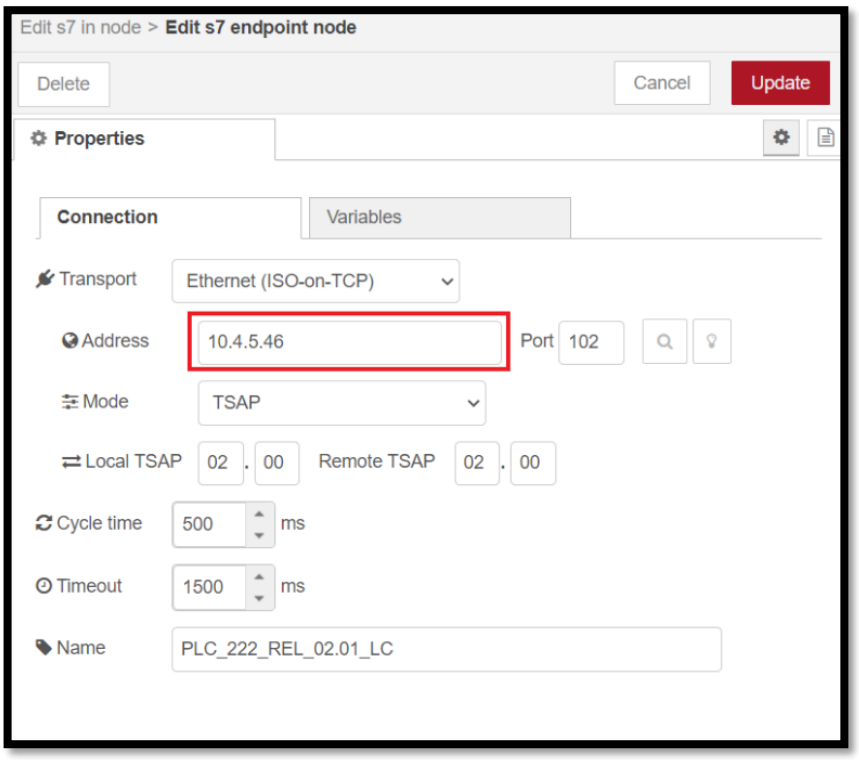

Figura 2-14. Configuración de IP dentro de nodo siemens.

También se deben registrar todas las variables que se usarán en el proyecto. Para esto, la información del nodo hace entrega de una tabla que indica cómo deben ser definido cada uno de los datos dependiendo el tipo que sea.

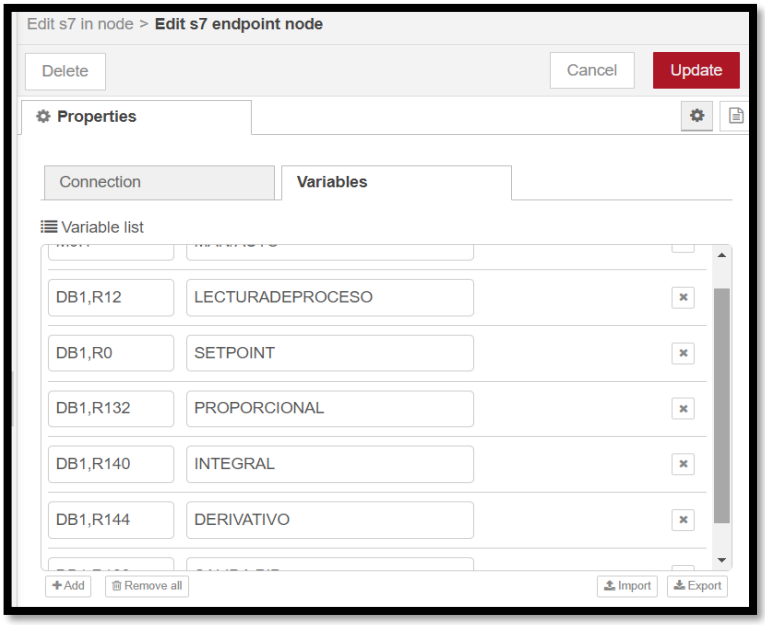

Figura 2-15. Configuración de variables de control dentro de nodo siemens.

En la siguiente tabla queda especificado como irán definidos las variables en este proyecto.

Asignación	Nombre
DB1,R12	LECTURADEPROCESO
DB1,R0	SETPOINT
DB1,R132	PROPORCIONAL
DB1,R140	INTEGRAL
DB1,R144	DERIVATIVO
DB1,R128	SALIDA PID

Tabla 2-8. Configuración de variables de control nodo siemens.

Luego, cada vez que se utilice el nodo de siemens, se debe especificar que variable se leerá.

Una vez configurado el nodo siemens, se pasa a configurar los nodos MQTT IN y MQTT OUT, los cuales permiten la transferencia de datos con el bróker.

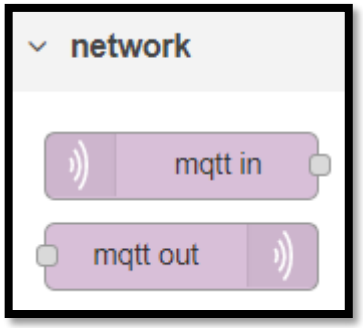

Figura 2-16. Nodos MQTT.

En la configuración de estos, se debe escribir la IP donde está alojado el bróker. Para esto, de igual forma que se configuró el nodo de PLC, se debe registrar un nuevo bróker.

En la siguiente figura se puede observar cómo quedaría la configuración del bróker. Cabe destacar que solo se configurará la conexión de este.

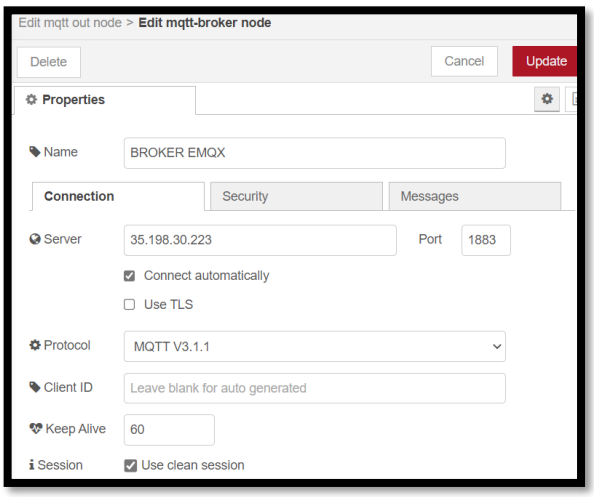

Figura 2-17. Configuración del bróker dentro de Node-RED.

Luego, cuando se utilice uno de estos nodos, se debe seleccionar el bróker correspondiente, además de tener definida la estructura de nuestros tópicos.

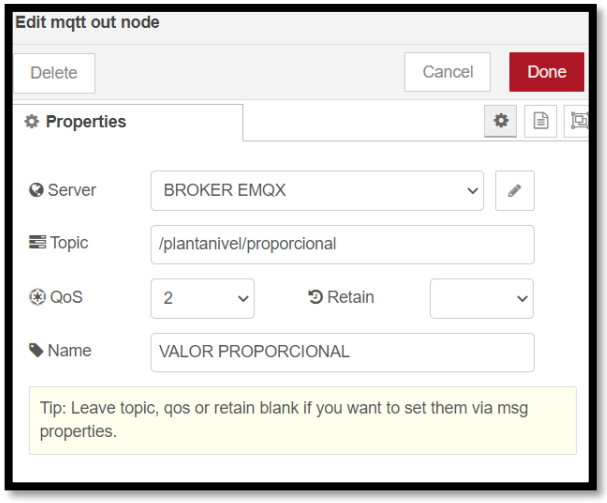

Figura 2-18. Ejemplo de configuración nodo MQTT OUT.

En este proyecto, los tópicos utilizados son los siguientes:

Tópico	Variable
/plantanivel/proporcional	Valor proporcional
/plantanivel/integral	Valor integral
/plantanivel/derivativo	Valor derivativo
/plantanivel/setpoint	Setpoint asignado
/plantanivel/vproceso	Nivel
/plantanivel/pid	Salida del PID

Tabla 2-9. Tópicos utilizados dentro del proyecto.

Al tener configurados estos dos nodos principales, se procede a realizar la programación para poder visualizar las variables establecidas.

Esta programación se puede demostrar en la siguiente figura:

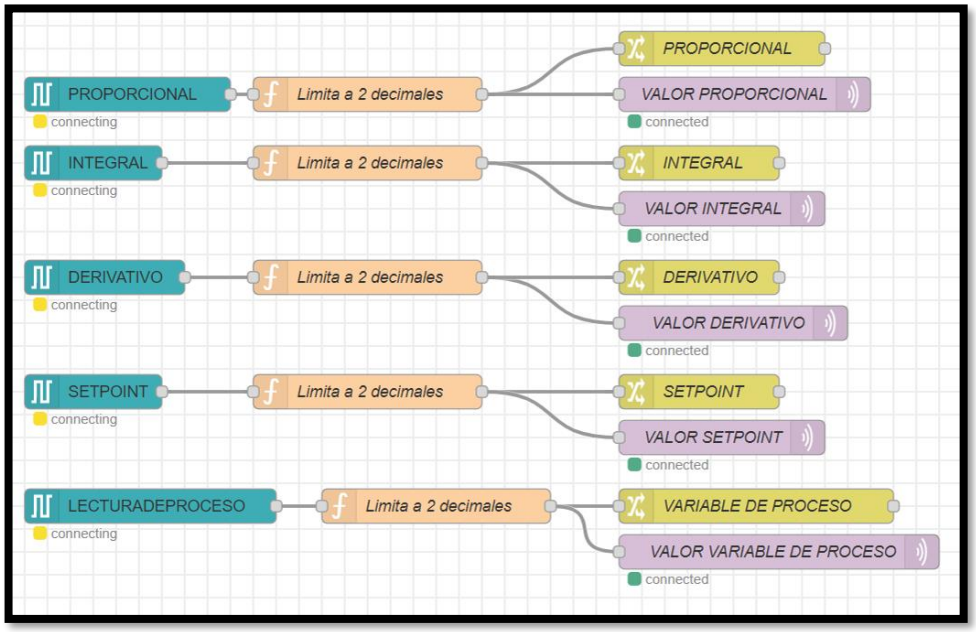

Figura 2-19. Programación para la extracción de datos.

En este flujo, el nodo de siemens extraerá el dato de la variable en interés y lo asignará a la variable msg. payload. Luego, a través del nodo de función, se limitarán los decimales del dato para luego ser enviados al bróker. Además, se ocupará el nodo change para convertir los mensajes en variables del tipo Flow y global. Esto se realiza con el fin de poder utilizar estas variables en otros flujos de programación.

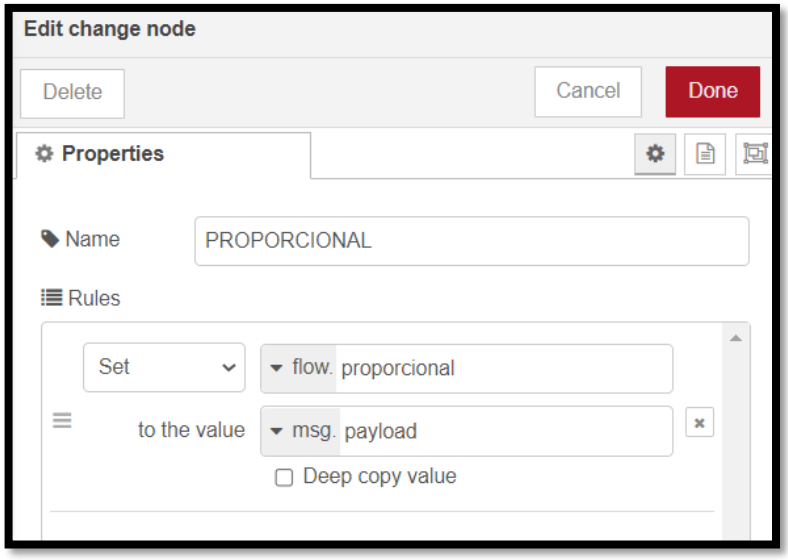

Figura 2-20. Configuración nodo change.

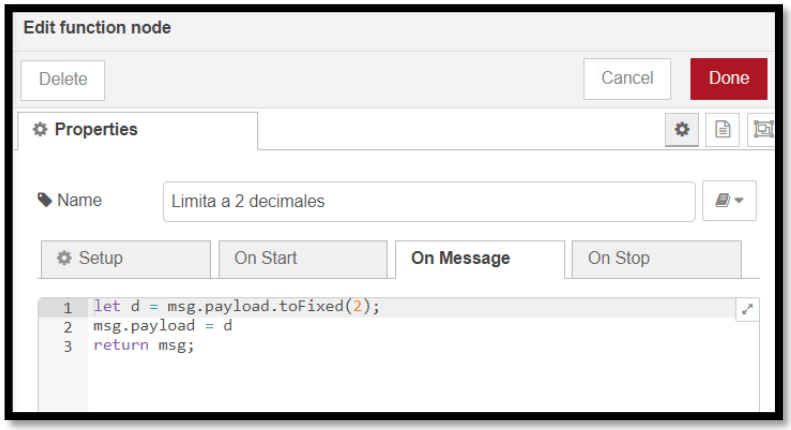

Figura 2-21. Configuración de nodo función para la reducción de decimales en los datos.

Como los datos ya fueron entregados al bróker, los usuarios que estén suscritos a los tópicos mencionados ya estarían recibiendo los mensajes del proceso.

Para hacer demostración de esto, se creará en otro flujo una programación que reciba los datos del bróker y los muestre en un dashboard o KPI.

2.7 DASHBOARD NODE-RED.

Para la implementación de este dashboard, se debe descargar e instalar el nodo de "node-red-dashboard". Este trae consigo diferentes nodos que permitirán la visualización gráfica en el dashboard.

La programación se muestra en la siguiente figura:

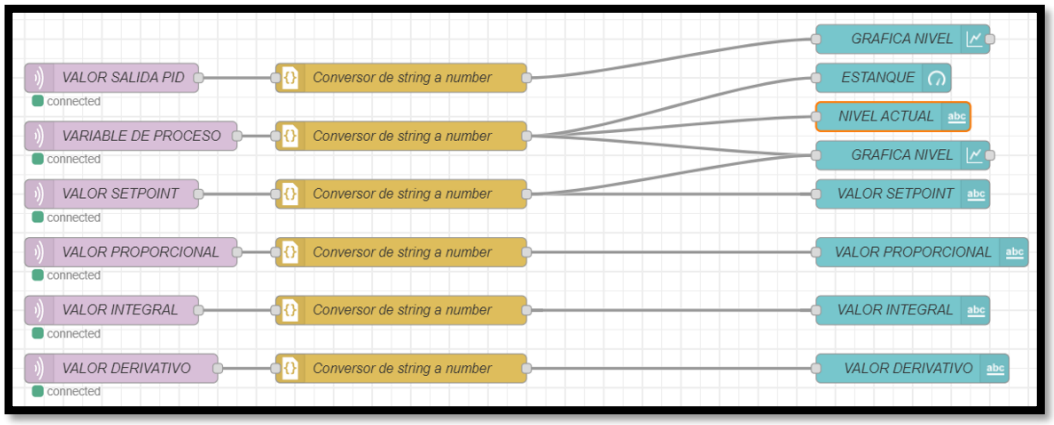

Figura 2-22. Programación dashboard.

Cada flujo se compone de un nodo MQTT IN, un nodo JSON y el nodo de visualización gráfica. El nodo MQTT IN se configura de la misma manera que el nodo MQTT OUT. El nodo JSON transformará la propiedad del dato, ya que llega en formato string y se necesita en formato number. En los últimos nodos, se tienen gráficas, animaciones y textos que podemos ubicar en el dashboard.

La configuración de los nodos de dashboard se muestran en las siguientes figuras:

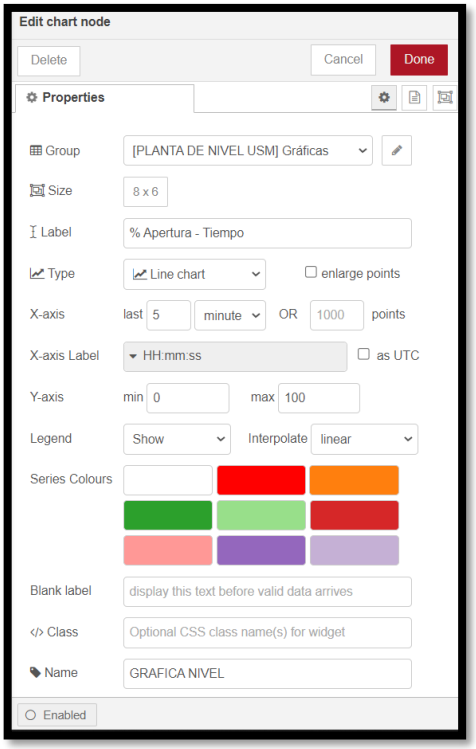

Figura 2-23. Configuración nodo chart.

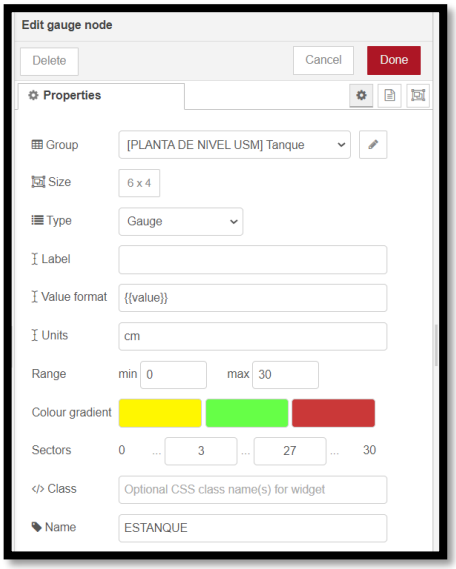

Figura 2-24. Configuración nodo gauge.

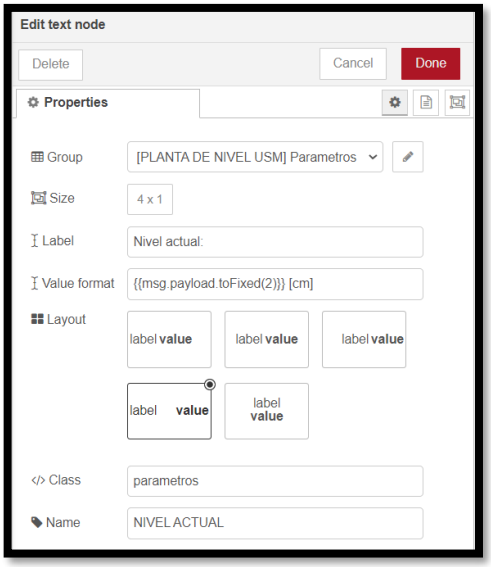

Figura 2-25. Configuración nodo de texto.

En este punto, se podría exportar este flujo e instalarlo en cualquier dispositivo compatible con Node-RED. Para realizar esto, se debe exportar el programa en formato JSON o descargando el archivo.

De esta manera, el dashboard realizado se puede ver en la siguiente figura:

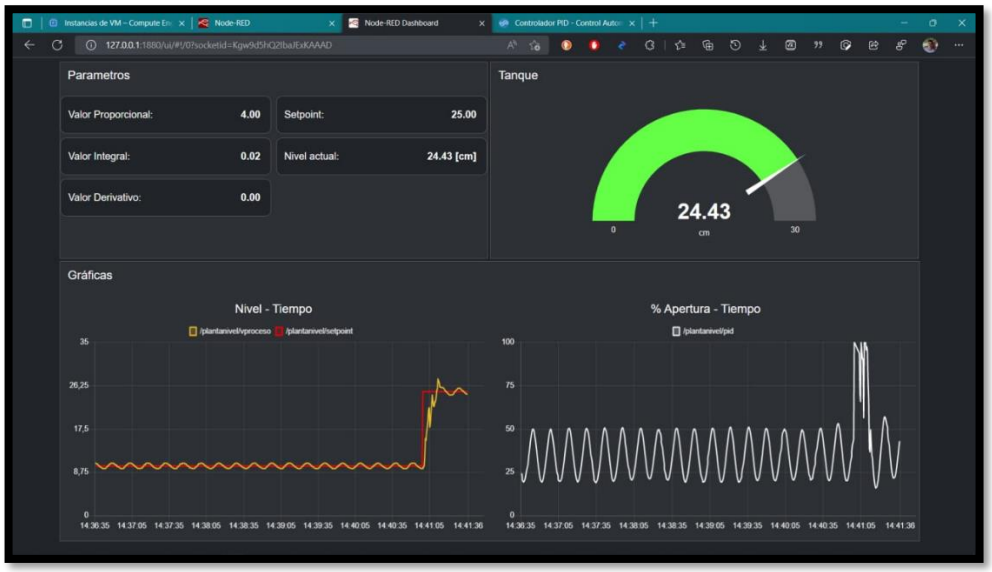

Figura 2-26. Dashboard del proceso.

En este, se pueden ver los valores del proceso y las gráficas de nivel y apertura de la válvula.

2.8 APLICACIÓN EN TELEGRAM.

Como ya se mencionó, en este proyecto se desarrollarían dos formas de visualizar los datos del proceso. Una sería a través del dashboard ya mostrado y la otra, a través de la aplicación de mensajería Telegram.

Con esta aplicación se desarrollará un Bot o usuario ficticio que esté conectado con el proceso a través de Node-RED, de esta manera, se le podrá solicitar información desde los dispositivos móviles.

Para esto, lo primero que se debe hacer, es dirigirse a la tienda de aplicaciones del dispositivo a utilizar y descargar Telegram. Esta aplicación es compatible con teléfonos inteligentes, tabletas o computadoras. Como ejemplo, para este proyecto se utilizó la aplicación de Google Play para descargar e instalar Telegram.

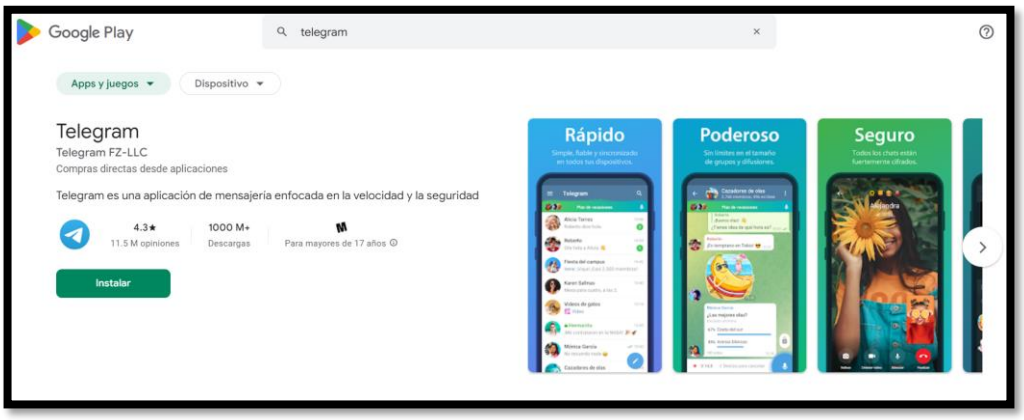

Figura 2-27. Aplicación Telegram.

Dentro de Telegram, se debe buscar el usuario @BotFather, API que dispone la aplicación para desarrollar los Bots.

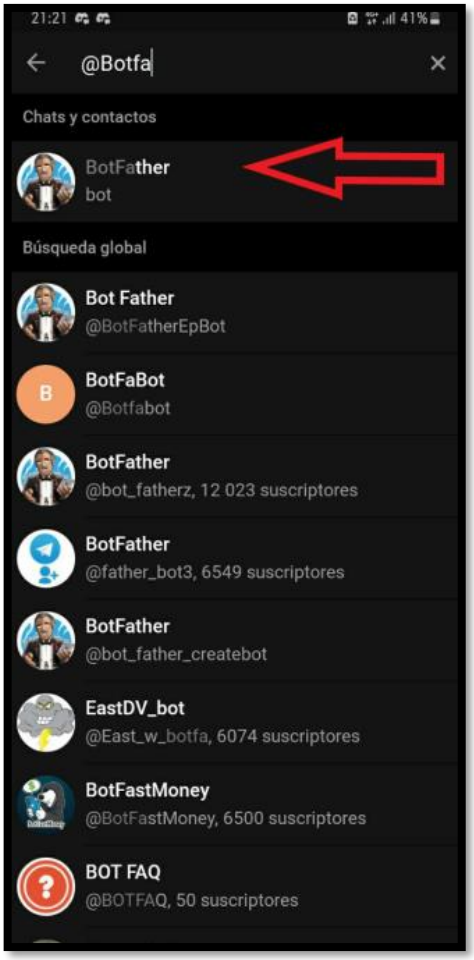

Figura 2-28. Bot @BotFather.

Una vez encontrado el usuario, se debe establecer un chat con el Bot y enviarle el comando /start. El Bot responderá de manera automática con la información necesaria para crear un nuevo Bot.

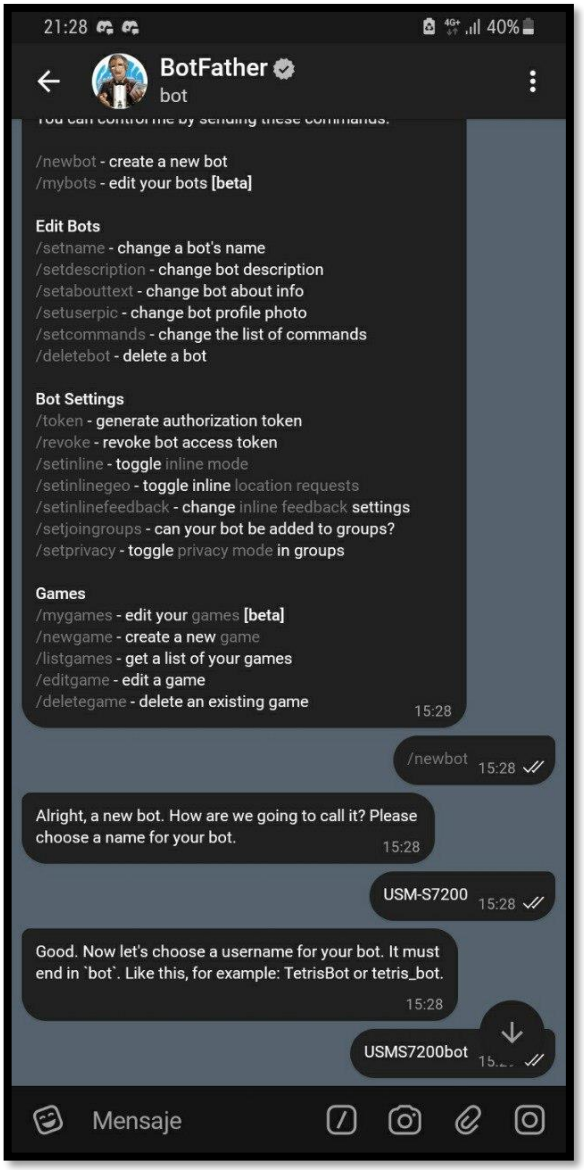

Figura 2-29. Instrucciones por seguir para crear un nuevo Bot.

Después de crear el bot, se hace entrega del token que permitirá la comunicación de otras aplicaciones con el bot creado. Para este proyecto, la aplicación que se utilizará será Node-RED. Por lo tanto, se debe descargar el nodo “node-red-contrib-telegrambot” y configurarlo de la siguiente manera:

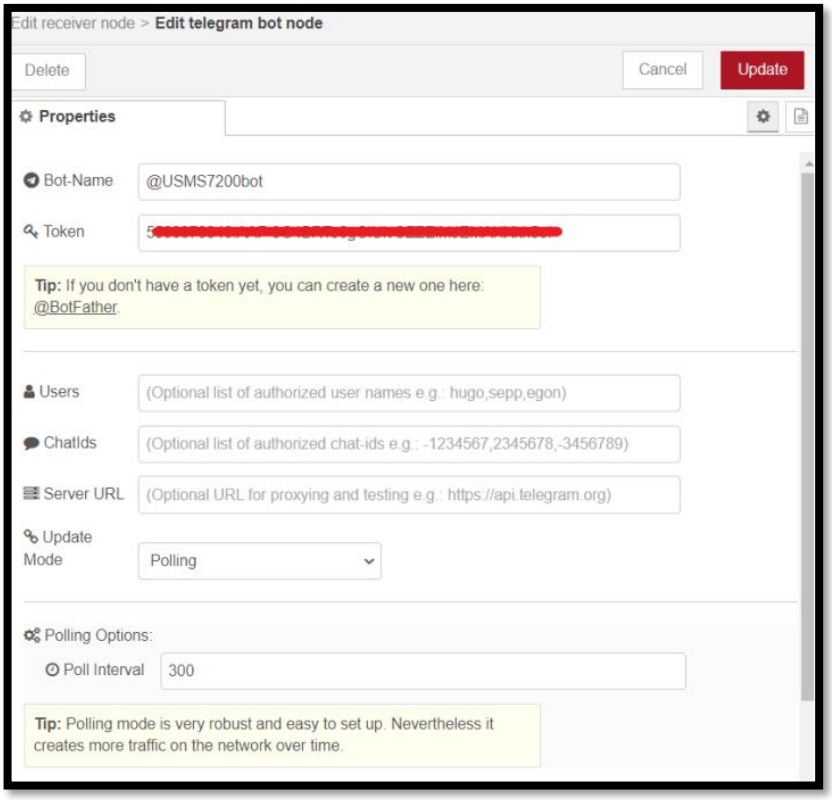

Figura 2-30. Configuración del nodo Bot dentro de Node-RED.

La programación se realizará en el mismo flujo donde se estableció la comunicación con el bróker. De esta manera, el flujo de Telegram quedaría de la siguiente manera:

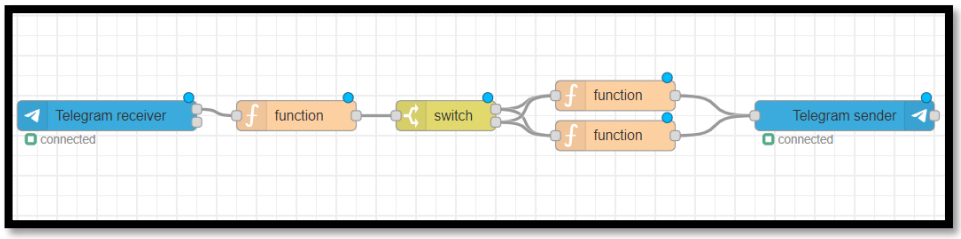

Figura 2-31. Flujo Telegram.

El primer nodo estaría recibiendo los mensajes que se le envíen al Bot creado. Este mensaje se puede visualizar de manera rápida conectando a la salida del nodo un nodo Debug. De esta forma, desde la ventana de reportes Debug, se podrán ver los mensajes recibidos y sus características. Un ejemplo de esto se muestra en las siguientes figuras:

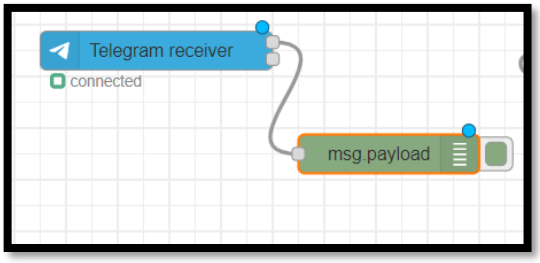

Figura 2-32. Demostración nodo debug: configuración del flujo.

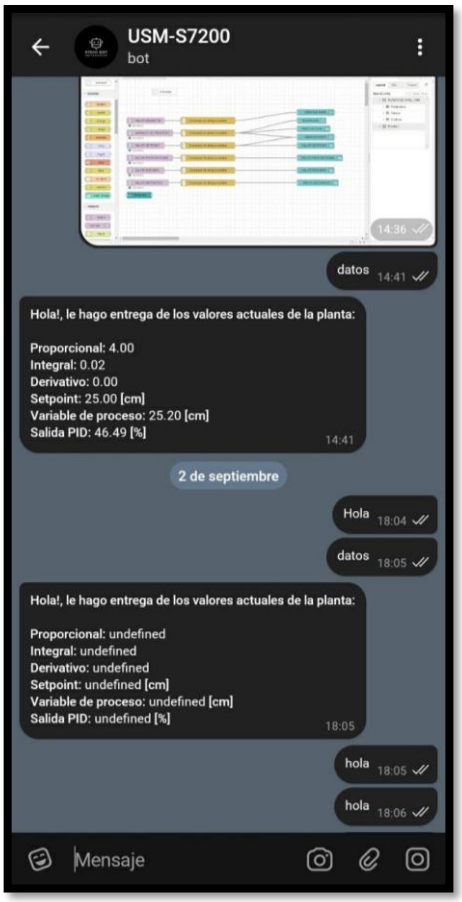

Figura 2-33. Demostración nodo debug: mensaje enviado.

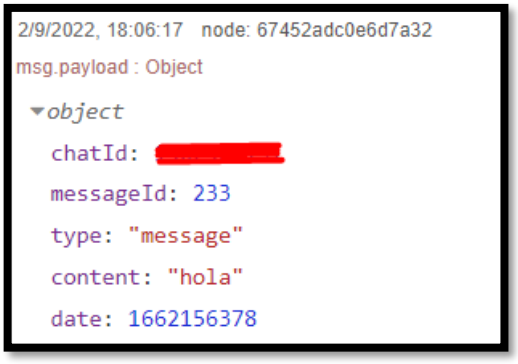

Figura 2-34. Demostración nodo debug: Mensaje recibido.

Como se puede ver en la figura 2-35, el mensaje recibido por Node-RED es un mensaje tipo objeto, el cual contiene diferentes datos dentro de él. Estos datos son:

Datos	Significado
chatId	Identificador de nuestro usuario.
messageId	Identificador del mensaje enviado.
type	Tipo de mensaje.
content	El contenido del mensaje.
date	La fecha del mensaje.

Tabla 2-10. Datos enviados desde chat de Telegram.

De estos sub-mensajes, solo se utilizará el msg. content, ya que este contendrá el mensaje de instrucción para que el Bot tome los datos del proceso y los envíe al chat.

Para esto, a través de un nodo function se realiza un script que convertirá el dato recibido. El script realizado es el siguiente:

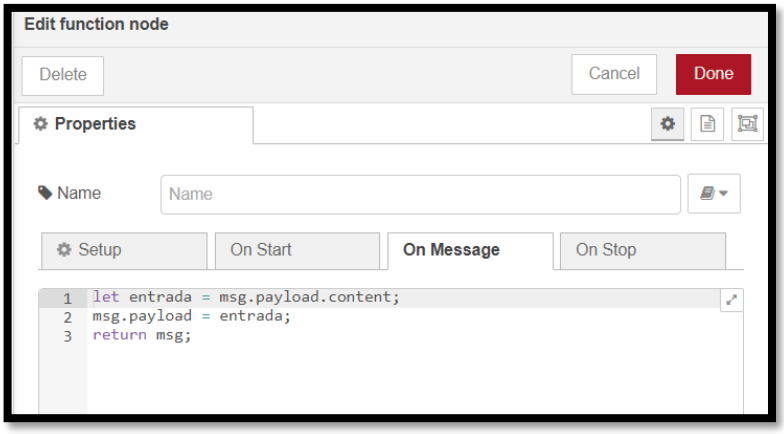

Figura 2-35. Código para convertir dato tipo objeto.

En este script, se crea una variable llamada “entrada” a la que se le asigna el valor del msg.payload.content. Luego, esta variable se define como el nuevo msg. payload. Al repetir la prueba anterior, se puede ver que el mensaje recibido ya no es tipo object, sino que tipo string.

Figura 2-36. Cambio de dato tipo objeto.

En este punto, se ha definido que el mensaje con el cual el Bot realizará la acción será la palabra “datos” o “Datos”. Por lo tanto, siguiendo el flujo creado, se tiene un nodo Switch que permitirá filtrar los mensajes de entrada. La configuración de este nodo se ve en la siguiente figura:

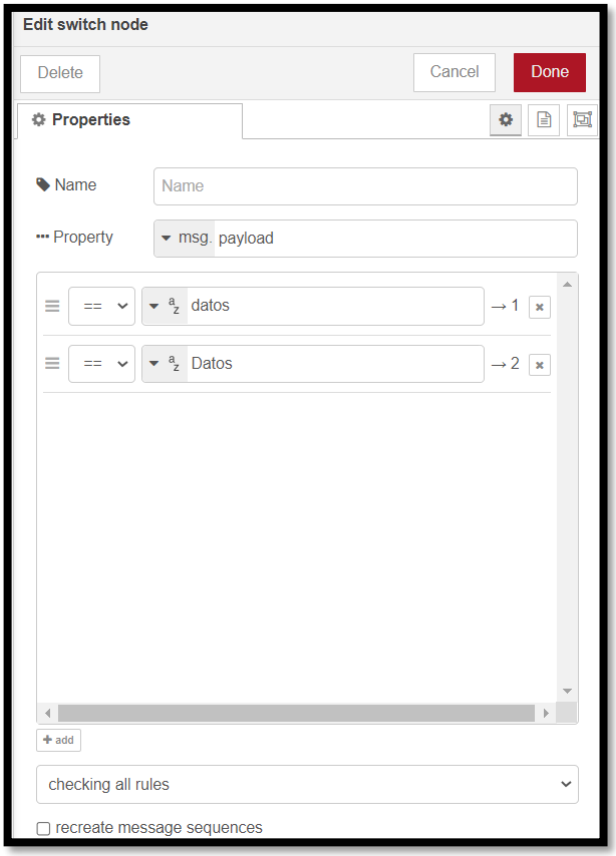

Figura 2-37. Configuración nodo switch para filtrar los mensajes recibidos.

De esta manera, el Bot solo enviara los datos cuando reciba el mensaje “datos” o “Datos”.

Seguido de este filtro se tiene el nodo function que contiene el script que permitirá crear un mensaje con los datos del proceso y enviarlos al chat de Telegram.

Este script se representa en la siguiente figura:

Figura 2-38. Código realizado para la configuración del mensaje a enviar.

De la línea 1 a 6 se crearon variables y se asignaron los valores que fueron extraídos del PLC. En esta asignación se utilizó la función flow.get para llamar a dichas variables.

En la línea 7, se escribió el mensaje que se enviará cuando se soliciten los datos.

En la línea 8, se define el contenido del mensaje, el tipo de mensaje y a quien será enviado el mensaje. Esto es necesario para que el envío se lleve a cabo.

En la línea 9 se establecen las opciones del mensaje. El Mark Down ayudará a personalizar el mensaje enviado.

Finalmente se coloca el nodo sender de Telegram y le asignamos el Bot.

Ahora, al enviar el mensaje “datos” al Bot, este tomará los datos del proceso y los enviará al chat Id establecido.

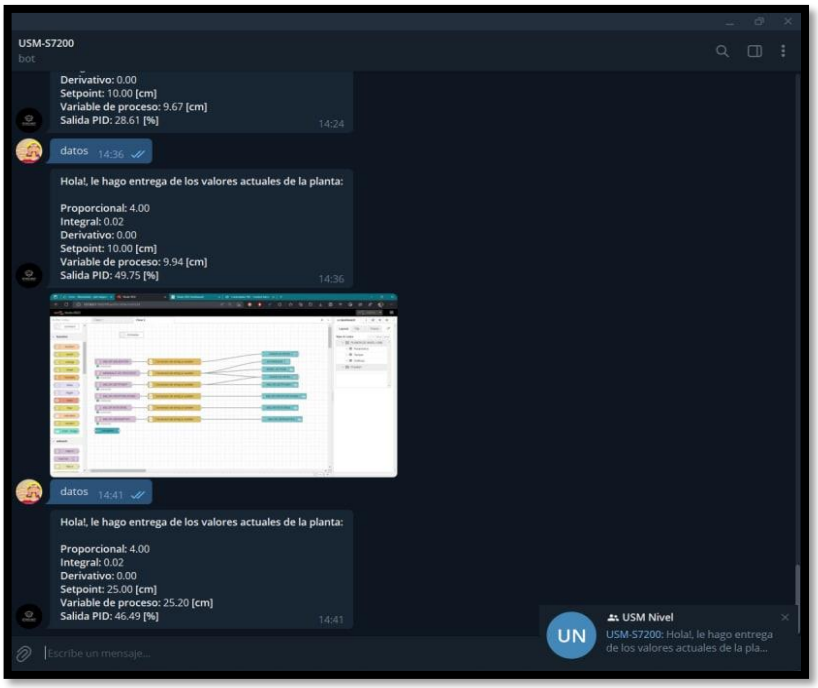

Figura 2-39. Demostración de funcionamiento del Bot.

De esta forma, se tiene una manera más rápida y sencilla de obtener los datos del proceso.

CONCLUSIÓN

La implementación de la herramienta Node-RED en el proceso escogido, ha permitido desarrollar dos aplicaciones para la visualización de las variables de proceso a través de internet.

La primera aplicación está enfocada para ser implementada en lugares fijos, ya que provee de un dashboard que contiene gráficas e imágenes que se irán actualizando en tiempo real.

La segunda aplicación, fue desarrollada para brindar al usuario una manera de acceder a los datos del proceso de forma rápida y sencilla. Para esto, se dispone de un Bot el cual entregará los datos cuando se soliciten.

Ambas aplicaciones podrían llegar a ser totalmente gratuitas dependiendo de los servicios que se ocupen para su desarrollo. En el caso de este proyecto, el único servicio de pago es la máquina virtual en la que es alojado el bróker MQTT. Esto hace más fiable la transmisión de datos, ya que el bróker estará siempre activo en los servidores de Google. Pero cabe destacar que esta opción no es obligatoria, ya que este bróker puede ser instalado en otra computadora o también en dispositivos como la raspberry pi.

Para este proceso, la ventaja que ofrecería la implementación de estas aplicaciones sería totalmente educativas, ya que es un proceso en el cual los alumnos de nuestra universidad aprenderían y practicarían sobre los procesos industriales. No obstante, estas aplicaciones podrían ser la guía para que en un futuro los estudiantes puedan aplicar la monitorización de procesos a través de internet en su quehacer profesional.

BIBLIOGRAFÍA

1. BRAUN, Andrew. History of iot: a timeline of development - iot tech trends. IoT Tech Trends [en línea]. [sin fecha] [consultado el 7 de mayo de 2022]. Disponible en: https://www.iottechtrends.com/history-of-iot/

2. ELDER, Jeff. The internet's first thing – John Romkey's 'smart' toaster. Avast Blog [en línea]. 3 de septiembre de 2019 [consultado el 7 de mayo de 2022]. Disponible en: https://blog.avast.com/the-internets-first-smart-device

3. ELETTRONICA VENETA. Level control - elettronica veneta s.p.a. Elettronica Veneta S.p.A. [en línea]. [sin fecha] [consultado el 7 de mayo de 2022]. Disponible en: https://www.elettronicaveneta.com/en/prodotto/level-control/

4. GOOGLE CLOUD PLATAFORM. Creating and starting a VM instance | compute engine documentation | google cloud. Google Cloud [en línea]. [sin fecha] [consultado el 13 de abril de 2022]. Disponible en: https://cloud.google.com/compute/docs/instances/create-start-instance

5. HERRERA, Diana. Qué es Node.js: casos de uso comunes y cómo instalarlo. Tutoriales Hostinger [en línea]. 8 de julio de 2022 [consultado el 4 de agosto de 2022]. Disponible en: https://www.hostinger.com.ar/tutoriales/que-es-node-js

6. LUIS LLAMAS. ¿Qué es MQTT? Su importancia como protocolo IoT. Luis Llamas [en línea]. 17 de abril de 2019 [consultado el 20 de octubre de 2021]. Disponible en: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

7. MQTT - the standard for iot messaging. MQTT - The Standard for IoT Messaging [en línea]. [sin fecha] [consultado el 4 de septiembre de 2021]. Disponible en: https://mqtt.org/

8. PAESSLER. ¿Qué es MQTT? Definición y detalles. Paessler - The Monitoring Experts [en línea]. [sin fecha] [consultado el 4 de agosto de 2022]. Disponible en: https://www.paessler.com/es/it-explained/mqtt

9. Sobre Node-RED. Node-RED [en línea]. [sin fecha] [consultado el 11 agosto de 2021]. Disponible en: https://nodered.org/about/

10. Telegram FAQ. Telegram [en línea]. [sin fecha] [consultado el 7 de mayo de 2022]. Disponible en: https://telegram.org/faq