

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE DE CONCEPCIÓN – REY BALDUINO DE BÉLGICA

IMPLEMENTACIÓN DE DASHBOARD MEDIANTE EL USO DE MQTT.

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

Alumno:

Ricardo Alexander Varela Basoalto

Profesor Guía:

Nelson Artemio Vásquez Concha

2021

RESUMEN

El presente trabajo de título describe un proyecto que será el paso inicial para implementar de forma eficiente la comunicación MQTT para lo que sería el IoT (orientado al público en general) y también orientado para el IIoT (industria 4.0).

Esto mediante el uso de diferentes softwares que en su conjunto brindan el servicio deseado para el usuario final, los clientes en este caso. Partiendo por la generación de una instancia en la Nube con ayuda de la plataforma GCP, en dicha instancia se instalará un SO de Ubuntu, el cual mediante consola de comandos se le añadirán un Broker MQTT, Node-RED.

Esto para poder representar en la nube lo que sería una simulación o un caso práctico de una programación en PLC, el cual será accesible por medio de un KPI a varios usuarios en simultaneo, todo esto será visible gracias a la comunicación MQTT que se establecerá entre el software local y el que estará en la Nube.

ÍNDICE

INTRODUCCIÓN:	1
Objetivo General:	2
Objetivos Específicos:	2
1.1 PROTOCOLOS DE COMUNICACIÓN	4
1.1.1 Algunos protocolos de comunicación en la industria:	4
1.1.1.1 HART (Highway Addressable Remote Transducer):	5
1.1.1.2 Profibus DP (Decentralized Periphery):	5
1.1.1.3 Profibus PA (Process Automation):	5
1.1.1.4 Profibus FMS (Fieldbus Message Specification):	5
1.1.1.5 Foundation Fieldbus (FF):	6
1.1.1.6 MODBUS:	6
1.2 MQTT.....	7
1.2.1 Protocolo de comunicación MQTT:	7
1.2.2 ¿Qué es MQTT?	7
1.2.3 ¿Cómo funciona el MQTT?	7
1.2.4 Estructura de un mensaje MQTT:	10
1.2.5 Calidad del servicio:	11
1.2.6 Seguridad en MQTT:	12
1.2.7 Ventajas del MQTT:	12
1.3 BROKER.....	13
1.3.1 Definición:	13
1.3.2 Estructura:	13
1.3.3 Principales brokers:	15
1.3.3.1 Broker EMQ X:	16
2.1 PLC M221:	19
2.1.1 Descripción general del PLC TM221C:	19
2.1.1.1 Lenguajes de Programación:	19
2.1.2 Características físicas (TM221CE16R):	19
2.2 PROGRAMACIÓN EN M221:	20
2.2.1 Bloques ocupados en la programación:	22

2.2.2 Simulación de estructura Ladder de un registro por pulsos internos:	23
2.2.3 Simulación de estructura Ladder de un contador y secuencia de LEDs:.....	24
3.1 GOOGLE CLOUD:	26
3.1.1 ¿Qué es Google Cloud?:	26
3.1.2 Instancia VM:	26
3.1.2.1 ¿Qué es una Instancia:	26
3.1.2.2 Instancias y Proyectos:	27
3.1.2.3 Instancias y opciones de almacenamiento:	27
3.1.2.4 Instancias y Redes:	27
3.1.2.5 Instancias y contenedores:	27
3.1.3 Creación de Instancia:	28
3.2 NODE RED:	33
3.2.1 ¿Qué es Node RED?:.....	33
3.2.1.1 Edición de flujo basada en navegador:	33
3.2.1.2 Estructura base:.....	33
3.2.2 Instalación de Node.js y Node RED en la instancia creada:	34
4.1 INTERFAZ Y HERRAMIENTAS DE NODE-RED:	41
4.1.1 Nodos:	41
4.1.1.1 Algunos Nodos Ocupados en Tabla 4-3:.....	41
4.2 FLUJO CREADO EN NODE-RED:.....	42
4.2.1 Comunicación Local entre Node-RED y la simulación del PLC:	42
4.2.2 Local (Simulación):.....	44
4.2.3 Instancia (Simulación):	47
4.3 DASHBOARD (KPI):	49
4.3.1 Simulación:	49
5.1 GCP:	54
5.1.1 Apache:	54
CONCLUSIONES:	56
BIBLIOGRAFIA:	57
ANEXO 1:	59
ANEXO 2:	60
ANEXO 3:	62

ÍNDICE DE FIGURAS

Figura 1-1. Publicador a suscriptor a través de un broker.....	8
Figura 1-2. Mensaje y respuesta entre cliente y broker.....	8
Figura 1-3. Publicación de mensaje hacia el broker	9
Figura 1-4. Mensaje de suscripción entre cliente y broker	9
Figura 1-5. Estructura de un mensaje MQTT.....	10
Figura 1-6. Logo MQTT EMQ X.....	16
Figura 1-7. Dashboard de EMQ X y los Topics creados	17
Figura 1-8. Conexiones Local y en la Nube.....	17
Figura 1-9. Suscripciones realizadas por la comunicación MQTT	18
Figura 1-10. Lista de algunos puertos utilizados para realizar la comunicación.....	18
Figura 2-11. PLC M221	19
Figura 2-12. Pantalla de Inicio software EcoStruxure Machine Expert Basic	20
Figura 2-13. Selección del PLC para ocupar en la configuración del programa.....	20
Figura 2-14. Modbus TCP del PLC M221	21
Figura 2-15. Programación Ladder de un registro por pulsos internos.....	23
Figura 2-16. Programación Ladder de un contador	24
Figura 2-17. Programación Ladder de una secuencia con LEDs	24
Figura 2-18. Logotipo GCP	28
Figura 3-19. Localización de la herramienta Compute Engine	28
Figura 3-20. Panel de Compute Engine	28
Figura 3-21. Configuración para la creación de instancia.....	29
Figura 3-22. Sistema operativo montado en la instancia	30
Figura 3-23. Permisos de Firewall	30
Figura 3-24. Asignación de IP estática para la instancia	31
Figura 3-25. Configuración para abrir puerto TCP 1880	32
Figura 3-26. Abriendo consola mediante la opción SSH.....	34
Figura 3-27. Consola de comando de Ubuntu dentro de la instancia	34
Figura 3-28. Paquetes esenciales de compilación	35
Figura 3-29. Dar acceso a ocupar espacio de memoria para la instalación del paquete	35
Figura 3-30. Comando de descarga para el software Node.js.....	35

Figura 3-31. Lista de operaciones con descarga completada de Node.js..... 36

Figura 3-32. Comando de instalación de Node.js..... 36

Figura 3-33. Instalación correcta de Node.js 37

Figura 3-34. Comando de instalación Node-RED 37

Figura 3-35. Node-RED instalado correctamente..... 38

Figura 3-36. Comando para inicializar Node-RED 38

Figura 3-37. Node-RED Inicializado..... 38

Figura 3-38. Dirección estática de la instancia en donde se encuentra Node-RED..... 39

Figura 3-39. Entorno de programación de Node-RED..... 39

Figura 4-40. Nodo Modbus Read en Node-RED 42

Figura 4-41. Configuración Modbus TCP..... 43

Figura 4-42. Configuración Partir 44

Figura 4-43. Flujo creado de lectura Modbus de partida y registro..... 45

Figura 4-44. Configuración del servidor MQTT en el Nodo 45

Figura 4-45. Tópico declarado para la lectura de partir 46

Figura 4-46. Flujo de Contador y secuencia de LEDs 46

Figura 4-47. Prueba de deslizador..... 47

Figura 4-48. Registro de pulsos con verificación ON/OFF (LED)..... 48

Figura 4-49. Flujo con los Nodos MQTT asociados a los tópicos y representación del Dashboard 48

Figura 4-50. Dashboard de prueba del deslizador y el gauge 49

Figura 4-51. Dashboard con el gauge de registro, contador y secuencia de LEDs 50

Figura 5-52. Permisos de firewall para accesos HTTP y HTTPS 54

ÍNDICE DE TABLAS

Tabla 1-1. Tipos de mensajes y sus códigos del protocolo MQTT 11

Tabla 2-2. Contactos de programación Ladder 22

Tabla 4-3. Nodos Ocupados para la programación de Node-RED 42

SIGLAS Y SÍMBOLOGIA

GCP:	Google Cloud Plataform
PLC:	Programmable Logic Controller
MQTT:	Message Queuing Telemetry Transport
TCP:	Trasmission Control Protocol
HTTP:	Hypertext Transfer Protocol
VPS:	Servidor Virtual Privado
M2M:	Machine to Machine
KPI:	Key Performance Indicator
HMI:	Human Machine Interface
VM:	Virtual Machine

INTRODUCCIÓN:

En los últimos tiempos se ha visto un desarrollo notable en lo que es la comunicación entre M2M, visualización y control por HMI en la industria o de forma pública (IoT), y actualmente se está avanzando aún más en la adaptación e implementación del IIoT, la que viene siendo la llamada Industria 4.0.

Tanto así que se está viendo cada vez más el manejo de softwares especializados en control y visualización de datos de forma remota en la industria, en vez de recurrir a una HMI que se encuentra entregando datos, pero de forma Local, la industria 4.0 actualmente requiere más que eso, ya que se espera que los operadores puedan ver datos de algún proceso ya desde sus mismos dispositivos móviles, por ejemplo.

Este proyecto va enfocado a la visualización remota de variables que puedan ser generadas en algún tipo de proceso en industria y que sea fácil de entender e interpretar, que a su vez esté al alcance y el acceso sea rápido, por lo cual se llevara a cabo la implementación del llamado KPI, este tiene como objetivo ser visto por los múltiples usuarios que la industria requiera, esto logrado con la comunicación MQTT, que en simples maneras es el acceso a datos por petición de permisos a un broker, entre publicador y subscriptor (cliente/usuario).

Esta es la tecnología de la cual se quiere detallar en este proyecto y poder representar de forma eficiente un KPI bajo su comunicación (MQTT).

Objetivo General:

- ❖ Establecer la comunicación utilizando el protocolo MQTT entre un programa en PLC M221 de Schneider y un Dashboard desarrollado en Node Red hacia la nube.

Objetivos Específicos:

- ❖ Investigar y explicar sobre el protocolo de comunicación MQTT
- ❖ Desarrollar un programa en PLC M221.
- ❖ Crear una instancia VPS para alojar Node Red.
- ❖ Implementar un Dashboard (KPI) en Node Red, relacionado al programa del PLC.
- ❖ Seleccionar un hosting web en donde se alojará el KPI.

CAPITULO 1: PROTOCOLO MQTT

1.1 PROTOCOLOS DE COMUNICACIÓN

1.1.1 Algunos protocolos de comunicación en la industria:

Un número importante de empresas en nuestro país presentan las llamadas islas automatizadas (equipos o infraestructuras de trabajo sin comunicación entre sí), siendo para estos casos las redes y los protocolos de comunicación Industrial, los cuales son indispensables para realizar un enlace entre las distintas etapas de conforman el proceso.

La irrupción de los microprocesadores en la industria ha posibilitado su integración a redes de comunicación con importantes ventajas, entre las cuales se tiene:

- ❖ Mayor precisión derivada de la integración de tecnología digital en las mediciones.
- ❖ Mayor y mejor disponibilidad de información de los dispositivos de campo.
- ❖ Diagnóstico remoto de componentes.

La integración de las mencionadas islas automatizadas suele hacerse dividiendo las tareas entre grupos de procesos jerárquicamente anidados. Esto da lugar a una estructura de redes industriales, las cuales es posible agrupar en tres categorías:

- ❖ Buses de campo.
- ❖ Redes LAN.
- ❖ Redes LAN-WAN.

Los buses de datos que permiten la integración de equipos para la medición y control de variables de proceso reciben la denominación genérica de buses de campo. Un Bus de campo es un sistema de transmisión de información (datos) que simplifica enormemente la instalación y operación de máquinas y equipamientos industriales utilizados en procesos de producción.

El objetivo de un bus de campo es sustituir las conexiones punto a punto entre los elementos de campo y el equipo de control a través del tradicional lazo de corriente de 4-20mA o 0-10Vdc, según corresponda. Generalmente son redes digitales, bidireccionales, multipunto, montadas sobre un bus serie, que conectan dispositivos de campo como PLCs, transductores, actuadores, sensores y equipos de supervisión. Varios grupos han intentado generar e imponer una norma que permita la integración de equipos de distintos proveedores. Los buses de campo con mayor presencia en el área de control y automatización de procesos son:

1.1.1.1 HART (Highway Addressable Remote Transducer):

Agrupar la información digital sobre la señal analógica típica de 4 a 20mA. La señal digital usa dos frecuencias individuales de 1.200 y 2.200Hz, que representan los dígitos 1 y 0 respectivamente, y que en conjunto forman una onda sinusoidal que se superpone al lazo de corriente de 4-20mA.

1.1.1.2 Profibus DP (Decentralized Periphery):

Orientado a sensores/actuadores enlazados a procesadores (PLCs) o terminales.

1.1.1.3 Profibus PA (Process Automation):

Para control de proceso, cumple normas especiales de seguridad para la industria química.

1.1.1.4 Profibus FMS (Fieldbus Message Specification):

Para comunicación entre células de procesos o equipos de automatización.

1.1.1.5 Foundation Fieldbus (FF):

Es un protocolo de comunicación digital para redes industriales, específicamente utilizado en aplicaciones de control distribuido. Puede comunicar grandes volúmenes de información, ideal para aplicaciones con varios lazos complejos de control de procesos y automatización. Orientado principalmente a la interconexión de dispositivos en industrias de proceso continuo.

1.1.1.6 MODBUS:

Es un protocolo de transmisión para sistemas de control y supervisión de procesos (SCADA) con control centralizado, puede comunicarse con una o varias estaciones remotas (RTU) con la finalidad de obtener datos de campo para la supervisión y control de un proceso. Las interfaces de capa física pueden estar configurada en: RS-232, RS-422, RS-485.

1.2 MQTT

1.2.1 Protocolo de comunicación MQTT:

El protocolo MQTT se ha convertido en uno de los principales pilares del IoT por su sencillez y ligereza. Ambos son condicionantes importantes dado que los dispositivos IoT, a menudo, tienen limitaciones de potencia, consumo y ancho de banda.

1.2.2 ¿Qué es MQTT?

MQTT son las siglas MQ Telemetry Transport, aunque en primer lugar fue conocido como Message Queuing Telemetry Transport. Es un protocolo de comunicación M2M (machine to machine) de to message queue.

Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se reutiliza en cada comunicación. Es una diferencia, por ejemplo, a una petición HTTP 1.0 donde cada transmisión se realiza a través de conexión.

MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera.

Inicialmente fue lanzado en formato propietario, pero en 2010 fue liberado y paso a ser un estándar en 2014 según la OASIS (Organization for the Advancement of Structured Information Standards).

1.2.3 ¿Cómo funciona el MQTT?

MQTT funciona como un servicio de mensajería push con patrón publicador/suscriptor (pub-sub). En este tipo de infraestructuras los clientes se conectan con un servidor central denominado broker.

Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos (Figura 1-1).

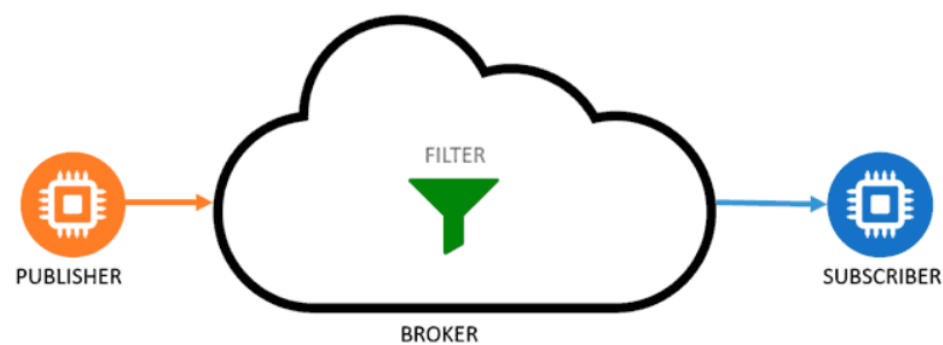


Figura 1-1. Publicador a suscriptor a través de un broker

Los clientes inician una conexión TCP/IP con el broker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id). El broker responde con un mensaje CONNACK (Figura 1-2), que contiene el resultado de la conexión (aceptada, rechazada, etc).

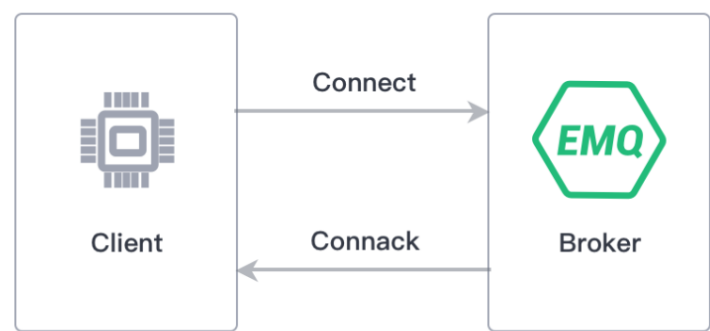


Figura 1-2. Mensaje y respuesta entre cliente y broker

Para enviar los mensajes, el cliente emplea mensajes PUBLISH, que contienen el topic y el payload (Figura 1-3).

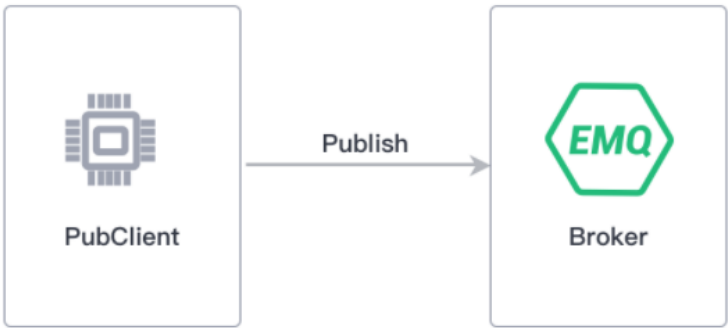


Figura 1-3. Publicación de mensaje hacia el broker

Para suscribirse y desuscribirse se emplean mensajes SUBSCRIBE Y UNSUBSCRIBE, que el servidor responde con SUBACK Y UNSUBACK (Fig 1-4).

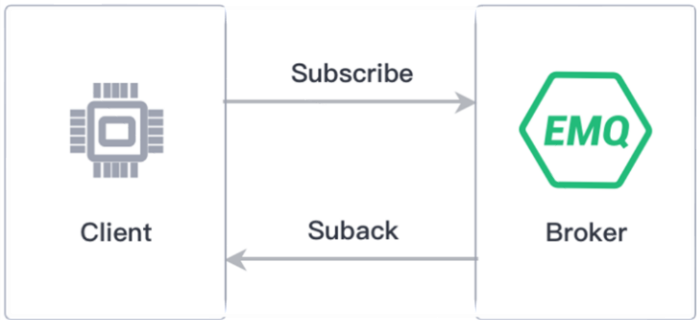


Figura 1-4. Mensaje de suscripción entre cliente y broker

También se tiene que, para asegurar que la conexión esta activa, los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje DISCONNECT.

1.2.4 Estructura de un mensaje MQQT:

Uno de los componentes más importantes del protocolo MQQT es la definición y tipología de los mensajes, ya que son una de las bases de la agilidad en la que radica su fortaleza. Cada mensaje consta de 3 partes (Figura 1-5).

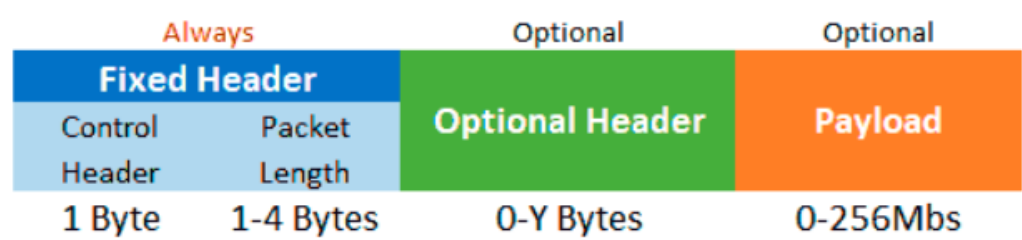


Figura 1-5. Estructura de un mensaje MQTT

- ❖ 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.
- ❖ Cabecera variable: Opcional, contiene información adicional que es necesaria en cierto mensajes o situaciones.
- ❖ Contenido (Payload): Es el contenido real del mensaje. Puede tener un máximo de 256 MB, aunque en implementaciones reales, el máximo es de 2 a 4 kB.

Los tipos de mensajes y códigos de control que se envían en el protocolo MQTT son (Tabla 1-1):

Message	Code
CONNECT	0x10
CONNACK	0x20
PUBLISH	0x30
PUBACK	0x40
PUBREC	0x50
PUBREL	0x60
PUBCOMP	0x70
SUSBCRIBE	0x80
SUBACK	0x90
UNSUBSCRIBE	0xA0
UNSUBACK	0xB0
PINGREQ	0xC=
PINGRESP	0xD0
DISCONNECT	0xE0

Tabla 1-1. Tipos de mensajes y sus códigos del protocolo MQTT

1.2.5 Calidad del servicio:

MQTT dispone de un mecanismo de calidad del servicio o QoS, entendido como la forma de gestionar la robustez del envío de mensajes al cliente ante fallos (por ejemplo, de conectividad).

MQTT tiene tres niveles QoS posibles:

- ❖ QoS 0 unacknowledged (at most one): El mensaje se envía una única vez. En caso de fallo por lo que puede que alguno no se entregue.
- ❖ QoS 1 acknowledged (at least one): El mensaje se envía hasta que se garantiza la entrega. En caso de fallo, el suscriptor puede recibir algún mensaje duplicado.
- ❖ QoS 2 assured (axactly one): Se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Usar un nivel u otro depende de las características y necesidades de fiabilidad de nuestro sistema. Lógicamente, un nivel de QoS superior requiere un mayor intercambio de mansajes verificados con el cliente, y, por tanto, mayor carga al sistema.

1.2.6 Seguridad en MQTT:

La seguridad siempre debe ser un factor importante para considerar en cualquier sistema de comunicación M2M. El protocolo MQTT dispone de distintas medidas de seguridad que podemos adoptar para proteger las comunicaciones.

Esto incluye transporte SSL/TLS y autenticación por usuario y contraseña o mediante certificado. Sin embargo, hay que tener en cuenta que muchos de los dispositivos IoT disponen de escasa capacidad, por lo que el SSL/TLS puede suponer una carga importante.

En muchos casos, la autenticación consiste en una contraseña y usuario que son enviados como texto plano. Por último, también es posible configurar el broker para aceptar conexiones anónimas.

Todo esto debe ser tenido en cuenta a la hora de configurar un sistema MQTT, y entender los riesgos de cada uno de ellos, así como su impacto en la eficiencia del sistema.

1.2.7 Ventajas del MQTT:

Son varias las ventajas que se pueden sacar del protocolo MQTT como protocolo de comunicación M2M. por un lado, tenemos todas las ventajas del patrón pub/sub referidos a la escalabilidad, asincronismo y desacoplamiento entre clientes.

Además, MQTT aporta una serie de características que le han hecho sobresalir por encima de otros competidores. La principal, es su sencillez y ligereza. Esto lo hace adecuado para aplicaciones IoT, donde frecuentemente se utilizan dispositivos de escasa potencia.

Por otro lado, esta menor necesidad de recursos se traduce en un menor gasto de energía, lo cual es muy eficiente para dispositivos que funcionan 24/7 y muy especialmente en dispositivos alimentados por batería.

También para tener en cuenta, sumado a la ligereza del protocolo MQTT es que se necesita de un ancho de banda mínimo, lo cual es importante en redes inalámbricas, o conexiones con posibles problemas de calidad.

Por último, MQTT dispone de medidas adicionales importantes, como la protección y calidad del servicio (QoS). Por todo esto, MQTT aporta robustez y fiabilidad.

1.3 **BROKER**

1.3.1 Definición:

Un Broker es un intermediario, quien se encargará de recibir un mensaje y redistribuirlo. Por tanto, es un patrón de arquitectura que se utiliza para estructurar sistemas de software distribuidos con componentes desacoplados que interactúan por invocaciones de servicios remotos. Esto quiere decir que, si un componente necesita un servicio de otro que está en otra ubicación que no conoce, el broker se encarga de proporcionar la conexión.

Algunas características:

- ❖ Favorece el desacoplamiento de los componentes.
- ❖ Esconde detalles de implementación a los usuarios.
- ❖ Aporta una mayor flexibilidad para añadir, quitar, cambiar o reubicar objetos o funcionalidades.
- ❖ Facilita la comunicación en redes heterogéneas.
- ❖ Reduce la complejidad en el desarrollo de aplicaciones distribuidas.

1.3.2 Estructura:

- ❖ Servidor: Implementan objetos que exponen su funcionalidad a través e interfaces que consisten en operaciones y atributos. Dichas interfaces están disponibles a través de un lenguaje de definición de interfaz (IDL) o un estándar binario.

Se pueden clasificar dos tipos:

- Servidores que ofrecen servicios comunes a muchos dominios de aplicación.
- Servidores específicos para dominios específicos.

- ❖ Cliente: Son aplicaciones que acceden los servicios de al menos un servicio. La interacción entre clientes y servidores se basa en un modelo dinámico, lo cual significa que los servidores también pueden actuar como clientes. Los clientes no necesitan conocer la ubicación de los servidores a los que acceden, esto es importante porque permite la agregación de nuevos servicios y el movimiento de os servicios existentes a otras ubicaciones, aun mientras el sistema este ejecutándose.

- ❖ **Broker:** Es el responsable de la comunicación entre los clientes y servidores. Cuando un cliente necesita un servicio de un servidor, envía una petición al broker, que a su vez se la envía al servidor. El servidor realiza la funcionalidad correspondiente y envía la respuesta (o excepción) de vuelta al broker, que se devuelve al cliente. También ofrece APIs a clientes y servidores que incluyen operaciones para el registro de servidores, y la invocación de métodos de servidores.

Si el servidor especificado es mantenido por otro broker, el broker local encuentra una ruta al broker remoto y le envía la solicitud.

- ❖ **Proxies:** Se pueden distinguir dos tipos de Proxies, los del cliente y del servidor:
 - **Proxies del cliente:** Se ubican entre el cliente y el broker para esconder detalles de implementación como el marshalling (empaquetado de parámetros), la creación y eliminación de bloques de memoria y el mecanismo de comunicación utilizado para enviar el mensaje.
 - **Proxies del servidor:** Son análogos a los proxies del cliente, con el añadido de que reciben solicitudes, las desempaquetan, llaman al servicio y hacen el marshalling de la respuesta o excepción.
- ❖ **Puentes:** Son un componente opcional utilizado para esconder detalles de implementación cuando los brokers interoperan.

1.3.3 Principales brokers:

- ❖ Mosquitto: Es uno de los broker MQTT más conocidos en el sector doméstico. Es un broker Open Source, desarrollado por la fundación Eclipse y distribuido bajo licencia EPL/EDL. Está programado en C, y es multiplataforma. Es un broker liviano y adecuado para uso y servidores de baja potencia.
- ❖ Mosca: Broker MQTT Open Source para Node.js, desarrollado en JavaScript por Matteo Collina. Puede ser empleado como aplicación independiente o embebido en cualquier proyecto de Node.js.
- ❖ Aedes: Del mismo autor que Mosca, Aedes es un servidor MQTT Open Source para Node.js diseñado para ser un reemplazo a Mosca.
- ❖ HBMQTT: Broker MQTT Open Source escrito en Python que funciona sobre asyncio, introducido en Python 3.4.
- ❖ EMQTT: Erlan MQTT es un broker Open Source, desarrollado en Erlang/OTP, está diseñado para aplicaciones con grandes exigencias de escalabilidad.
- ❖ RabbitMQ: Es un popular broker de mensajería AMQP Open Source, que también permite emplear el protocolo MQTT a través de un adaptador.
- ❖ HiveMQ CE: La versión Community del popular HiveMQ es un broker Open Source basado en Java.
- ❖ ActiveMQ: Es un broker de mensajería JMS (Java Message Script) Open Source desarrollado por Apache, que también admite protocolo MQTT.
- ❖ Moquette: Un broker MQTT Open Source escrito en Java desarrollado por Eclipse, que destaca por su ligereza.
- ❖ MQTTnet: Un broker Open Source para .NET.

1.3.3.1 Broker EMQ X:



Figura 1-6. Logo MQTT EMQ X

El broker seleccionado para este proyecto, es el EMQ X (Figura 1-6), fue escogido a base de variadas búsquedas, y se seleccionó por la comodidad al usarlo, instalación intuitiva y por prestar cantidad de servicios útiles de forma gratuita.

EMQ X, es un broker Open Source, desarrollado en Apache, el cual permite conectar millones de dispositivos IoT a través de un clúster EMQ X. También es un broker compatible con todos los clientes MQTT.

Algunas características:

- ❖ MQTT v5.0: integración flexible con todos los clientes MQTT.
- ❖ Alta Simultaneidad: Se puede manejar más de 2 millones de conexiones MQTT simultáneas utilizando un solo nodo de intermediario.
- ❖ Alta disponibilidad: Una arquitectura de agrupación en clúster sin maestro, para lograr una alta disponibilidad y escalabilidad horizontal.
- ❖ Alto rendimiento: Procesa millones de mensajes MQTT por segundo en un grupo de servidores básicos.
- ❖ Baja latencia: Garantiza una latencia de milisegundos en la entrega de mensajes.

Para su instalación primero se realizó la creación de la instancia por medio de GCP, de la cual se detalla en el Capítulo 3, por tanto, teniendo los comandos dados por el fabricante en su web, se realiza la instalación y se abre los puertos de este bajo comandos de consola en Ubuntu (sistema operativo instalado en la instancia), se conceden permisos de super usuario y en orden se ejecutan los comandos de descarga del software específico para la versión de Ubuntu 18.04, seguido por su instalación.

Después de abrir los puertos se ejecuta el software de EMQ X y se puede visualizar en un navegador web, bajo la IP estática de la instancia (Figura 3-38) y el puerto específico de EMQ X el cual es el 18080.

En la Figura 1-7 se puede apreciar lo que sería el Dashboard principal del software EMQ X, en esta imagen específicamente se muestran algunos topics creados en Node-RED, los cuales en su mayoría están vinculados a la dirección de Loopback, bajo la cual está realizada la simulación del programa en el PLC.

EMQ Dashboard

MONITORING

Overview

Connections

Sessions

Topics

Subscriptions

MANAGEMENT

Plugins

Listeners

TOOLS

Topics

Topic	Node
/topic4	emqx@127.0.0.1
/topic5	emqx@127.0.0.1
topic0	emqx@127.0.0.1
/topic3	emqx@127.0.0.1
/topic7	emqx@127.0.0.1
/topic6	emqx@127.0.0.1
/topic1	emqx@127.0.0.1
/topic2	emqx@127.0.0.1

Figura 1-7. Dashboard de EMQ X y los Topics creados

Para el caso de las conexiones establecidas, se tienen dos (Figura 1-8), la que esta como servidor local con la dirección de Loopback (ocupada en la simulación del PLC) y la otra es la que se encuentra en la nube, vinculada a la dirección IP estática de la instancia.

Connections

All Nodes

Clientid

Node	Clientid	Username	IPAddr	Port	CleanStart	ProtoVer	Keepalive(s)	ConnectedAt
emqx@127.0.0.1	mqtt_0ec0065bd05eafc8	undefined	34.134.251.90	35862	true	4	60	2021-08-20 22:19:57
emqx@127.0.0.1	mqtt_e0755019.3966	undefined	190.5.44.65	43698	true	4	60	2021-08-20 22:20:20
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	undefined	127.0.0.1	58024	true	4	60	2021-08-20 22:19:57

Figura 1-8. Conexiones Local y en la Nube

En la Figura 1-9 se representa algunas Subscripciones establecidas mediante la comunicación MQTT en Node-RED, y se pueden ver de forma visual en el Dashboard de EMQ X, tanto así su Nodo, dirección de cliente, el topic y el tipo de seguridad usado en la configuración al momento de realizar el programa en Node-RED.

Subscriptions			
		All Nodes	
Node	ClientId	Topic	QoS
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	/topic7	2
emqx@127.0.0.1	mqtt_e0755019.3966	/topic01	2
emqx@127.0.0.1	mqtt_0ec0065bd05eafc8	/topic1	2
emqx@127.0.0.1	mqtt_0ec0065bd05eafc8	/topic2	2
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	/topic6	2
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	/topic3	2
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	/topic4	2
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	topic0	2
emqx@127.0.0.1	mqtt_155eb5e2db5b0335	/topic5	2

Figura 1-9. Subscripciones realizadas por la comunicación MQTT

En la Figura 1-10 se tienen algunos de los puertos utilizados y que están operando actualmente para realizar la comunicación MQTT, varios de estos puertos tuvieron que ser abiertos tanto en la configuración de Firewall de la instancia en GCP, y en consola de comandos de Ubuntu.

Listeners		
Protocol	Listen On	Max Connections
mqtt:ssl	8883	102400
mqtt:tcp	0.0.0.0:1883	1024000
mqtt:tcp	127.0.0.1:11883	1024000
http:dashboard	18083	512
http:management	8080	512
mqtt:ws	8083	102400
mqtt:wss	8084	16

Figura 1-10. Lista de algunos puertos utilizados para realizar la comunicación

CAPITULO 2: PROGRAMACIÓN EN PLC M221

2.1 PLC M221:



Figura 2-11. PLC M221

2.1.1 Descripción general del PLC TM221C:

Este controlador lógico programable (Figura 2-11) tiene una variedad de potentes funciones y puede servir para una amplia gama de aplicaciones.

2.1.1.1 Lenguajes de Programación:

Este PLC se configura y programa con el software SoMachine Basic, compatible con los siguientes lenguajes:

- ❖ IL: Lista de instrucciones.
- ❖ LD: Diagrama de contactos.
- ❖ Grafcet (lista).
- ❖ Grafcet (SFC).

2.1.2 Características físicas (TM221CE16R):

- ❖ 9 entradas digitales: 4 entradas rápidas (HSC) y 5 entradas normales.
- ❖ 7 salidas digitales: 7 salidas de relé.
- ❖ 2 entradas analógicas.
- ❖ Puertos de comunicación: 1 puerto de línea serie, 1 puerto de programación USB mini-B y 1 puerto Ethernet.

2.2 PROGRAMACIÓN EN M221:

Para la programación del PLC, se ocupa el software EcoStruxure Machine Expert – Basic de Schneider (Figura 2-12), el cual es una herramienta de programación gráfica diseñada para facilitar la configuración, el desarrollo y puesta en marcha de programas para PLCs, este en específico admite el PLC M221 a ocupar en el proyecto, si bien tiene una interfaz bastante básica, es muy intuitiva de utilizar. Para conceptos de visualización a realizar en el KPI (Node-RED), se programó una simulación en el PLC M221 que se seleccionó en la configuración (Figura 2-13), la cual permita obtener la información deseada e interpretar de forma cómoda los datos obtenidos, y así apreciar la correcta utilización del protocolo MQTT.



Figura 2-12. Pantalla de Inicio software EcoStruxure Machine Expert Basic

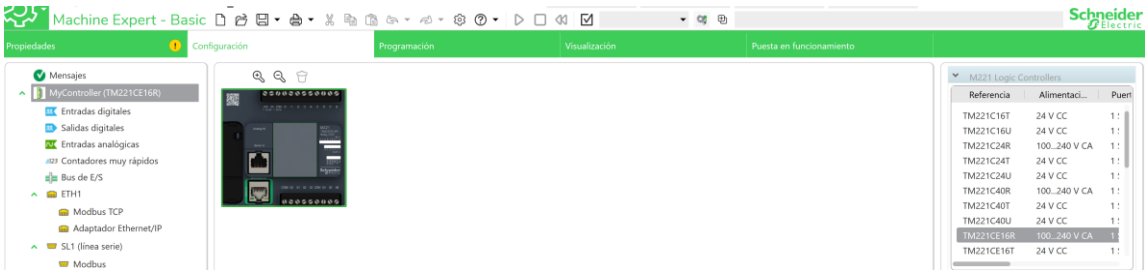


Figura 2-13. Selección del PLC para ocupar en la configuración del programa

Para la comunicación a realizarse entre el PLC y el software de Node-RED, se utiliza el módulo ethernet que trae el PLC, con el protocolo de comunicación Modbus TCP (Figura 2-14), en el caso de la comunicación entre la simulación, por defecto se utilizara la dirección de Loopback propio del sistema (127.0.0.1), esta dirección se configurara en un nodo Modbus de Node-RED (Figura 4-41) para crear el vínculo entre estos dos software, caso contrario, al conectar un PLC físico, en ese nodo se configuraría con la dirección IP que tiene designado el equipo.

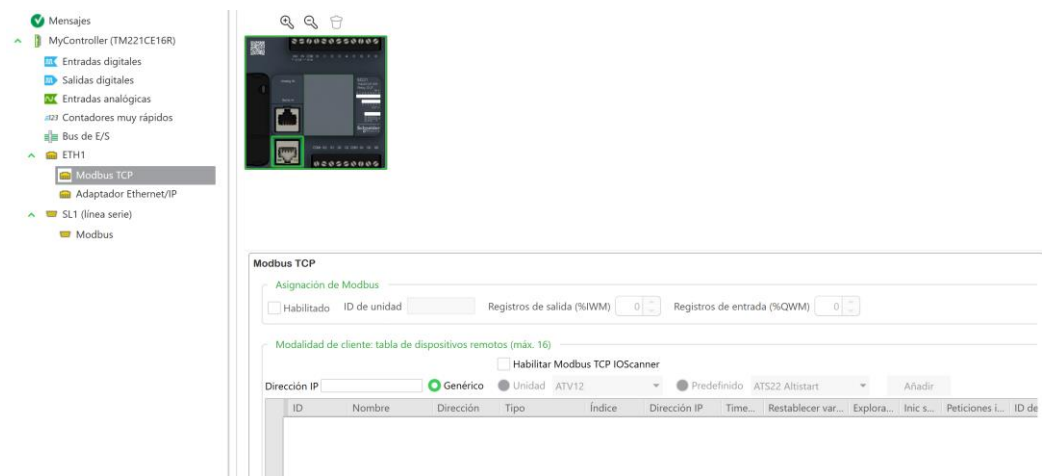


Figura 2-14. Modbus TCP del PLC M221

2.2.1 Bloques ocupados en la programación:

En la tabla 2-2 se representa algunos contactos usados para la programación:

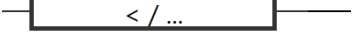
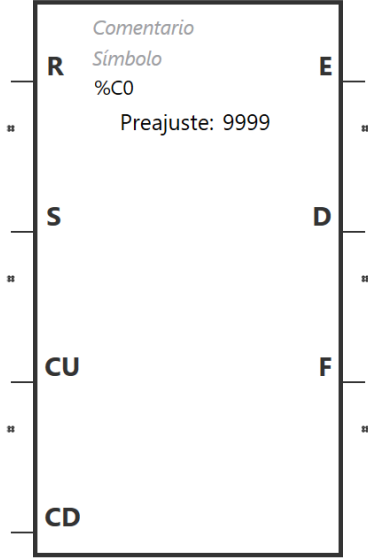
	Nombre	Descripción
Símbolo Dirección 	Contacto NA	Contacto normal abierto, el cual al activarse se cierra.
Símbolo Dirección 	Contacto NC	Contacto normal cerrado, el cual al activarse se abre.
Símbolo Dirección 	Salida	Asignación de salida hacia alguna dirección física o bit interno.
Símbolo Expresión de comparación/operación 	Bloque de comparación	Bloque que almacena valores y compara entre otro dado.
expresion de operacion 	Bloque de operación	Bloque en el que se puede almacenar valores.
Comentario Símbolo %C0 Preajuste: 9999 	Contador	El bloque de funciones counter (contador) cuenta los eventos de forma progresiva y regresiva. Estas 2 operaciones se pueden realizar de manera simultánea. Entradas: R = Restablecer entrada (o instrucción). S = Establecer entrada (o instrucción). CU = Conteo progresivo. CD = Conteo regresivo. Salidas: E = Desborde de conteo regresivo. D = Salida predeterminada alcanzada. F = Desborde de conteo progresivo.

Tabla 2-2. Contactos de programación Ladder

2.2.2 Simulación de estructura Ladder de un registro por pulsos internos:

En la Figura 2-15 se presentan una estructura Ladder la cual consiste en guardar un registro creado a través de pulsos internos cada 1 segundo. Se desarrollo con entradas y salidas de bits internos para el uso correcto de la simulación. Los pulsos son registrados en un bloque de operación y luego comparados en otro bloque colocando un límite de 200 registros, valor en el que se reinicia el registro de pulsaciones.

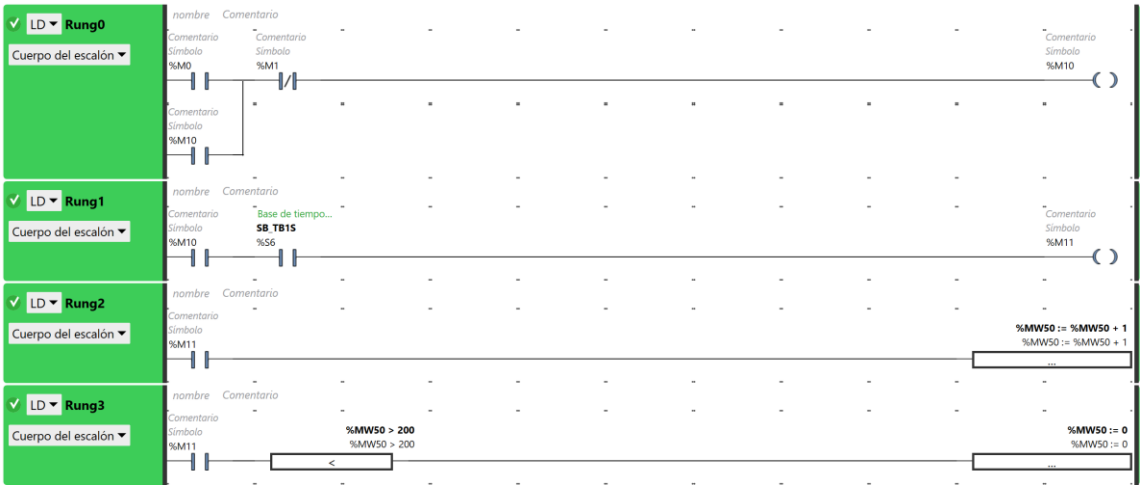


Figura 2-15. Programación Ladder de un registro por pulsos internos.

2.2.3 Simulación de estructura Ladder de un contador y secuencia de LEDs:

En la Figura 2-16 se desarrolló una segunda simulación, la cual consta de un contador, con el que a su vez se realizó una secuencia ON/OFF para 3 salidas.

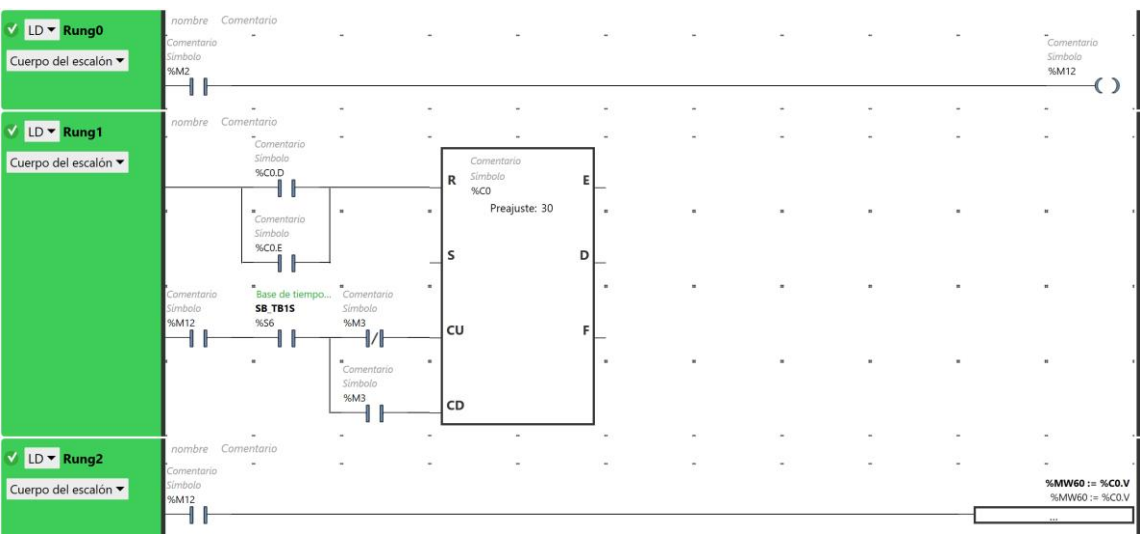


Figura 2-16. Programación Ladder de un contador

En la Figura 2-17 está la parte de la secuencia de salidas ON/OFF, la que visualmente en el KPI se representaría por encendido y apagado de leds, en este caso cada uno se enciende pasado 5 segundos entre cada uno, tiempo dado por el contador y comparado en los bloques.

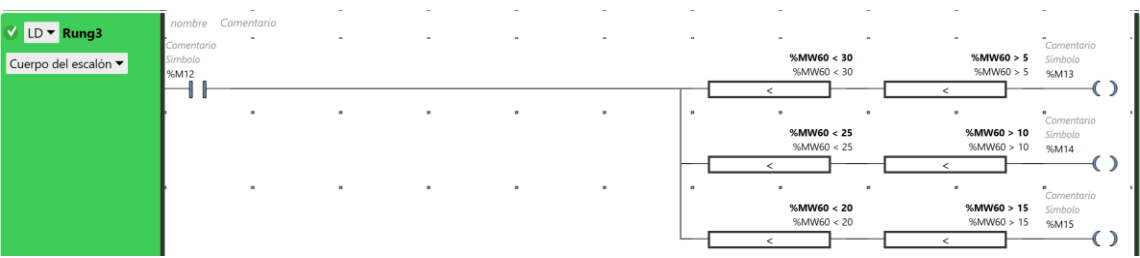


Figura 2-17. Programación Ladder de una secuencia con LEDs

CAPITULO 3: CREACIÓN INSTANCIA GOOGLE CLOUD E INSTALACIÓN
NODE RED

3.1 GOOGLE CLOUD:

3.1.1 ¿Qué es Google Cloud?:

Google Cloud consiste en un conjunto de recursos físicos, como computadoras y unidades de discos duro, y recursos virtuales, como máquinas virtuales (VM), que se encuentran en los centros de datos de Google en todo el mundo. Cada centro de datos está ubicado en una región, específicamente en Asia, Australia, Europa, América del Norte y América del Sur. Cada región es una colección de zonas aisladas entre sí dentro de cada región. Estas zonas se identifican mediante nombres que combinan una letra identificadora con el nombre de la región. Por ejemplo, la zona "a" en la región de Asia oriental se llama "Asia-east1-a".

Esta distribución de los recursos brinda varios beneficios, que incluyen redundancia en casos de fallas y menor latencia, ya que los recursos se encuentran más cerca de los clientes. La distribución también presenta algunas reglas sobre cómo se pueden usar los recursos en conjunto.

3.1.2 Instancia VM:

3.1.2.1 ¿Qué es una Instancia:

Una instancia es una máquina virtual (VM) alojada en la infraestructura de Google. Para utilizarla se debe crear en Google Cloud Console, la herramienta de comandos de "gcloud" o la API de Compute Engine.

Las instancias de Compute Engine pueden ejecutar las imágenes públicas de Linux y Windows Server que proporciona Google, así como las imágenes personalizadas privadas que puedes crear o importar desde algún sistema existente. Si se usa un conjunto de tipos predefinidos de máquinas o se crea un propio tipo personalizado de máquinas, se puede elegir propiedades de máquinas de las instancias, como la cantidad de CPU virtuales y de memoria.

3.1.2.2 Instancias y Proyectos:

Cada instancia pertenece a un proyecto de Google Cloud Console y cada proyecto puede tener una o más instancias. Cuando se crea una instancia en un proyecto, se especifica la zona, el sistema operativo y el tipo de máquina de esa instancia. Cuando se borra la instancia, se quita del proyecto.

3.1.2.3 Instancias y opciones de almacenamiento:

De forma predeterminada, cada instancia de Compute Engine tiene un pequeño disco de arranque persistente que contiene el sistema operativo. Cuando las aplicaciones que se ejecutan en la instancia requieren más espacio de almacenamiento, se puede agregar opciones de almacenamiento adicionales a la instancia.

3.1.2.4 Instancias y Redes:

Cada interfaz de red de una instancia de Compute Engine está asociada con una subred de una red VPC única.

3.1.2.5 Instancias y contenedores:

Las instancias de Compute Engine admiten un método declarativo para iniciar aplicaciones con contenedores. Cuando se crea una VM o una plantilla de instancia, se puede proporcionar un nombre de imagen de Docker y, luego iniciar la configuración. Compute Engine se encargará del resto, incluido el suministro de una imagen actualizada de Container-Optimized OS con Docker instalado y el lanzamiento del contenedor cuando se inicia la VM.

3.1.3 Creación de Instancia:

El primer paso para la realización y creación de la instancia es dirigirse al sitio web de Google Cloud (Figura 4-18), registrarnos gratuitamente con una cuenta Gmail, la cual nos dará un bono de 300\$ dólares para ocupar en la creación de instancias y otras herramientas que se utilicen, si se llega a acceder a más del presupuesto dado, Google alertara de que se adjuntara un cargo a la tarjeta vinculada cuando se creó la cuenta.

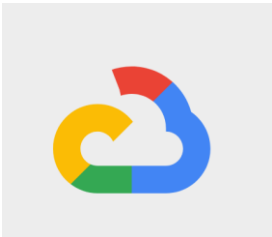


Figura 2-18. Logotipo GCP

En el buscador dentro de Google Cloud, se digita la frase Compute Engine (Figura 3-19) y se dirige a la interfaz en donde se procederá a la creación de la instancia (Figura 3-20).

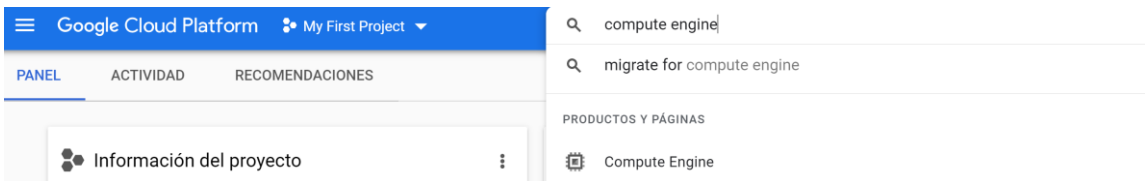


Figura 3-19. Localización de la herramienta Compute Engine

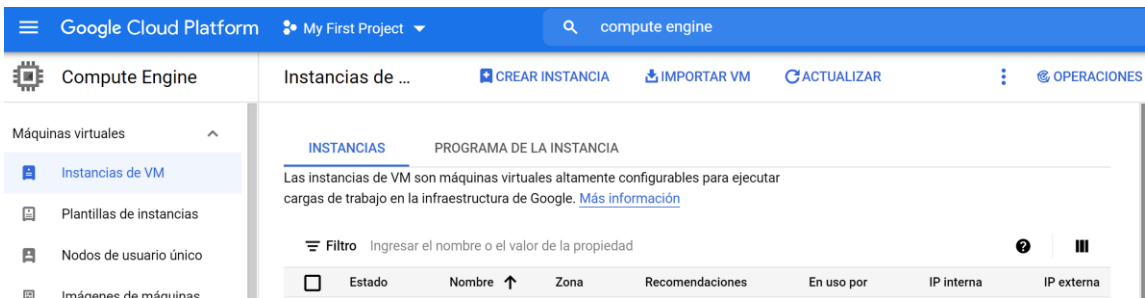


Figura 3-20. Panel de Compute Engine

En la Figura 3-21 se ve la configuración a realizar para la creación de la interfaz en donde se alojará Node RED.

En la configuración de la máquina se deja por defecto, que cuenta con los requisitos mínimos para la creación de la instancia y realizar las operaciones dentro de ella, que no son tan pesadas.

Configuración de la máquina

Familia de máquinas

Uso general

Optimizada para procesamiento

Memoria optimizada

GPU

Tipos de máquinas para cargas de trabajo comunes, optimizados en función del costo y la flexibilidad

Series

E2

Selección de la plataforma de CPU según la disponibilidad

Tipo de máquina

e2-medium (2 CPU virtuales, 4 GB de memoria)

vCPU

1 núcleo compartido

Memoria

4 GB

GPU

-

Plataforma de CPU y GPU

Servicio de VM confidencial

☐

Habilita el servicio de procesamiento confidencial en esta instancia de VM.

Contenedor

☐

Implementa una imagen de contenedor en esta instancia de VM. [Más información](#)

Disco de arranque

Nuevo disco persistente equilibrado de 10 GB

Imagen

Ubuntu 18.04 LTS

Cambiar

Identidad y acceso a la API

Cuenta de servicio

Compute Engine default service account

Permiso de acceso

☒ Permitir el acceso predeterminado

☐ Permitir el acceso total a todas las API de Cloud

☐ Configurar el acceso para cada API

Figura 3-21. Configuración para la creación de instancia

En el disco de arranque, por comodidad de la instalación de Node.js y el alcance de comandos para la consola, se utilizara el sistema operativo Ubuntu, que es una distribución GNU/Linux, en este caso específico la versión 18.04 LTS (Figura 3-22), se selecciona el tipo de disco a ocupar y el tamaño, con el más básico es suficiente, como se dijo, la operación a utilizar es liviana, por tanto no se necesita de tantos recursos y obviando que ocupar recursos más altos aumenta el costo asociado al cobro que realiza Google Cloud.

Disco de arranque

Selecciona una imagen o instantánea para crear un disco de arranque o adjuntar un disco existente. en [Marketplace](#).

Imágenes públicas

Imágenes personalizadas

Instantáneas

Discos existentes

Sistema operativo

Ubuntu

Versión

Ubuntu 18.04 LTS

amd64 bionic image built on 2021-07-20, supports Shielded VM features ?

Tipo de disco de arranque ?

Tamaño (GB) ?

Disco persistente equilibrado

10

Figura 3-22. Sistema operativo montado en la instancia

Para finalizar, se da los permisos de Firewall (Figura 3-23) correspondiente y se da a crear.

Firewall ?

Agregar etiquetas y reglas de firewall para permitir un tráfico de red determinado desde Internet

☒ Permitir tráfico HTTP

☒ Permitir tráfico HTTPS

⌵

Administración, seguridad, discos, redes, usuario único

Se aplicarán cargos por esta instancia. [Precios de Compute Engine](#) ↗

Crear

Cancelar

Figura 3-23. Permisos de Firewall

Luego, como configuración adicional, se dará una IP estática a la instancia, que por defecto viene de forma dinámica, esta configuración, más que nada es para dar una comodidad a la hora de realizar el lanzamiento de Node RED, así, siempre ocuparemos una misma dirección IP (Fig 3-24). Para esto, en el buscador se digita la frase Red VPC, y en la sección de Direcciones Externas, se modifica el tipo de Red de la instancia recién creada.

Adjuntar dirección IP

Dirección IP

node-red01 (35.222.101.19)

Adjuntar a

node-red01 (zona us-central1-a)  

Es posible que algunas de las instancias estén inhabilitadas debido a la política de la organización "IP externas para las instancias de VM". [Más información](#)

CANCELAR

ACEPTAR

Figura 3-24. Asignación de IP estática para la instancia

Por último, se realizará la configuración de abrir un puerto TCP, específicamente el 1880 para poder ocupar Node RED sin problema, ya que este es el puerto base que ocupa para funcionar.

Se le asigna un nombre, en este caso "allow-tcp-in-1880", además de agregar un filtro de origen como "Rango de IP", y se especifica un rango de origen "0.0.0.0/0", lo demás se deja por defecto (Fig 3-25).

allow-tcp-in-1880

Descripción

Registros

Activar los registros de firewall puede generar una gran cantidad de registros y aumentar los costos en Cloud Logging.[Más información](#)

Activado

Desactivado

Red

default

Prioridad *

100

VERIFICAR LA PRIORIDAD DE OTRAS REGLAS DE FIREWALL

?

La prioridad puede ser de 0 a 65535

Dirección

Entrada

Acción en caso de coincidencia

Permitir

Destinos

Todas las instancias de la red

Filtro de origen

Rangos de IP

?

Rangos de IP de origen *

0.0.0.0/0

✕

por ejemplo, 0.0.0.0/0, 192.168.2.0/24

?

Figura 3-25. Configuración para abrir puerto TCP 1880

3.2 NODE RED:

3.2.1 ¿Qué es Node RED?:

Node-RED es una herramienta de programación para conectar dispositivos de hardware, API y servicios en línea de formas nuevas e interesantes.

Proporciona un editor basado en navegador que facilita la interconexión de flujos utilizando la amplia gama de nodos de la paleta que se pueden implementar en su tiempo de ejecución con un solo clic.

3.2.1.1 Edición de flujo basada en navegador:

Node-RED proporciona un editor de flujo basado en navegador, que facilita la conexión de flujos utilizando una amplia gama de nodos de la paleta. Luego, los flujos se pueden implementar en el tiempo de ejecución con un solo clic.

Las funciones de JavaScript se pueden crear dentro del editor usando un editor de texto enriquecido.

Una biblioteca incorporada le permite guardar funciones, plantillas o flujos útiles para su reutilización.

3.2.1.2 Estructura base:

Está basado en una construcción de tiempo de ejecución liviano como Node.js, aprovechando al máximo su modelo sin bloqueo impulsado por eventos. Esto lo hace ideal para ejecutarse en redes de hardware de bajos recursos o de bajo costo como Raspberry Pi, así como en la nube.

Con más de 225.000 módulos en el repositorio de paquetes de Node, es fácil ampliar la gama de nodos de paleta para agregar nuevas capacidades.

3.2.2 Instalación de Node.js y Node RED en la instancia creada:

Primero, para poder ocupar Node RED en la instancia, se debe instalar a través de consola los archivos necesarios y bases para la utilización de este, por tanto, para esto, como primer paso, es abrir la instancia ya creada, seleccionando SSH en el apartado “Conectar” de la instancia (Figura 3-26).

Nombre ↑	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
node-red	us-central1-a	 Ahorrar \$14/mes		10.128.0.5 (nic0)	34.134.251.90 	SSH ▼

Figura 3-26. Abriendo consola mediante la opción SSH

Hecho esto, se abrirá la instancia en una nueva ventana, con la consola de comandos abierta, por tanto, ya se está dentro del entorno del sistema operativo de Ubuntu (Figura 3-27).

```
ricardo_varela_2603@nodered: ~ - Google Chrome
ssh.cloud.google.com/projects/dynamic-digit-320319/zones/us-central1-a/instances/nodered?authu
Connected, host fingerprint: ssh-rsa 0 26:2C:6E:2A:3A:47:75:07:61:16
:1B:DA:C1:F1:8C:5C:05:D8:36:D1:6D:0B:93:63:DF:A3:ED:83
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1049-gcp x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

System information as of Mon Aug 23 22:34:28 UTC 2021

System load:  0.32               Processes:            114
Usage of /:   34.0% of 9.52GB    Users logged in:     0
Memory usage: 11%               IP address for ens4: 10.128.0.7
Swap usage:   0%

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Thu Aug 12 20:50:35 2021 from 35.235.240.1
ricardo_varela_2603@nodered:~$
```

Figura 3-27. Consola de comando de Ubuntu dentro de la instancia

Se procede a utilizar el siguiente comando:

❖ “sudo apt-get install build-essential”

```
ricardo_varela_2603@nodered:~$ sudo apt-get install build-essential
```

Figura 3-28. Paquetes esenciales de compilación

El Comando de la Figura 3-28 es para instalar los paquetes necesarios para compilar software, incluye el depurador GNU, la colección de compiladores g++/GNU y algunas otras herramientas y bibliotecas necesarias para compilar un programa.

Se da los permisos para continuar con la instalación del paquete anterior, el cual ocupara 165 MB dentro de la instancia (Figura 3-29).

```
ricardo_varela_2603@node-red1:~$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libnumal
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-7 dpkg-dev fakeroot g++ g++-7
  gcc gcc-7 gcc-7-base libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0
  libcilkrts5 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-7-dev libgomp1 libisl19
  libitm1 liblsan0 libmpc3 libmpx2 libquadmath0 libstdc++-7-dev libtsan0 libubsan0
  linux-libc-dev make manpages-dev
Suggested packages:
  binutils-doc cpp-doc gcc-7-locales debian-keyring g++-multilib g++-7-multilib gcc-7-doc
  libstdc++6-7-dbg gcc-multilib autoconf automake libtool flex bison gdb gcc-doc
  gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan4-dbg
  liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg
  glibc-doc bzip libstdc++-7-doc make-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-7 dpkg-dev
  fakeroot g++ g++-7 gcc gcc-7 gcc-7-base libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0
  libcilkrts5 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-7-dev libgomp1 libisl19
  libitm1 liblsan0 libmpc3 libmpx2 libquadmath0 libstdc++-7-dev libtsan0 libubsan0
  linux-libc-dev make manpages-dev
0 upgraded, 40 newly installed, 0 to remove and 0 not upgraded.
Need to get 43.0 MB of archives.
After this operation, 165 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figura 3-29. Dar acceso a ocupar espacio de memoria para la instalación del paquete

Paso siguiente es la utilización del comando:

❖ “curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -”

```
ricardo_varela_2603@node-red1:~$ curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
```

Figura 3-30. Comando de descarga para el software Node.js

Este comando (Figura 3-30) es el encargado de descargar la versión solicitada de Node.js, en este caso la versión 14. Node.js es un entorno de tiempo de ejecución e JavaScript. Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript, como lo es Node-RED.

En la Figura 3-31 se muestra la descarga correcta de Node.js, lista para su instalación.

```
ricardo_varela_2603@node-red1: ~ - Google Chrome
ssh.cloud.google.com/projects/dynamic-digit-320319/zones/us-central1-a/instances/node-red1?authuser=0&hl=es_419&projectNumber=...
+ curl -sLf -o /dev/null 'https://deb.nodesource.com/node_14.x/dists/bionic/Release'

## Adding the NodeSource signing key to your keyring...

+ curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key | gpg --dearmor | tee /usr/share
/keyrings/nodesource.gpg >/dev/null
gpg: WARNING: unsafe ownership on homedir '/home/ricardo_varela_2603/.gnupg'

## Creating apt sources list file for the NodeSource Node.js 14.x repo...

+ echo 'deb [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_14.x
bionic main' > /etc/apt/sources.list.d/nodesource.list
+ echo 'deb-src [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_
14.x bionic main' >> /etc/apt/sources.list.d/nodesource.list

## Running `apt-get update` for you...

+ apt-get update
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:5 https://deb.nodesource.com/node_14.x bionic InRelease [4584 B]
Get:6 https://deb.nodesource.com/node_14.x bionic/main amd64 Packages [763 B]
Fetched 5347 B in 0s (11.2 kB/s)
Reading package lists... Done

## Run `sudo apt-get install -y nodejs` to install Node.js 14.x and npm
## You may also need development tools to build native addons:
    sudo apt-get install gcc g++ make
## To install the Yarn package manager, run:
    curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | gpg --dearmor | sudo tee /usr/share/ke
yrings/yarnkey.gpg >/dev/null
    echo "deb [signed-by=/usr/share/keyrings/yarnkey.gpg] https://dl.yarnpkg.com/debian stable
main" | sudo tee /etc/apt/sources.list.d/yarn.list
    sudo apt-get update && sudo apt-get install yarn

ricardo_varela_2603@node-red1:~$
```

Figura 3-31. Lista de operaciones con descarga completada de Node.js

Siguiendo con los pasos, se digita el comando (Figura 3-32):

❖ “sudo apt-get install -y nodejs”

```
ricardo_varela_2603@node-red1:~$ sudo apt-get install -y nodejs
```

Figura 3-32. Comando de instalación de Node.js

Ya con esto, se tiene instalado la versión de Node.js descargado, todo esto mediante consola de comandos por la interfaz de Ubuntu de forma intuitiva y al alcance, ya que los comandos se suelen encontrar en las mismas páginas del software especificado.

```
ricardo_varela_2603@node-red1: ~ - Google Chrome
ssh.cloud.google.com/projects/dynamic-digit-320319/zones/us-central1-a/instances/node-red1?authuser=0&hl=es_419&proj
Fetched 29.0 MB in 32s (916 kB/s)
Selecting previously unselected package libpython2.7-minimal:amd64.
(Reading database ... 71098 files and directories currently installed.)
Preparing to unpack .../0-libpython2.7-minimal_2.7.17-1~18.04ubuntu1.6_amd64.deb ...
Unpacking libpython2.7-minimal:amd64 (2.7.17-1~18.04ubuntu1.6) ...
Selecting previously unselected package python2.7-minimal.
Preparing to unpack .../1-python2.7-minimal_2.7.17-1~18.04ubuntu1.6_amd64.deb ...
Unpacking python2.7-minimal (2.7.17-1~18.04ubuntu1.6) ...
Selecting previously unselected package python-minimal.
Preparing to unpack .../2-python-minimal_2.7.15~rc1-1_amd64.deb ...
Unpacking python-minimal (2.7.15~rc1-1) ...
Selecting previously unselected package libpython2.7-stdlib:amd64.
Preparing to unpack .../3-libpython2.7-stdlib_2.7.17-1~18.04ubuntu1.6_amd64.deb ...
Unpacking libpython2.7-stdlib:amd64 (2.7.17-1~18.04ubuntu1.6) ...
Selecting previously unselected package python2.7.
Preparing to unpack .../4-python2.7_2.7.17-1~18.04ubuntu1.6_amd64.deb ...
Unpacking python2.7 (2.7.17-1~18.04ubuntu1.6) ...
Selecting previously unselected package libpython-stdlib:amd64.
Preparing to unpack .../5-libpython-stdlib_2.7.15~rc1-1_amd64.deb ...
Unpacking libpython-stdlib:amd64 (2.7.15~rc1-1) ...
Setting up libpython2.7-minimal:amd64 (2.7.17-1~18.04ubuntu1.6) ...
Setting up python2.7-minimal (2.7.17-1~18.04ubuntu1.6) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up python-minimal (2.7.15~rc1-1) ...
Selecting previously unselected package python.
(Reading database ... 71846 files and directories currently installed.)
Preparing to unpack .../python_2.7.15~rc1-1_amd64.deb ...
Unpacking python (2.7.15~rc1-1) ...
Selecting previously unselected package nodejs.
Preparing to unpack .../nodejs_14.17.5-1nodesource1_amd64.deb ...
Unpacking nodejs (14.17.5-1nodesource1) ...
Setting up nodejs (14.17.5-1nodesource1) ...
Setting up libpython2.7-stdlib:amd64 (2.7.17-1~18.04ubuntu1.6) ...
Setting up python2.7 (2.7.17-1~18.04ubuntu1.6) ...
Setting up libpython-stdlib:amd64 (2.7.15~rc1-1) ...
Setting up python (2.7.15~rc1-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
ricardo_varela_2603@node-red1:~$
```

Figura 3-33. Instalación correcta de Node.js

Realizado estos comandos, se completó la instalación del programa Node.js (Figura 3-33), para posterior instalación de Node-RED.

Comando para instalación (Figura 3-34) de Node-RED:

❖ “sudo npm install -g -unsafe-perm node-red”

```
ricardo_varela_2603@node-red1:~$ sudo npm install -g -unsafe-perm node-red
```

Figura 3-34. Comando de instalación Node-RED

Y con el presente comando, se ha realizado la instalación de Node RED (Figura 3-35) y queda lista para lanzarse sobre la instancia creada en la plataforma de Google Cloud.

```
ricardo_varela_2603@node-red1:~$ sudo npm install -g -unsafe-perm node-red
/usr/bin/node-red -> /usr/lib/node_modules/node-red/red.js
/usr/bin/node-red-pi -> /usr/lib/node_modules/node-red/bin/node-red-pi

> bcrypt@5.0.1 install /usr/lib/node_modules/node-red/node_modules/bcrypt
> node-pre-gyp install --fallback-to-build

[bcrypt] Success: "/usr/lib/node_modules/node-red/node_modules/bcrypt/lib/binding/napi-v3/bcrypt
t_lib.node" is installed via remote
+ node-red@2.0.5
added 290 packages from 370 contributors in 19.316s
ricardo_varela_2603@node-red1:~$ █
```

Figura 3-35. Node-RED instalado correctamente

Para lanzar la aplicación web de Node-RED (Figura 3-36), se utiliza el comando:

❖ “node-red”

```
ricardo_varela_2603@node-red1:~$ node-red
```

Figura 3-36. Comando para inicializar Node-RED

```
ricardo_varela_2603@node-red1:~$ node-red
23 Aug 23:21:07 - [info]

Welcome to Node-RED
=====

23 Aug 23:21:07 - [info] Node-RED version: v2.0.5
23 Aug 23:21:07 - [info] Node.js version: v14.17.5
23 Aug 23:21:07 - [info] Linux 5.4.0-1051-gcp x64 LE
23 Aug 23:21:07 - [info] Loading palette nodes
23 Aug 23:21:08 - [info] Settings file : /home/ricardo_varela_2603/.node-red/settings.js
23 Aug 23:21:08 - [info] Context store : 'default' [module=memory]
23 Aug 23:21:08 - [info] User directory : /home/ricardo_varela_2603/.node-red
23 Aug 23:21:08 - [warn] Projects disabled : editorTheme.projects.enabled=false
23 Aug 23:21:08 - [info] Flows file : /home/ricardo_varela_2603/.node-red/flows.json
23 Aug 23:21:08 - [info] Creating new flow file
23 Aug 23:21:08 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

23 Aug 23:21:08 - [info] Server now running at http://127.0.0.1:1880/
23 Aug 23:21:08 - [info] Starting flows
23 Aug 23:21:08 - [info] Started flows
```

Figura 3-37. Node-RED Inicializado

Ya inicializado Node-RED (Figura 3-37), con la dirección IP externa (estática) que nos proporciona Google para la instancia (Figura 3-38), configurada previamente, es con la cual está corriendo node-red, por tanto, dicha dirección IP se coloca en un navegador web para así visualizar el entorno gráfico y de programación de Node-RED.

Dirección IP y puerto:34.134.251.90/1880

Nombre ↑	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
node-red	us-central1-a	 Ahorrar \$14/mes		10.128.0.5 (nic0)	34.134.251.90 	SSH 

Figura 3-38. Dirección estática de la instancia en donde se encuentra Node-RED

En la Figura 3-39 se aprecia el entorno gráfico y de programación de node-RED en el navegador.

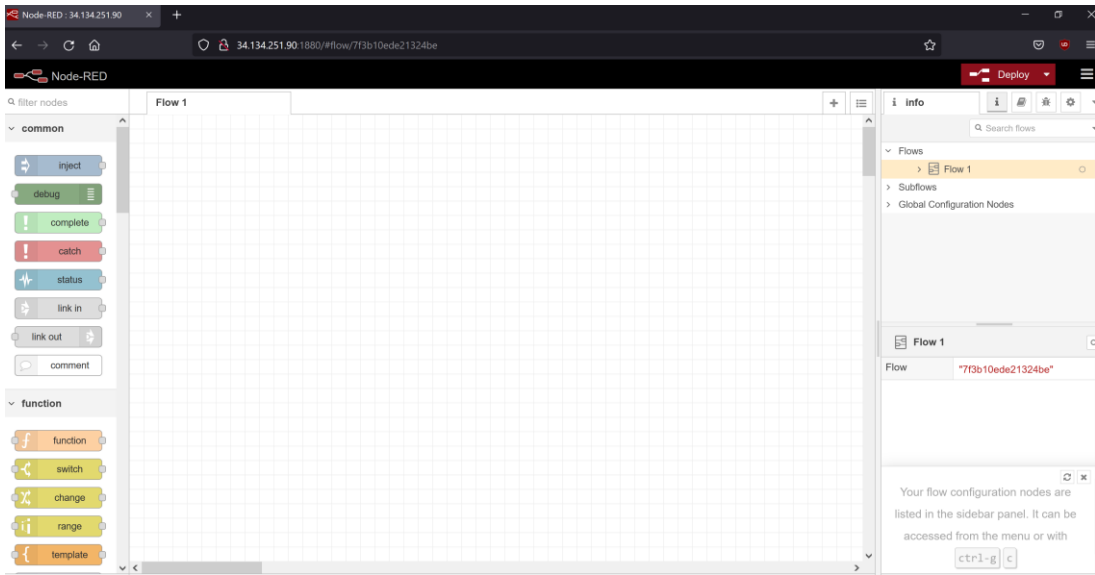


Figura 3-39. Entorno de programación de Node-RED

CAPITULO 4: DASHBOARD (KPI)

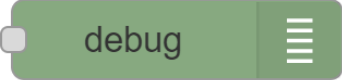
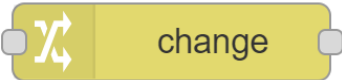
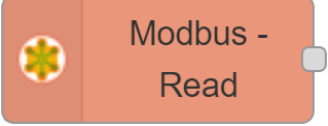
4.1 **INTERFAZ Y HERRAMIENTAS DE NODE-RED:**

4.1.1 Nodos:

La estructura mínima son los nodos. Estos se arrastran a través de la interfaz gráfica y nos permiten hacer una tarea concreta. Recibir una llamada HTTP, un mensaje MQTT o hasta lo más básico como la activación de un pulsador.

Todos estos nodos se organizan en flujos o flows, que agrupan nodos que se conectan entre ellos. Todo de una forma visual, sin apenas tener que programar.

4.1.1.1 Algunos Nodos Ocupados en Tabla 4-3:

	Nombre	Descripción
 comment	Comment	Un nodo que se puede utilizar para agregar comentarios a los flujos creados.
 debug	Debug	Muestra las propiedades del mensaje seleccionado, en la pestaña lateral de depuración.
 inject	Inject	Inyecta un mensaje en un flujo de forma manual o a intervalos regulares, este mensaje puede ser de varios tipos como: string, array, boolean, etc.
 change	Change	Establece, cambia, elimina o mueve propiedades de un mensaje.
 trigger	Trigger	Cuando se activa, puede enviar un mensaje y, opcionalmente un segundo mensaje.
 mqtt in	Mqtt in	Se conecta a un agente (broker) MQTT y se suscribe a mensajes del tema (tópicos) especificados.
 mqtt out	Mqtt out	Se conecta a un agente (broker) MQTT y publica mensajes.
 Modbus - Read	Modbus - Read	Se conecta a un Modbus TCP o serial para leer valores de registros/bobinas con una tasa de sondeo determinada.

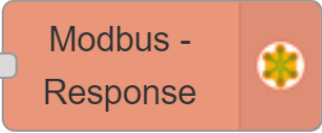
	Modbus - Response	Nodo que sirve para mostrar la longitud o respuesta de estados de una segunda salida de los Nodos Modbus Read/Write/Gett.
	json	Convierte entre una cadena Json y su representación de objeto JavaScript, en cualquier dirección.

Tabla 4-3. Nodos Ocupados para la programación de Node-RED

4.2 **FLUJO CREADO EN NODE-RED:**

4.2.1 Comunicación Local entre Node-RED y la simulación del PLC:

Como se mencionó en el Capítulo 2, la comunicación a realizarse entre el PLC (simulación/físico) y Node-RED (Local) se realizará mediante Ethernet, específicamente el protocolo de comunicación Modbus TCP, en la Figura 4-40 se aprecia el Nodo a utilizarse para la lectura de datos, los cuales son sacados de los Tags asignados a los contactos de la programación Ladder. Este Nodo en específico, no viene por defecto en los Nodos principales que trae Node-RED, por tanto, se instala como un repositorio externo.

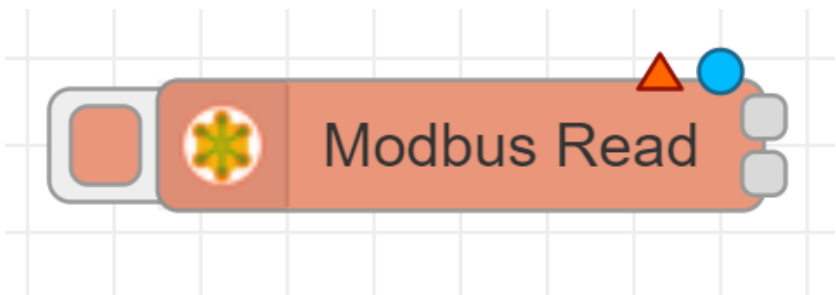


Figura 4-40. Nodo Modbus Read en Node-RED

Luego de seleccionar dicho Nodo Modbus, en este caso de lectura, haciendo doble click se accede a su configuración (Figura 4-41), donde se le asigna un nombre, el tipo de comunicación en este caso TCP, el Host que sería la dirección IP asignada al equipo PLC, en el caso de la simulación se utiliza la dirección de Loopback, además

de dejar el puerto por defecto 502. Con esto estaría enlazada la comunicación entre Node-RED (Local) y el equipo PLC por medio de Modbus TCP.

Edit Modbus-Read node > Edit modbus-client node

Delete

Cancel

Update

⚙ Properties

Name

M221

Type

TCP

Host

127.0.0.1

Port

502

TCP Type

DEFAULT

Unit-Id

1

Timeout (ms)

1000

Reconnect on timeout

☒

Reconnect timeout (ms)

2000

UnitId's in parallel

☒

Figura 4-41. Configuración Modbus TCP

Terminada la configuración del Nodo Modbus TCP, la siguiente opción es para darle la dirección exacta al tags que se quiere leer (Figura 4-42), en el caso presentado se tiene para una dirección de bit interno de Partida, como es una sola unidad se asigna 1, y FC 2 que sería la lectura estados de entradas, en Address indicamos el valor 10, ya que este es el bit interno declarado en la programación para el contacto de partida y en quantity el valor 1, para que lea solo un contacto desde la dirección 10.

Edit Modbus-Read node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🖨️

Settings

Optionals

Name

Partir

Topic

Topic

Unit-Id

1

FC

FC 2: Read Input Status

Address

10

Quantity

1

Poll Rate

100

millisecond(s)

⏻ Delay on start

☐

Server

M221

✎

Figura 4-42. Configuración Partir

4.2.2 Local (Simulación):

En la figura 4-43 se presenta el flujo creado en base de algunos Nodos Modbus, algunos Nodos de visualización como el led y el gauge, y más importante los Nodos MQTT vinculados a los Topics establecidos los cuales tendrán comunicación entre sí por parte del Node-RED Local y el que se encuentra en la Nube (Instancia).

Por tanto se tiene una lectura en el Nodo Modbus con el nombre Partir, el cual está vinculado a la dirección de tags 10 de la programación en el Ladder, esta lectura se puede visualizar en formato Json con el Modbus Response, solo para ver la lectura interna de este y si está funcionando correctamente, también se puede ver de forma visual con un LED de encendido o apagado para el Dashboard, principalmente este no es vital en esta parte ya que el que se verá realmente será el Dashboard generado en la instancia, y por último se tiene el Nodo MQTT, este se configura primero una sola vez para poder vincularlo al Broker EMQ X que se instaló en la instancia, para lo cual dando a las configuraciones del Nodo MQTT (Figura 4-44) se asigna un servidor el cual es donde está alojado el Broker, por tanto se declara el servidor 34.134.251.90, que es la dirección IP estática de la instancia y el puerto que se utiliza para la comunicación que sería la 1883.

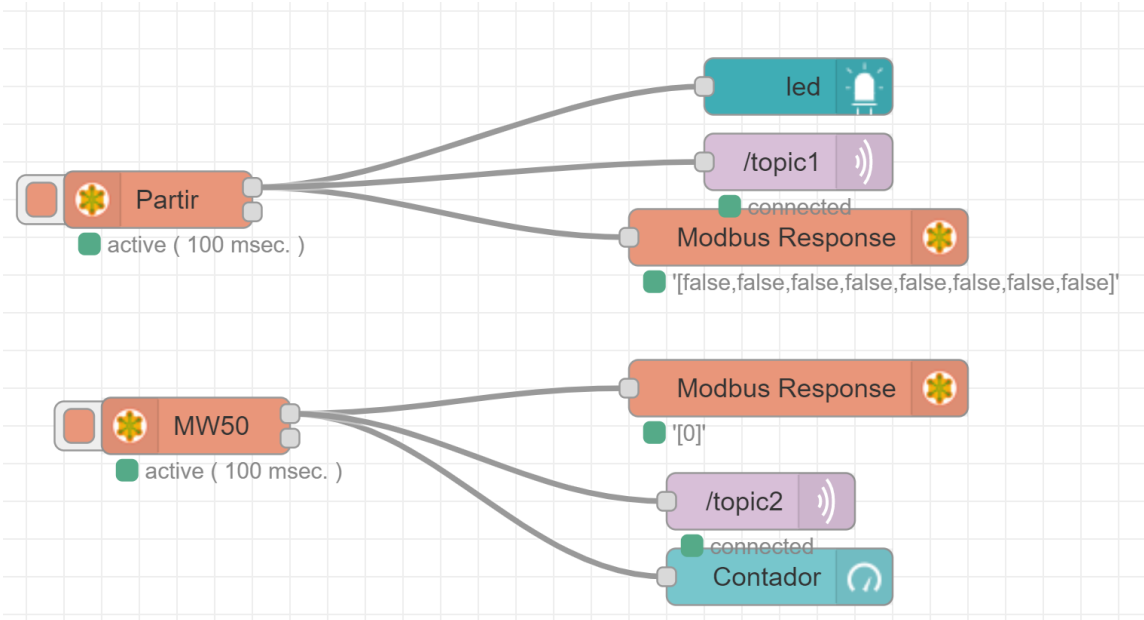


Figura 4-43. Flujo creado de lectura Modbus de partida y registro

Edit mqtt out node > Edit mqtt-broker node

Delete
Cancel
Update

Properties

Name
MQTT Broker

Connection
Security
Messages

Server
http://34.134.251.90/
Port
1883

☐ Use TLS

Protocol
MQTT V3.1.1

Client ID
Leave blank for auto generated

Keep Alive
60

Session
☒ Use clean session

Figura 4-44. Configuración del servidor MQTT en el Nodo

Terminada dicha configuración del servidor MQTT, para el presente nodo se le asigna un nombre como topic el cual es /topic1 (Figura 4-45), el cual debe ser idéntico al que se asignara en el nodo que se encontrara en la nube, para recibir la lectura que envía el presente nodo, realizando así los mensajes de suscriptor y publicador entre cliente y el broker.

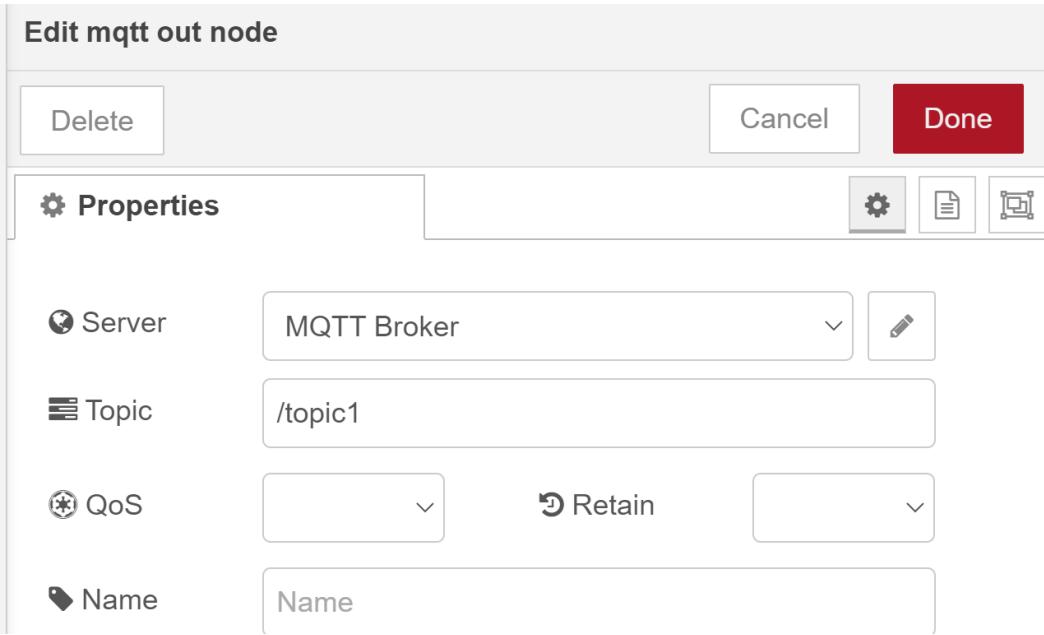


Figura 4-45. Tópico declarado para la lectura de partir

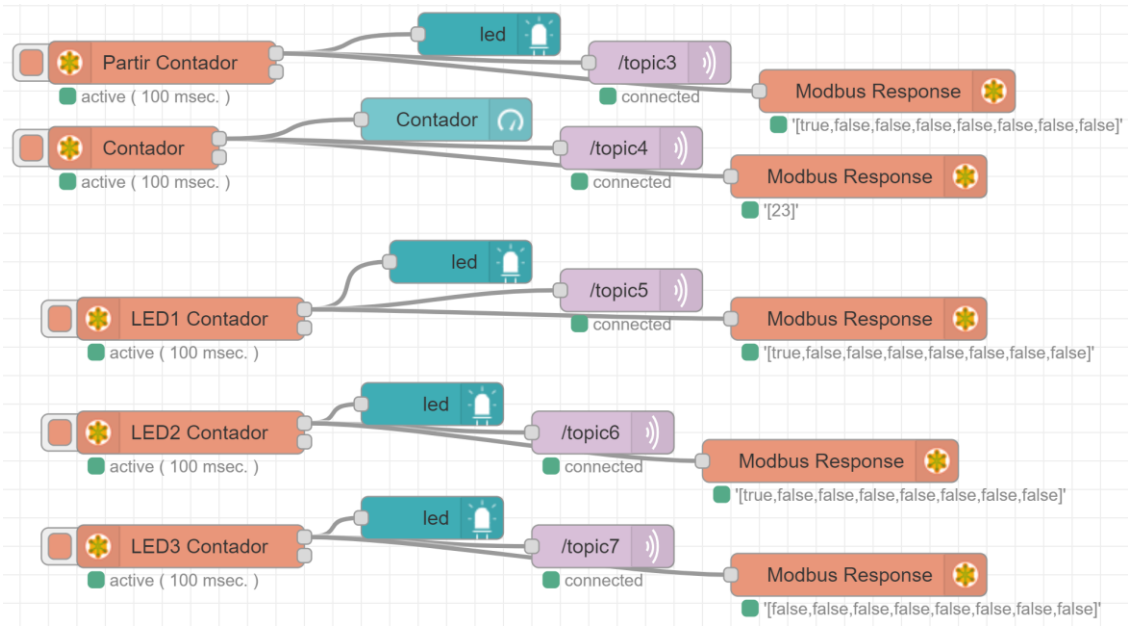


Figura 4-46. Flujo de Contador y secuencia de LEDs

En la Figura 4-46 se obtienen los datos de lectura por Nodos Modbus para lo que sería el contador y la secuencia de LEDs realizada en la programación Ladder, con sus respectivos Nodos de comunicación MQTT con los topics declarados asociados a los que están especificados en la Nube (instancia).

4.2.3 Instancia (Simulación):

En la Figura 4-47 se realizó un flujo de forma directa en la nube para representar un deslizador y un visualizador del tipo gauge, para pruebas de funcionamiento del Dashboard entre varios dispositivos que se conecten a este, además de un nodo inyector con un Debug, que sirve para ver el dato recibido en el apartado de depuración de Node-RED (solo accesible para el usuario local), esto para realizar pruebas de comunicación MQTT.

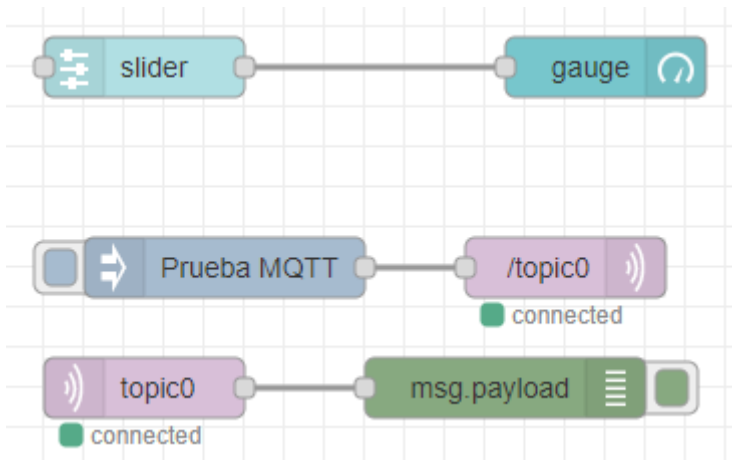


Figura 4-47. Prueba de deslizador

Para la Figura 4-48 se tiene la representación del flujo para una parte de lo que se reflejaría en el Dashboard como tal, por tanto, se tienen los datos recibidos por los respectivos Nodos MQTT y sus topics relacionados a los del Node-RED Local, este flujo es el que muestra el registro realizado por las pulsaciones por segundo con el LED ON/OFF. Además de por medio se tuvo que colocar Nodos Json, esto porque el dato enviado a través de la comunicación MQTT por medio de los Nodos y topics especificado realizan una conversión del tipo de dato a string, con ayuda del Nodo Json se puede transformar el tipo de dato para poder ser leído por el Nodo específico que lo visualiza en el Dashboard (Gauge, LED).

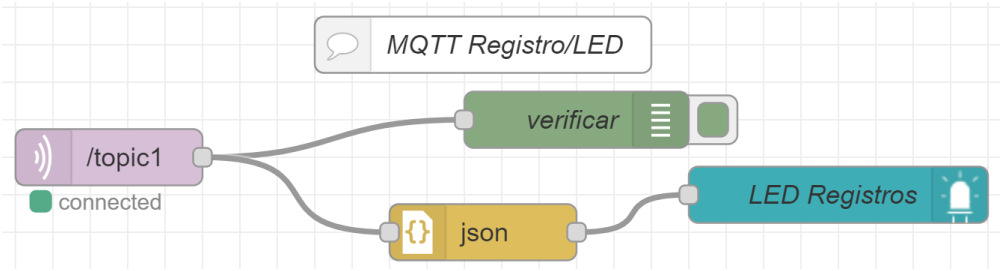


Figura 4-48. Registro de pulsos con verificación ON/OFF (LED)

Para la Figura 4-49 se ve el flujo creado para la representación del Contador y la secuencia de LEDs creado en el modo Local, que ahora son recibidos mediante la comunicación MQTT en el Node-RED de la nube para así tener el acceso al Dashboard en múltiples dispositivos. Al igual que en la Figura 4-48, se tuvo que utilizar los Nodos Jason para volver a transformar los tipos de datos hacia los Nodos del Dashboard. Cabe destacar que, para encontrar la solución a la transformación de los datos, se utilizó un Nodo Debug como el que se aprecia en la Figura 4-48 (verificar), este Nodo permitió ver el tipo de dato que llegaba en la interfaz de depuración del programa, detalle que permitió comprobar que el dato se transformaba a tipo String entre la comunicación de los Nodos MQTT, y el tipo de variable a ocupar para una correcta lectura de los datos en los Nodos de Dashboard eran del tipo Array.

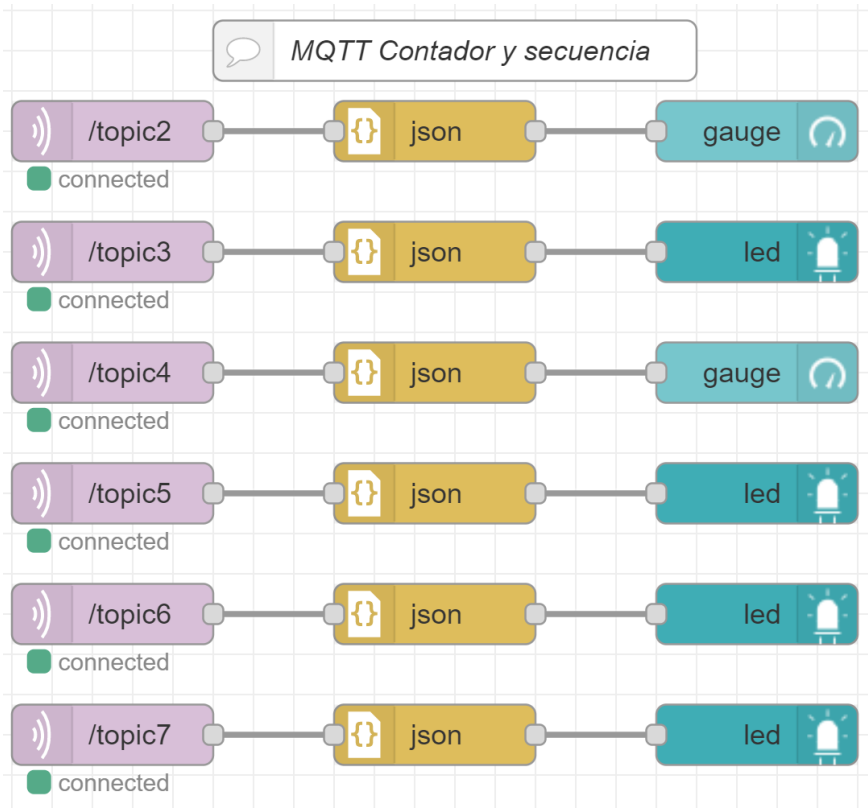


Figura 4-49. Flujo con los Nodos MQTT asociados a los tópicos y representación del Dashboard

4.3 DASHBOARD (KPI):

4.3.1 Simulación:

En la Figura 4-50, ya se ve una parte del Dashboard como tal, que es como lo vería un cliente con acceso a la dirección de este, a su vez puede controlar ese deslizador de prueba que se vería reflejado en el movimiento y contador del gauge. Esto en tiempo real para todos quienes estén conectados, por tanto, esto sirve de un modo de prueba para verificar que cada usuario ve en tiempo real y el control de este, lo que no representa en nada la comunicación MQTT, pero si la eficiencia de acceso simultaneo de una gran cantidad de usuarios que puedan interactuar desde la nube a un pequeño simulador visual.



Figura 4-50. Dashboard de prueba del deslizador y el gauge

Para la Figura 4-51, se ve el Dashboard de la programación visual que se realizó en el Ladder del registro de datos por pulsos cada un segundo, que se ve del lado izquierdo de la imagen (Simulación 1), con su LED ON/OFF y el Gauge que muestra la cantidad de pulsos registrados por segundos hasta llegar a 200 y luego se reinicia.

Al lado derecho de la imagen de la Figura 4-51 se tiene la Simulación 2, el que representa de forma visual lo que sería el Contador realizado más la secuencia de LEDs generada. Hay que destacar que solo se tomó en cuenta con un KPI (visual), ya que con un sistema de control remoto pone en riesgo de algún atacante o agente externo intervenga y modifique las variables (si se tratase para un sistema industrial). De forma didáctica o particular se puede hacer un sistema de control sin problemas.

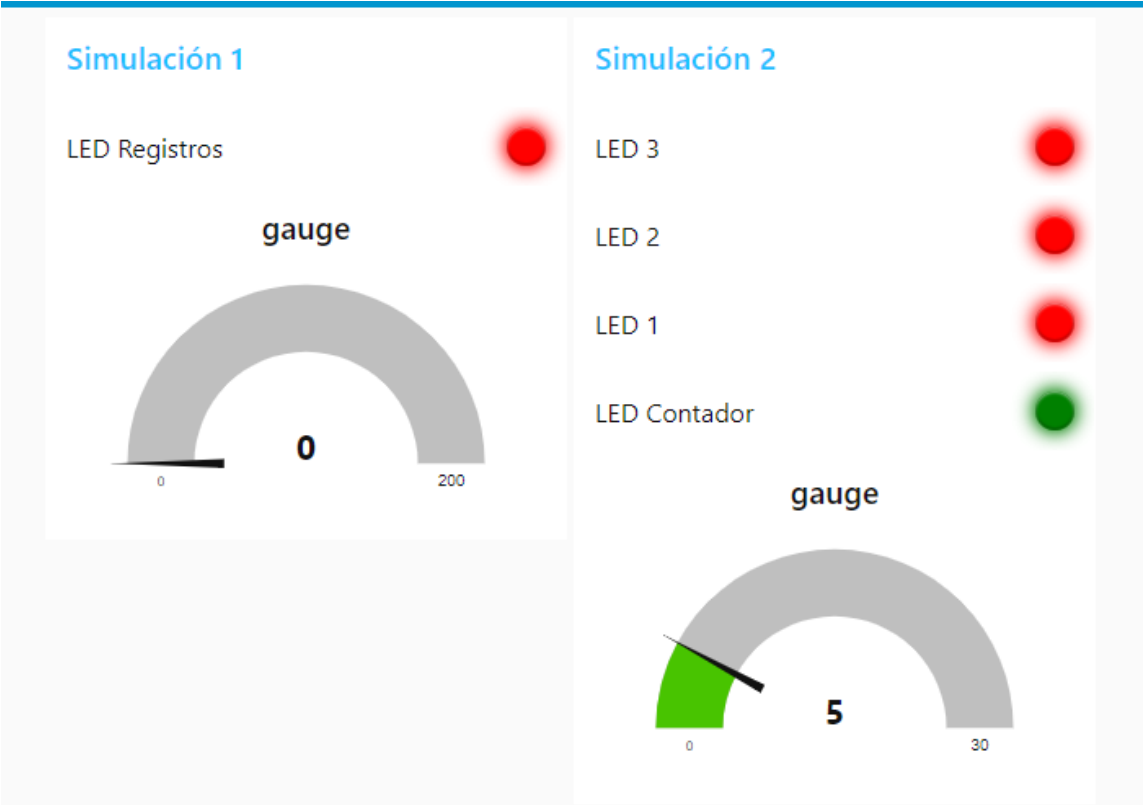


Figura 4-51. Dashboard con el gauge de registro, contador y secuencia de LEDs

CAPITULO 5: HOSTING WEB GCP

5.1 GCP:

El presente proyecto, se encuentra alojado en un servicio de hosting (servidor) web para poder visualizar desde distintos dispositivos con accesos a internet, y así que varios usuarios puedan tener acceso visual al KPI desarrollado.

De forma nativa se utilizó el hosting integrado de la misma plataforma en la que se creó la instancia para el proyecto, Google Cloud.

Cuando se creó la instancia, está por defecto en las opciones de firewall (Figura 5-52), entrega de la opción de habilitar el tráfico HTTP y HTTPS, el cual permite ingresar de otras plataformas a la instancia, algo que no lo hace muy seguro por dar acceso a cualquiera que obtenga la dirección estática creada para la instancia, por la misma razón de seguridad es que Node-RED para estos casos, solo sirve para brindar una ayuda visual del proceso que se quiera ver, pero no así el implementar algún tipo de HMI, lo que conlleva riesgos de adulteración o mala manipulación de algún tercero con acceso a la dirección.

Para este caso, en particular de querer dar a establecer la comunicación del protocolo MQTT, este acceso propio que brinda Google Cloud como un hosting básico, sirve bastante bien.

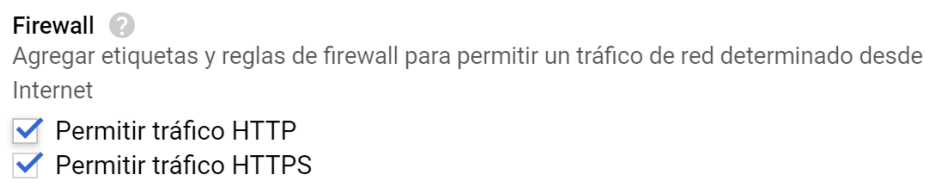


Figura 5-52. Permisos de firewall para accesos HTTP y HTTPS

5.1.1 Apache:

Google Cloud entre sus variadas herramientas, también ofrece servicios de Hosting Web profesionales, los cuales se pueden diseñar bajo otras plataformas que se integran a este, como el Apache.

Apache es un servidor web HTTP de código abierto. Esta desarrollado y mantenido por una comunidad de usuarios en torno a la Apache Software Foundation. Actualmente y desde 1996 es el servidor web más usado en todo el mundo debido a su seguridad y estabilidad.

La funcionalidad principal de este servicio web es servir a los usuarios todos los ficheros necesarios para visualizar la web. Las solicitudes de los usuarios se hacen normalmente mediante un navegador.

Algunas ventajas de Apache:

- ❖ De código abierto y gratuito, con gran comunidad de usuarios.
- ❖ Parches de seguridad regulares y actualizados con frecuencia.
- ❖ Estructura basada en módulos.
- ❖ Multiplataforma. Está disponible en servidores Windows y Linux.
- ❖ Personalización mediante .taccess independiente en cada hosting.

Desventajas de Apache:

- ❖ Presenta problemas de estabilidad por encima de las 10000 conexiones.
- ❖ Un uso abusivo de módulos puede generar brechas de seguridad.

CONCLUSIONES:

MQTT es un protocolo seguro, flexible y robusto, que para IoT/IIoT funciona muy bien. Por lo cual ayuda a definir o establecer comunicaciones entre dispositivos y la nube que pueden llegar a adaptarse a una gran cantidad de usuarios o clientes.

El desarrollo de la comunicación MQTT en conjunto de aplicación como las presentadas (KPI), seguirá siendo implementada durante los próximos años y cada vez más, siendo de vital importancia para la modernización de los procesos y, sobre todo, para que el internet siga siendo la base para que se vayan construyendo más innovaciones tecnológicas.

Si bien, como se ha detallado MQTT es una comunicación segura, para poder ocupar un control remoto es recomendable aplicar más capas de seguridad, afectando cerrar puertos, los cuales interfieren con la utilización de algunos softwares. Por lo cual, esta forma de comunicación se recomienda más para una visualización de procesos (KPI), como el mostrado en el presente proyecto, y para lo que sería control de procesos una HMI como tal.

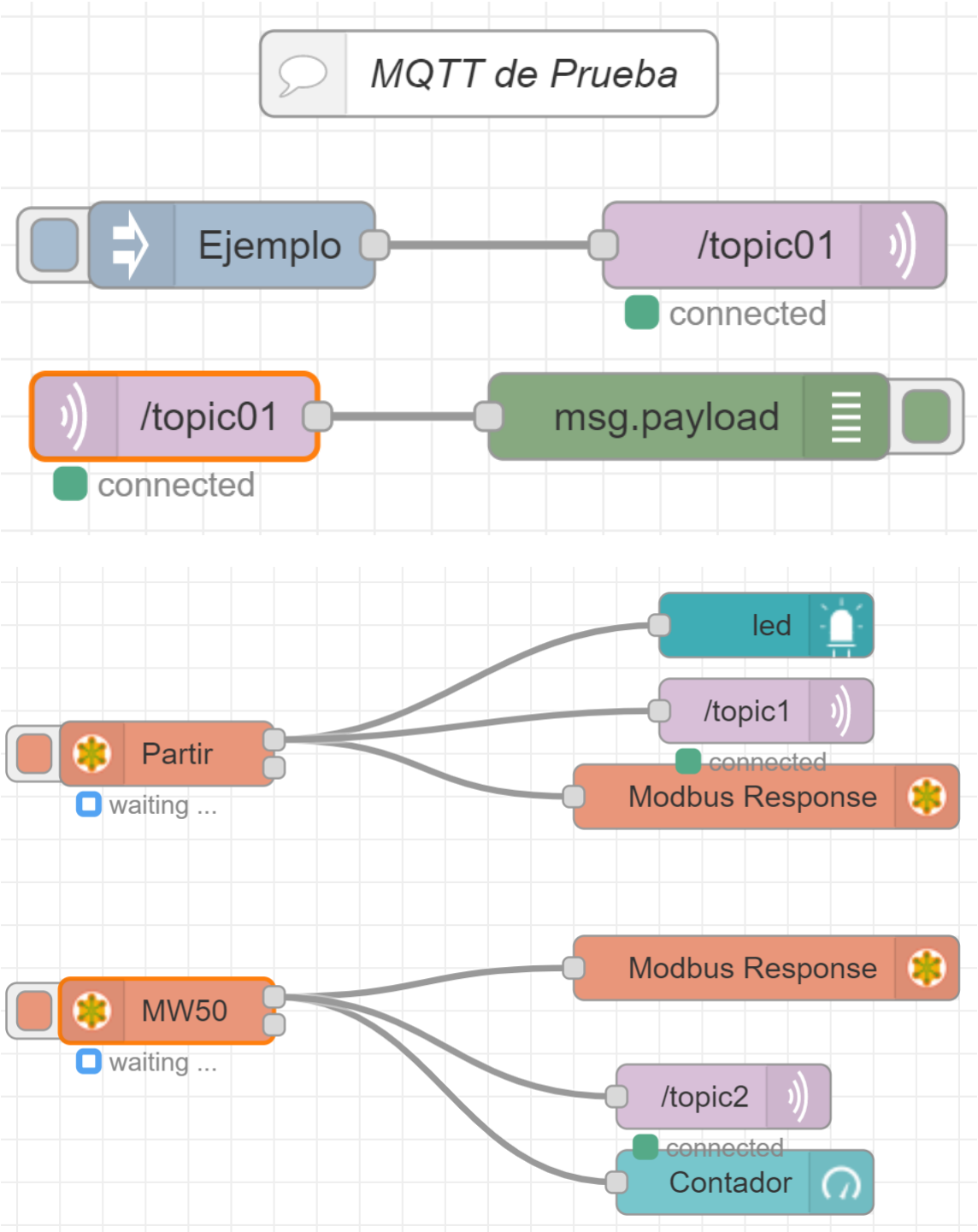
BIBLIOGRAFIA:

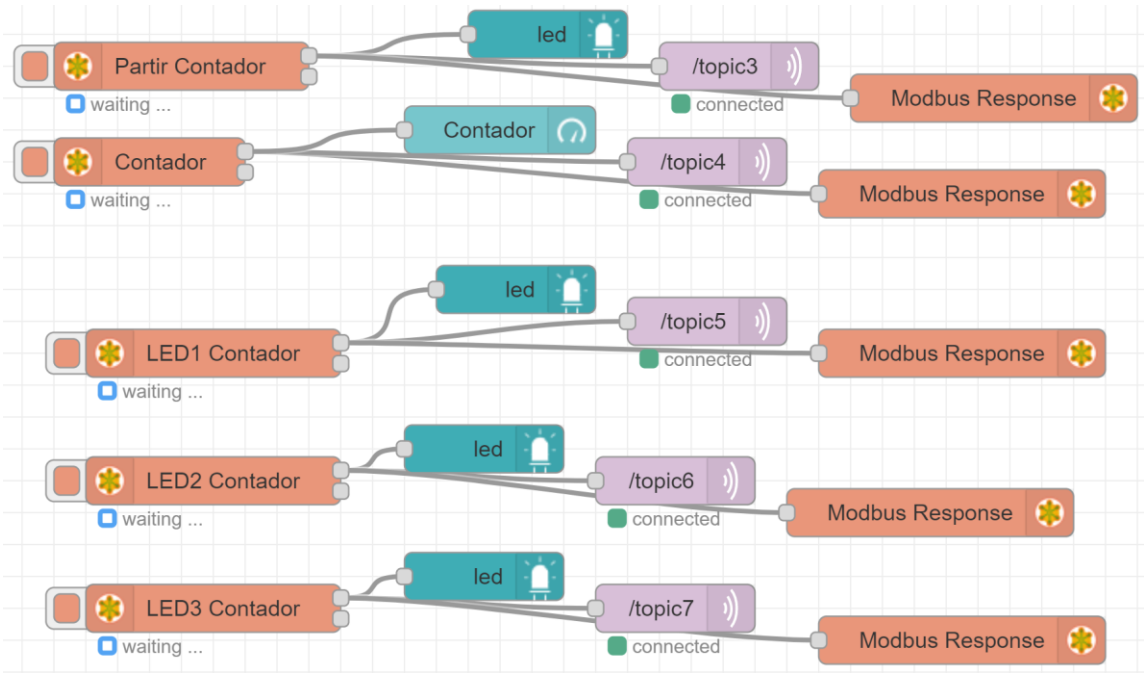
- Google Cloud. Crea e inicia una instancia de VM. (Disponible en: <https://cloud.google.com/compute/docs/instances/create-start-instance>).
- MQTT. ¿Qué es MQTT? Su importancia como protocolo IoT. (Disponible en: <https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/>).
- MQTT. Connect to an MQTT Broker. (Disponible en: <https://cookbook.nodered.org/mqtt/connect-to-broker#:~:text=Many%20users%20will%20run%20an,an%20MQTT%20broker...%20.>).
- Node-RED. Instalar node red en Windows. (Disponible en: <http://www.codigoelectronica.com/blog/instalar-node-red-en-windows>).
- Node-RED. ¿Qué es node red? (Disponible en: <http://codigoelectronica.com/blog/que-es-node-red#que-es-node-red>).
- Node.js. Installing Node.js via package manager (Disponible en: <https://nodejs.org/en/download/package-manager/#windows>).
- EMQ X. EMQ X Broker (Disponible en: <https://docs.emqx.io/en/enterprise/v4.3/>).

ANEXOS:

ANEXO 1:

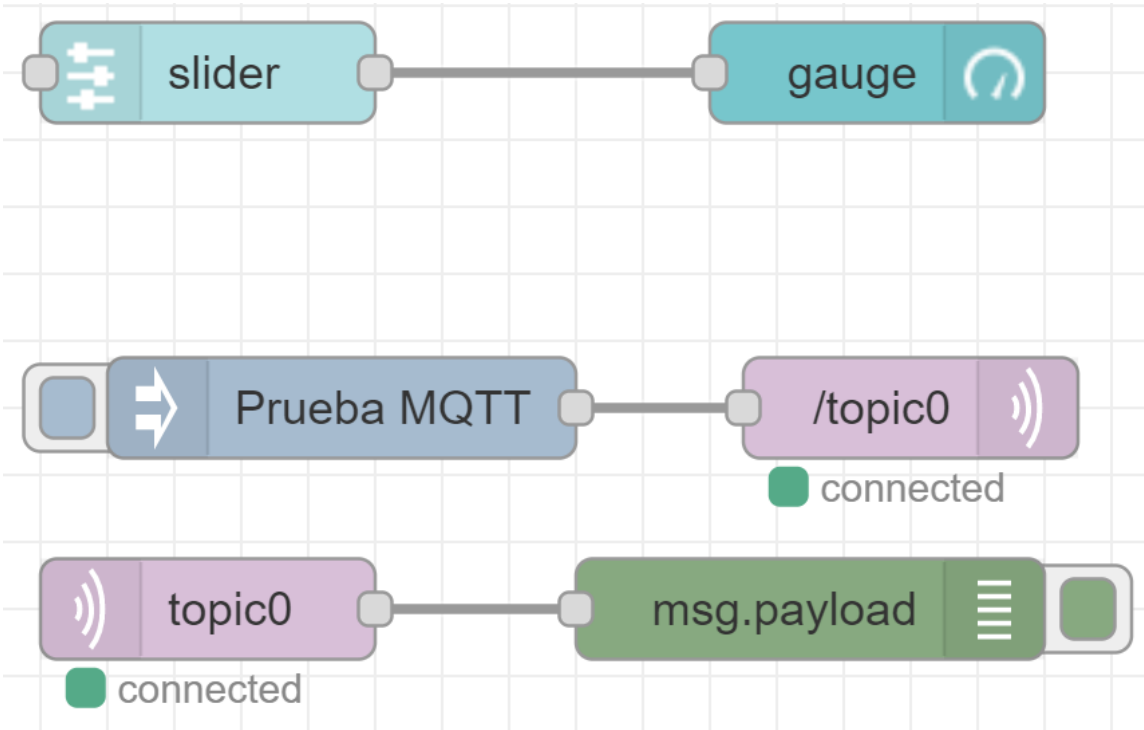
Programación Local en Node-RED:

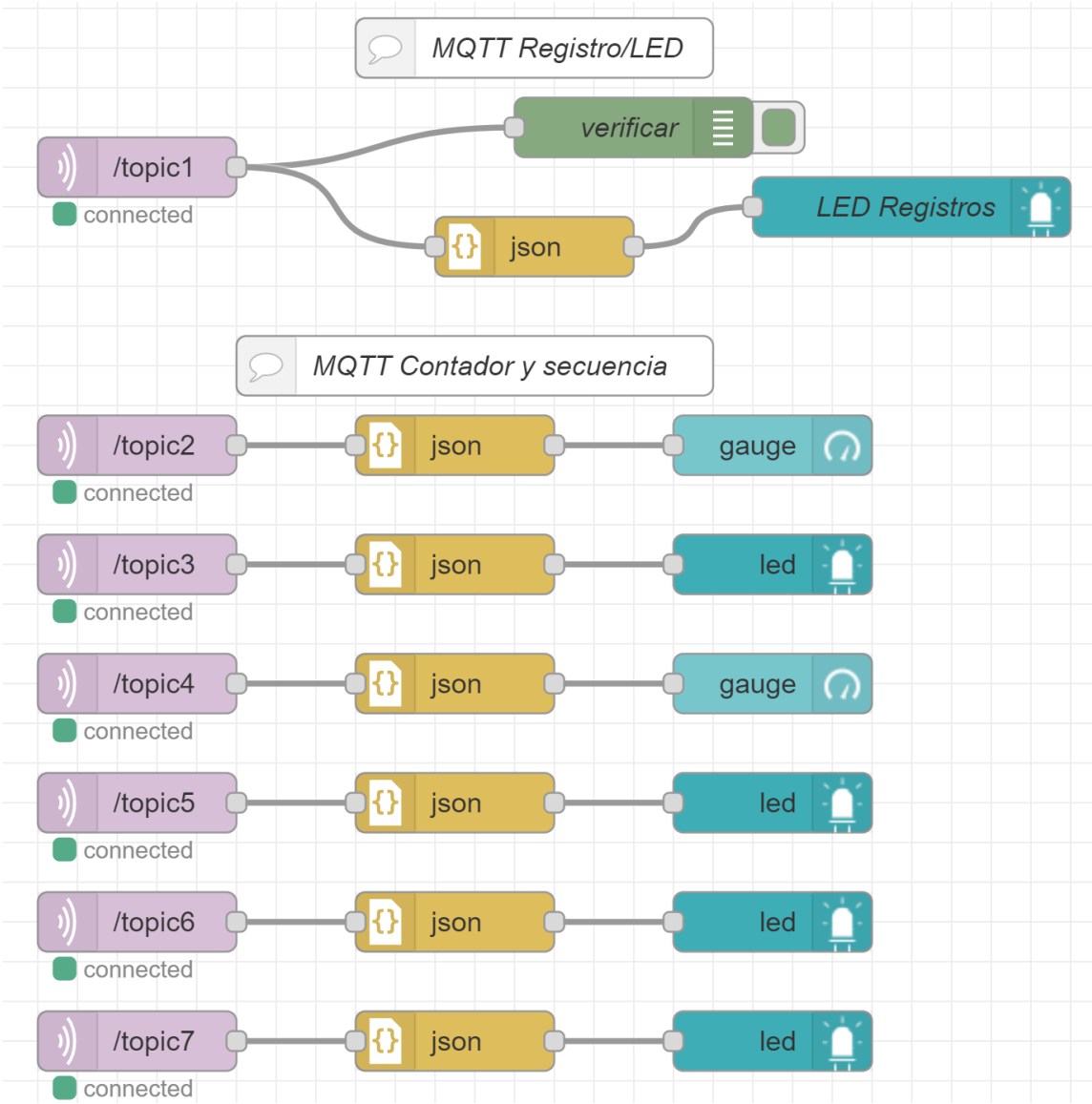




ANEXO 2:

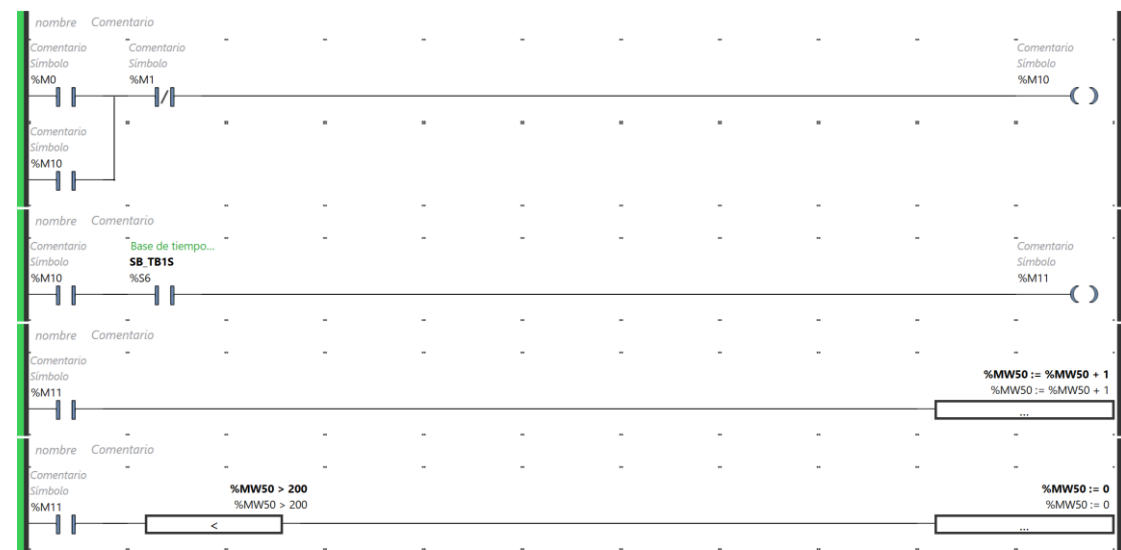
Programación en la Nube de Node-RED:





ANEXO 3:

Programación Registro de pulsos en PLC:



Programación Contador mas secuencia de LEDs en PLC:

