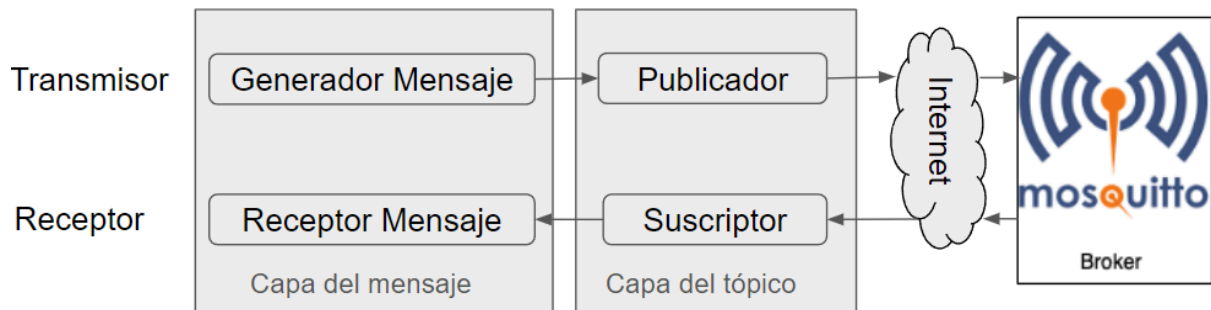


El servidor Mosquito y el servidor Node-RED

Tarea 1. Una aplicación Python para acceder remotamente a Mosquitto.

El reto consiste en implementar el sistema que se ilustra en la siguiente figura. Se usará python en un computador remoto para implementar el transmisor (Generador Mensaje y el Publicador), en otro computador remoto se implementará el receptor en python (Receptor Mensaje y Suscriptor). Está claro que en nuestra VM en Azure, tenemos el servidor de Mosquitto.



Paso 1. Debe estar encendida la VM en Azure. No es necesario entrar a la VM porque durante la instalación de Mosquitto, lo programamos para que arranque por defecto, osea que una instancia de mosquitto siempre estará trabajando.

Paso 2. Implementamos el Receptor, mediante Google Colab. [Enlace al código](#). Ponemos a correr ese código para que el suscriptor se quede escuchando a mosquitto.

Paso 3. Implementamos el Transmisor, mediante Google Colab. [Enlace al código](#). Corremos el código, mientras observamos si el mensaje generado en el transmisor llega al receptor.

Para el informe Tarea 1. Un transmisor hecho en python, ubicado en un lugar remoto del mundo, envía mensajes a un receptor similar en otro lugar remoto, usando a nuestro servidor Mosquitto como broker para la distribución de los mensajes.

Explique si se logra el responder positivamente al reto planteado y qué pasa en cada componente del sistema:

Si se logra cumplir lo planteado en el reto ya que se realiza la correcta distribución de los mensajes gracias a los códigos de Python ya sea en el entorno de Google Colab o cualquier editor de texto, así como se muestra más adelante con el ejemplos al ejecutar los códigos en Google Colab y en el IDLE Shell de Python; continuando con la explicación se define lo siguiente para lograr una efectiva distribución de los mensajes a través de los códigos del transmisor, receptor y obviamente la VM creada en Azure para tener la conexión con Mosquitto por medio del puerto 1883 previamente ya configurado en el firewall del PC y una regla de puerto de entrada en Azure, de lo contrario no se hará una correcta conexión y comunicación del sistema planteado.

Paso 1: Verificación de la VM

Se debe asegurar que la **VM en Azure esté encendida** y funcionando. Como ya se ha configurado que el servidor Mosquitto se inicie automáticamente (esto en una terminal CMD), este debería estar ejecutándose sin que necesites ingresar a la VM cada vez (para revisar el estado de Mosquitto se ingresa el siguiente comando en la terminal: `systemctl status mosquitto`).

Paso 2: Implementación del Receptor (Suscriptor) en Python

1. **Código del Receptor:** El receptor, que se suscribe al servidor Mosquitto, debe configurarse en Python para conectarse a la IP de la VM en Azure en el puerto 1883 (el puerto por defecto de MQTT).
 - En el código del receptor, ya se tiene configurada la conexión a Mosquitto usando la IP de la VM que en este caso es: **172.171.224.96**. Se debe asegurar de que la IP ingresada es correcta y que corresponde a la IP pública de la VM en Azure.
 - Se usa un **“Bucle de escucha”** para que el receptor se quede suscrito y recibiendo mensajes. En el código, la función `miCliente.loop_forever()` permite que el suscriptor se mantenga escuchando de forma continua.
2. **Ejecución del código del receptor:** Se ejecuta este código en Google Colab o cualquier editor de texto y se revisa que se mantenga corriendo (“Bucle de escucha”). Esto indica que el suscriptor está en espera de mensajes de Mosquitto.

Paso 3: Implementación del Transmisor (Publicador) en Python

1. **Código del Transmisor:** El transmisor es responsable de **publicar mensajes** en un tema (**tópico**) específico al servidor Mosquitto.
 - En tu código del transmisor, se define un tema (**tópico**) llamado **MFC_topic** en este caso y se envía el mensaje al broker de Mosquitto en la misma IP (**172.171.224.96**).
 - Se debe asegurar de que el mensaje esté bien configurado y que el tópico coincida con el tópico al cual el receptor está suscrito.
2. **Ejecución y Prueba del transmisor:** Se ejecuta este código en Google Colab o cualquier editor de texto y observa en el código del receptor si el mensaje es recibido. Si el receptor imprime el mensaje, significa que la comunicación entre ambos está funcionando correctamente.

Explicación Técnica

- **Mosquitto** es el broker de MQTT que está gestionando los mensajes entre el transmisor y el receptor. Como el servidor Mosquitto está en la VM en Azure, sirve como el intermediario para los mensajes que se envían y reciben.
- **Transmisor y Receptor:** El transmisor publica mensajes en un tópico en el servidor Mosquitto, y el receptor se suscribe a ese mismo tópico para recibir los mensajes. Esto simula un sistema de comunicación básico en IoT.

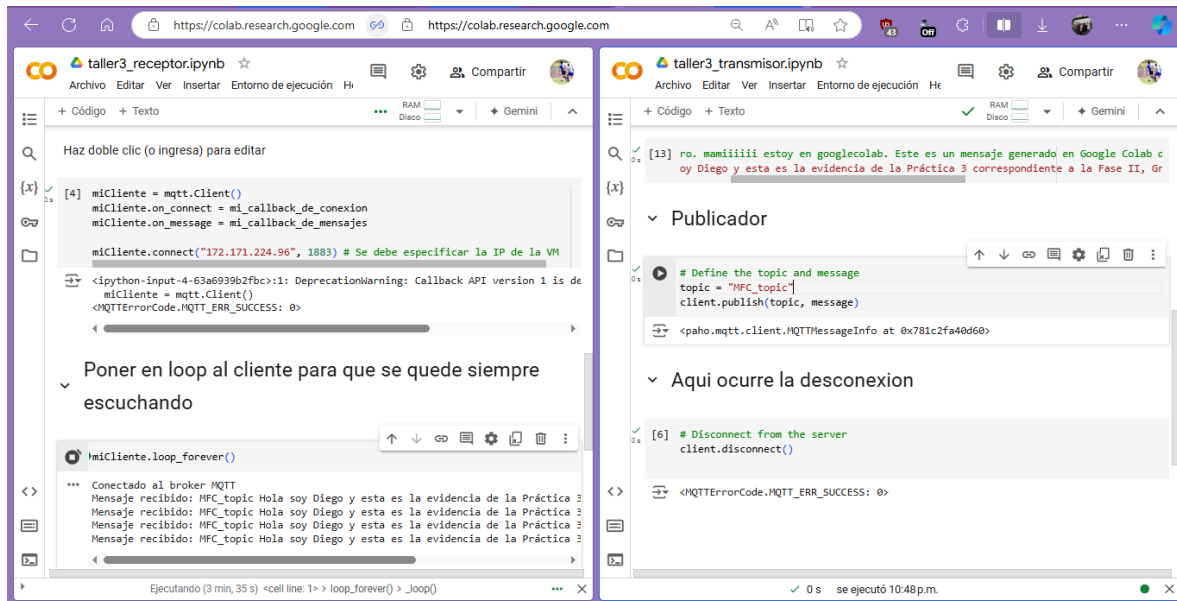
Notas Adicionales

- **Conexión Remota:** Al usar Google Colab o cualquier otro editor de texto, se está

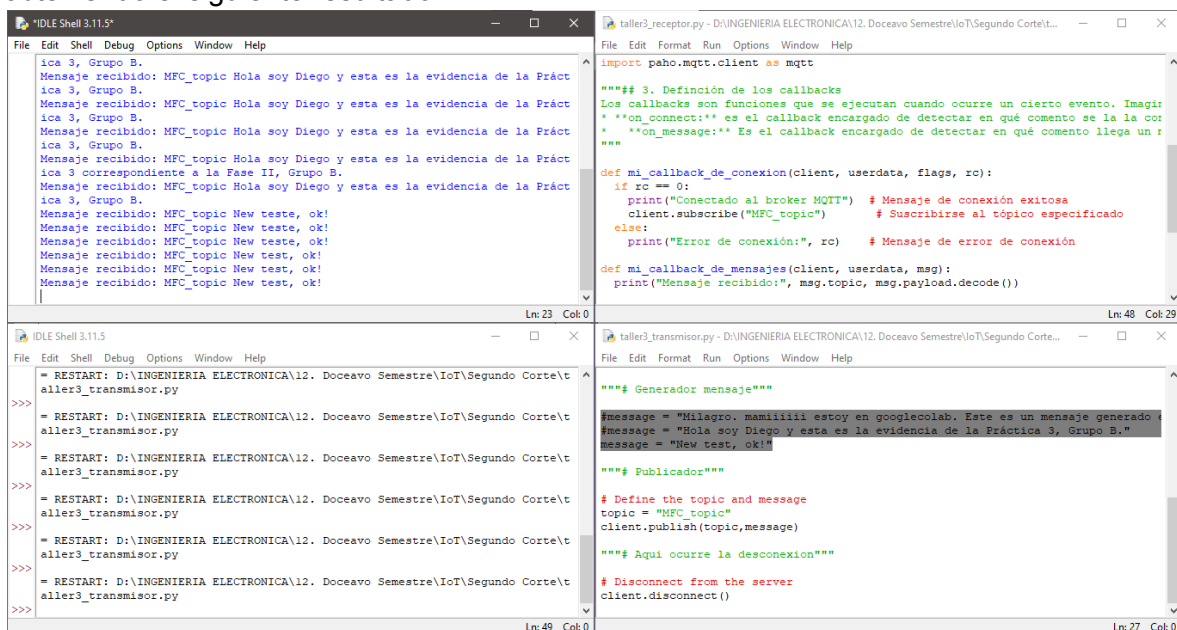
conectando remotamente al servidor Mosquitto en Azure, lo cual es ideal para pruebas sin tener que instalar MQTT localmente.

- **Temas (tópicos) de MQTT:** Es fundamental que el tópicos en el transmisor (**MFC_topic** en este caso) sea el mismo que en el receptor para que la comunicación funcione.

Inicialmente se quiso realizar la prueba directamente desde **Google Colab**, obteniendo el siguiente resultado, se aclara que debe estar activa la VM desde un terminal (CMD) y Mosquitto debe estar activo, para ello se verifica con el siguiente comando **sudo systemctl status mosquitto**.



Además de realizar la prueba desde el entorno de Google Colab, se quiso realizar también una prueba pero directamente en el **IDLE Shell 3.11.5** de Python que se instala por defecto al instalar Python directamente en el PC, se aclara que se debe instalar la librería en una terminal (CMD) con el siguiente comando **pip install paho-mqtt** obteniendo el siguiente resultado:



Se concluye que directamente en el PC y obviamente teniendo instalado Python y no solo en el IDLE Shell de Python sino en cualquier editor de texto, sin hacer uso del entorno de Google Colab, aunque igualmente se debe tener abierta la VM en una terminal (CMD).

Piense y describa una solución del mundo real que podría ser implementada usando este sistema como punto de partida.

Ejemplo: Sistema de Monitoreo y Control de Calidad del Agua en Granjas Acuapónicas

Descripción General

La calidad del agua es crucial para la salud de los peces y el crecimiento de las plantas en un sistema acuapónico. Variables como el pH, la temperatura y el oxígeno disuelto deben mantenerse en rangos específicos para evitar problemas en el ecosistema. Este sistema permitiría que los operadores monitoreen y ajusten automáticamente las condiciones del agua desde cualquier lugar, mejorando la eficiencia y reduciendo la necesidad de intervención manual.

Componentes del Sistema

1. **Sensores:** Dispositivos que miden variables críticas como:
 - **pH:** Para medir el nivel de acidez o alcalinidad del agua.
 - **Oxígeno disuelto (OD):** Para asegurar que haya suficiente oxígeno para los peces.
 - **Temperatura:** Para mantener el agua en un rango adecuado para la fauna y flora.
2. **Microcontroladores:** Controladores como Arduino o ESP32, que leen los datos de los sensores y los envían a través de MQTT al broker Mosquitto en la nube o en un servidor local.
3. **Broker MQTT (Mosquitto):** Actúa como intermediario para los mensajes entre los sensores y los sistemas de monitoreo o control. Este broker puede estar en la nube (como en la VM de Azure) o en una infraestructura local.
4. **Plataforma de Monitoreo:** Utilizando Node-RED o una aplicación web en la nube, los operadores pueden visualizar los datos en tiempo real y recibir alertas cuando alguna variable está fuera del rango óptimo. Además, se pueden programar reglas para activar respuestas automáticas.
5. **Actuadores:** Dispositivos como bombas de aire, calentadores de agua o válvulas dosificadoras que ajustan las condiciones del agua cuando el sistema lo detecta necesario.

Funcionamiento del Sistema

1. **Recolección y Envío de Datos:** Los sensores miden continuamente el pH, la temperatura y el oxígeno disuelto del agua. Estos datos son enviados por los microcontroladores al broker Mosquitto en la nube a través de MQTT.
2. **Procesamiento de Datos y Visualización:**
 - Los datos recibidos por el broker son procesados y visualizados en tiempo real en una plataforma como Node-RED.
 - Los operadores pueden monitorear el estado del agua y ver históricos de cada variable para tomar decisiones informadas.
3. **Automatización de Respuestas:**
 - Cuando una variable sale del rango deseado, el sistema puede enviar

comandos automáticamente a los actuadores correspondientes.

- Por ejemplo, si el nivel de oxígeno es bajo, el sistema puede activar una bomba de aire automáticamente.
- Si la temperatura es muy baja, puede activar un calentador, y si el pH está fuera del rango, se pueden activar válvulas para agregar productos que ajusten el pH.

4. **Alertas y Notificaciones:**

- En caso de situaciones críticas, el sistema envía notificaciones por correo electrónico o SMS a los operadores, permitiéndoles tomar acciones rápidas.
- Además, se pueden generar reportes periódicos sobre el estado del sistema y su funcionamiento.

Beneficios de la Solución

- **Monitoreo Remoto:** Los operadores pueden supervisar el sistema desde cualquier lugar con conexión a internet.
- **Automatización:** La capacidad de reaccionar automáticamente a condiciones fuera de rango minimiza la intervención humana, ahorra tiempo y reduce el riesgo de errores.
- **Escalabilidad:** La estructura MQTT permite que se agreguen más sensores y actuadores conforme crece la operación.
- **Sostenibilidad:** Ayuda a mantener condiciones óptimas para los peces y las plantas, optimizando el uso de recursos como el agua y los nutrientes.

Tarea 2. Configuración Node-RED. Instalación de complementos mínimos.

Paso 1. Asegúrate de tener la VM encendida. Entra a ella mediante una terminal SSH ([Enlace a explicación en el manual Azure sobre como abrir la VM por SSH](#)). Pon a correr node-RED mediante el comando:

```
node-red -v
```

Paso 2. Abrir Node-RED en un navegador

Comprueba que, sin tocar la máquina, el navegador de internet permite el acceso remoto a Node-RED. En el navegador escribe la IP pública seguida de “: 1880”, por ejemplo, si la IP es “54.236.56.149” deberías escribir:

<http://54.236.56.149:1880>

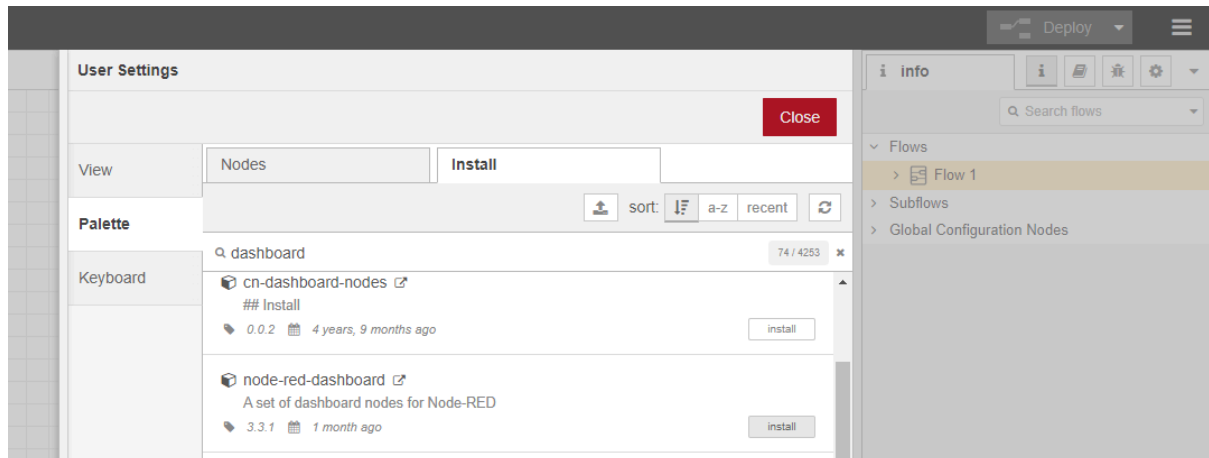
Paso 3. Instalar complementos mínimos de Node-RED.

El proceso para cualquier instalación es el siguiente:



> Manage Palette > Install > en el buscador escribe el nombre de lo que desea instalar y oprime enter > selecciona de la lista el producto que buscas y oprime el botón “Install”> Install

- Sigue el proceso señalado para instalar: **node-red-dashboard**
Es un módulo adicional para Node-RED que permite crear interfaces de usuario (dashboards) interactivas y personalizadas. Estas interfaces pueden incluir gráficos, botones, controles deslizantes, medidores, e indicadores que te permiten visualizar y controlar los flujos de datos en tiempo real. Este complemento es especialmente útil para aplicaciones de Internet de las Cosas (IoT), donde puedes necesitar una forma intuitiva de monitorizar y manejar dispositivos y sensores conectados.



- De manera similar instalar: **node-red-contrib-ui-led**
Permite agregar indicadores LED virtuales a tus dashboards creados con el "node-red-dashboard". Estos indicadores LED son útiles para mostrar el estado de diferentes procesos, dispositivos o señales de manera visual e intuitiva. Por ejemplo, puedes utilizar un LED verde para indicar que un sistema está operativo y un LED rojo para señalar un error o una condición de alerta.
- Instalar también: **node-red-mysql**. Nota: en 2023-2 vimos que ya no existe node-red-mysql sino **node-red-mysql-r2**. Todo indica que hace lo mismo- permite que un nodo ejecute solicitudes MySQL sobre una base de datos dada.

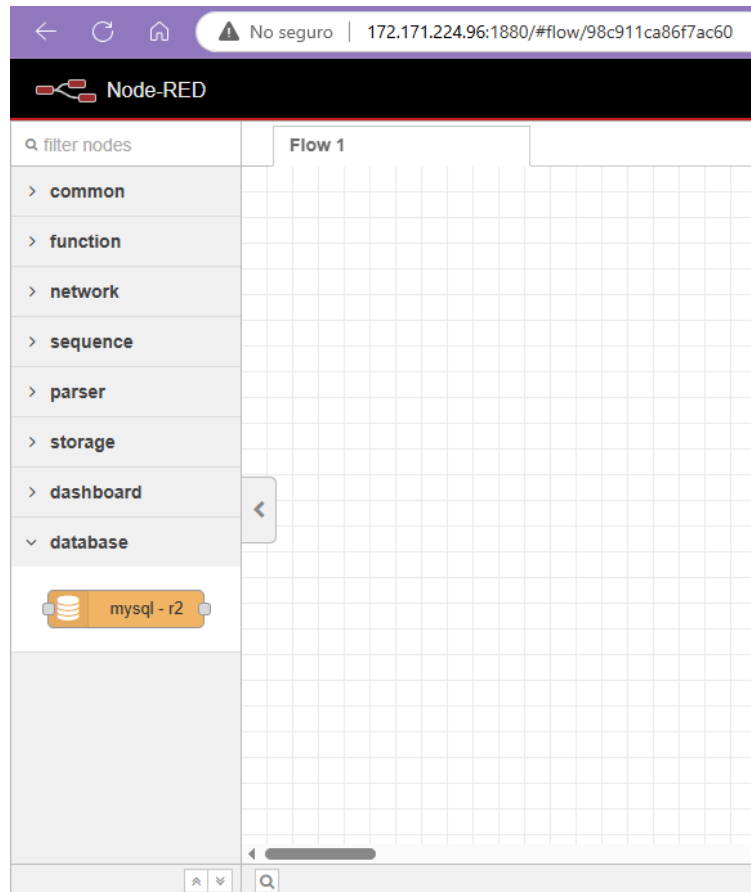
Paso 4. Llene la siguiente tabla para el informe.

Para el informe. Tarea 2: Evidencias de la instalación de complementos Node-RED

Evidencias del dashboard agregado. Tome una captura del menú lateral izquierdo, de manera que sea posible ver que existe una sección que despliega de nodos llamada "dashboard"



Evidencia de mysql - r2. Tome una captura del menú lateral izquierdo de manera que sea posible ver que existe una sección que despliega de nodos llamada "database" que tiene el nodo "mysql - r2"



Evidencia de led. Tome una captura del menú lateral izquierdo de manera que sea posible ver que existe el nodo “led”



Tarea 3. Pruebas de funcionamiento de Node Red

Node-RED usa el término “nodo” para referirse a lo que nosotros llamaríamos, de manera más natural, bloques. A veces le llamaremos bloques, pero recuerda que el término oficial es “nodo”. También es importante recordar que el conjunto de varios nodos enlazados, también se llama “nodo”, es muy similar a la programación orientada a objetos, donde varios objetos pueden componer a un nuevo objeto. Nosotros, para evitar confusiones llamaremos flujo-nodo a los nodos enlazados o flujogramas de nodos.

Paso 1. Implementación y configuración de un sistema IoT dentro de la misma VM

- Sistema a implementar:

La Figura siguiente contiene los elementos que implementaremos en esta prueba. Lo que indica es que hay que crear dos soluciones: una para el transmisor y otra para el receptor. El profesor ha ideado la manera de representar las dos cosas como un solo sistema de comunicaciones en un modelo de capas.

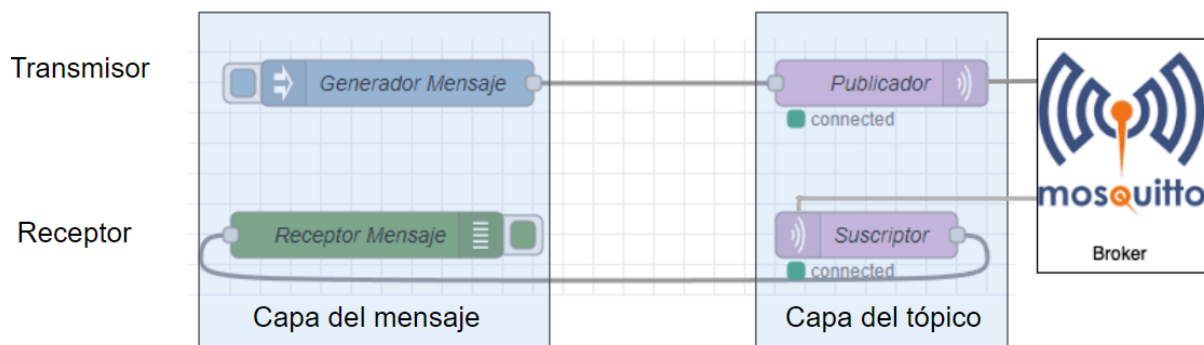


Figura 2.

- Bloques (nodos) a usar:

Busque en el menú lateral los siguientes bloques, arrástrelos al área de trabajo del flujo-nodo actual.

mqtt out: Representa al publicador “MiTopico”

mqtt in: Representa al suscriptor “MiTopico”










inject: Representa el Generador Mensaje











debug: Representa el Receptor Mensaje










y realice las conexiones así:



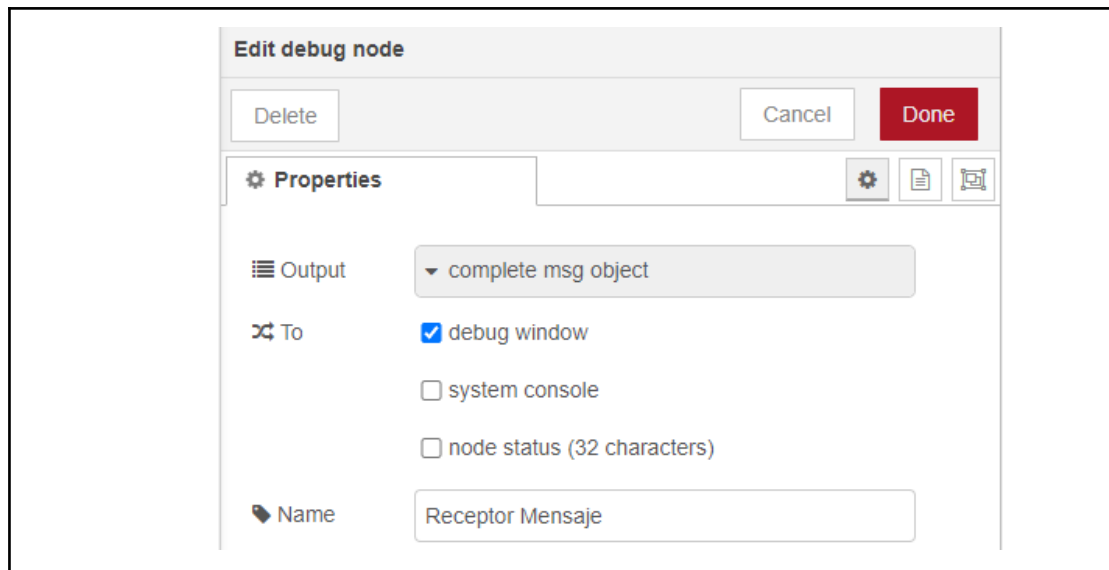
- Configuraciones

Publicador (nodo mqtt out)	
 Server	localhost:1883  
 Topic	MiTopico
 QoS	0   Retain 
 Name	Publicador



Suscriptor (nodo mqtt in)	
 Server	localhost:1883  
Action	Subscribe to single topic 
 Topic	MiTopico
 QoS	2 
 Output	auto-detect (parsed JSON object, string or buf 
 Name	Suscriptor

Generador Mensaje (nodo inject)	
 Name	Generador Mensaje
 msg. payload	=   hola, soy un bloquecillo de node red. 
 msg. topic	=   

Receptor Mensaje (nodo debug)	
-------------------------------	--




Paso 2. Funcionamiento

- Se oprime Deploy para que arranque
- Se oprime el botón izquierdo  del nodo "Generador Mensaje" cada vez que se desee enviar un nuevo mensaje
- Observa que el panel derecho, si se oprime  , corresponde a la ventana de visualización del nodo debug

La siguiente figura es como aparecen los resultados

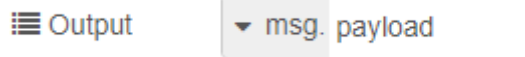
```
2/28/2023, 7:06:40 PM node: Receptor Mensaje
MiTopico : msg : Object
  ▶ { topic: "MiTopico", payload:
    "hello, soy un bloquesillo de n...",
    qos: 0, retain: false, _topic:
    "MiTopico" ... }
```

Que también, si se oprime  , veremos que se trata de los datos emitidos, pero organizados en formato JSON

```
2/28/2023, 7:06:40 PM node: Receptor Mensaje
MiTopico : msg : Object
  ▼ object
    topic: "MiTopico"
    payload: "hello, soy un bloquesillo
    de node red."
    qos: 0
    retain: false
    _topic: "MiTopico"
    _msgid: "7d863bebb5e0a8f5"
```

Paso 3. Conquista del Receptor Mensaje (nodo debug)

- Configurar el nodo para extraer del JSON y presentar en pantalla solo los datos que corresponden a valores. Entonces lo que interesa es msg.payload y la configuración es:




Oprima Done > Deploy >  en el "Generador Mensaje". El resultado debe ser

```
2/28/2023, 7:27:17 PM node: Receptor Mensaje
MiTopico : msg.payload : string[38]
"hello, soy un bloquesillo de node
red."
```

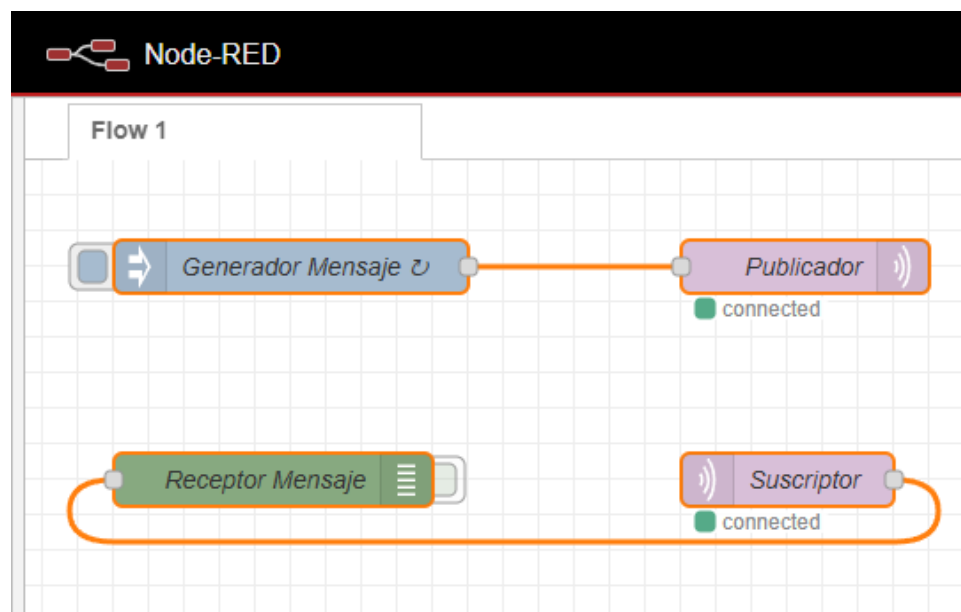
- De manera similar, configure el nodo para que muestre solo el tópico

Paso 4. Conquista total de los nodos usados:

Comprueba lo que ocurre cuando:

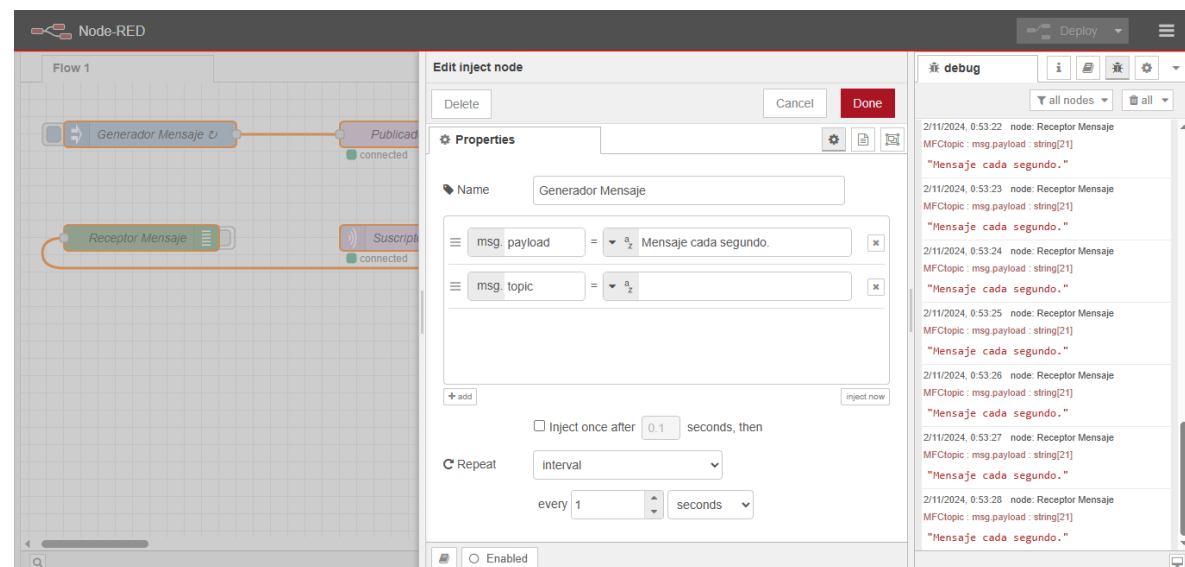
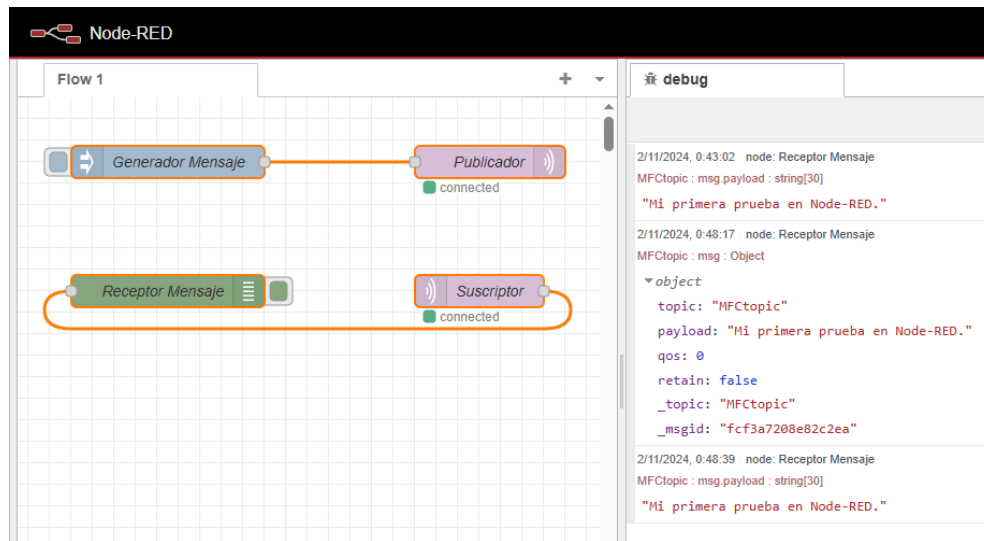
- Se activa o desactiva el botón que tiene a la derecha el bloque (nodo) debug (Receptor Mensaje).
- Se oprime el botón a la izquierda del bloque (nodo) inject (Generador Mensaje).
- Pruebe configurar el inject (Generador Mensaje) para que genere mensajes cada segundo.
- Compruebe que significa y en qué casos debería aparecer la señal  connected debajo de algunos bloques

Para el informe. Tarea 3. Tabla de resultados de primera implementación en Node-RED

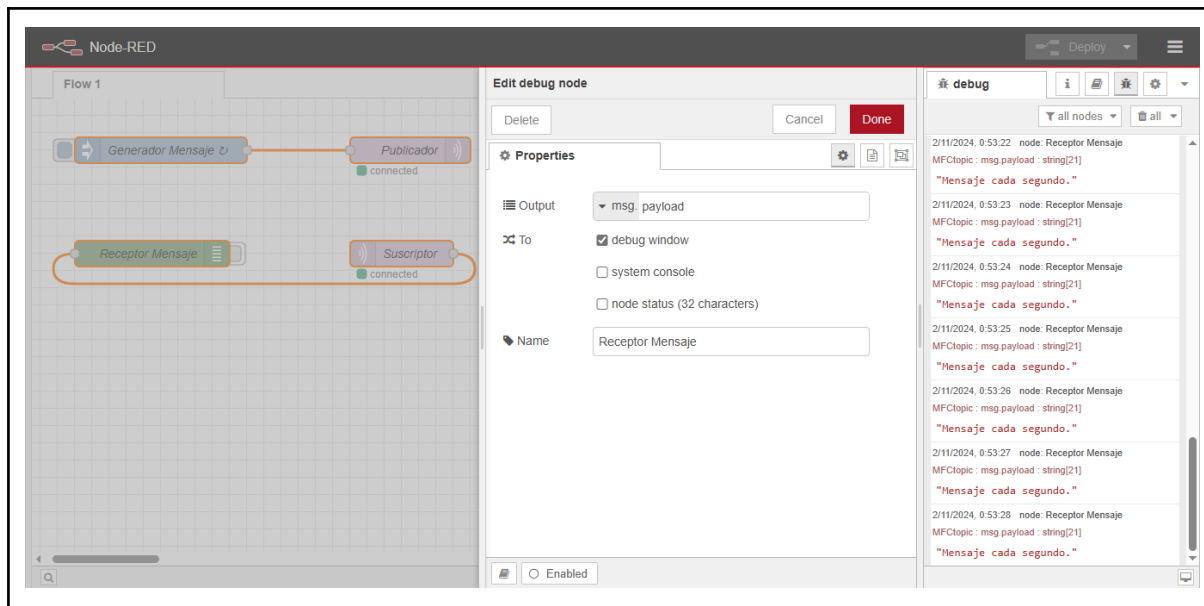


De acuerdo al paso 2, muestre un pantallazo en el que aparece la configuración del

“Generador Mensaje” y el resultado que entrega el “Receptor Mensaje”. Nota: use mensajes originales, propios, diferente a los de esta guía o de otros compañeros.

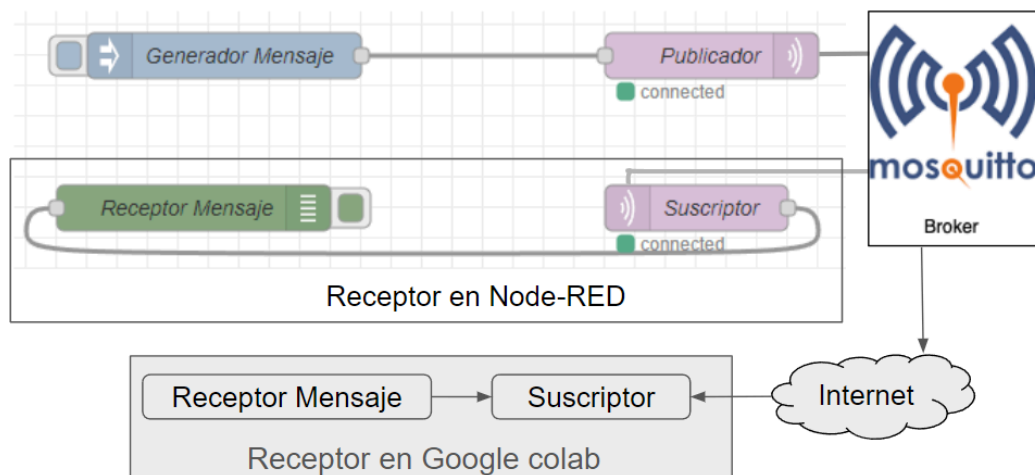


De acuerdo al paso 3, muestre un pantallazo de la configuración del “Receptor Mensaje” para obtener solo el tópic. También muestre el resultado que entrega el “Receptor Mensaje”



Tarea 4.

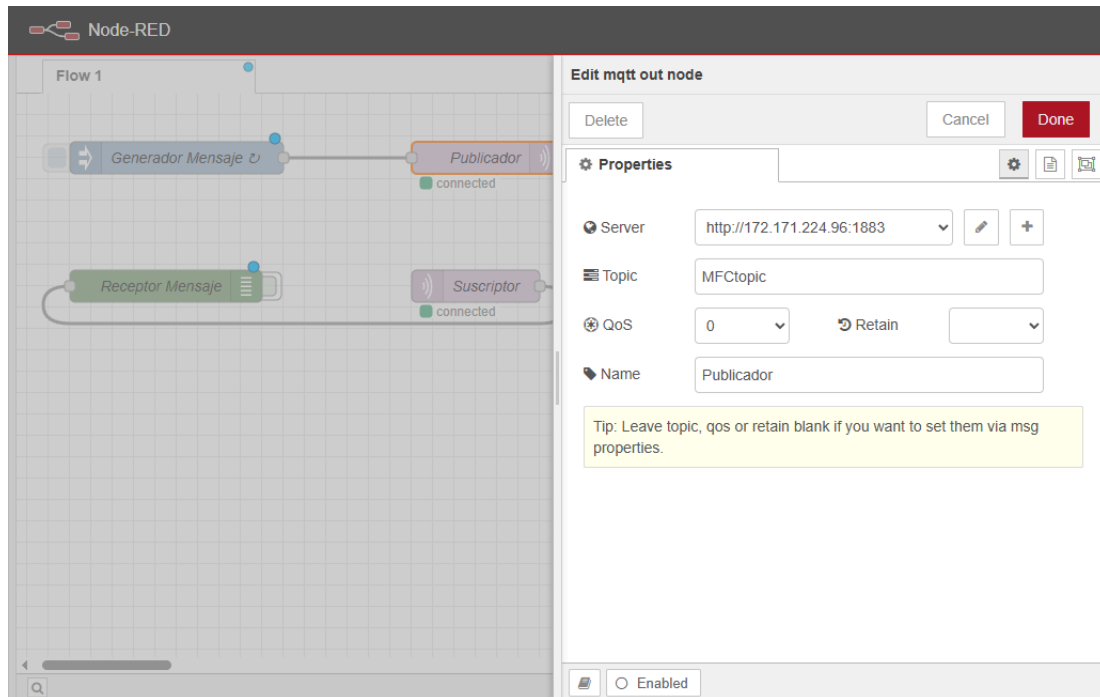
Usando todo lo aprendido, implemente una solución como la que se presenta en la figura siguiente



La idea es tan simple como comprobar que el mensaje que se genera, no solo llegue al “Receptor Mensaje” de Node-RED sino también, al mismo tiempo al implementado en Python con Google Colab.

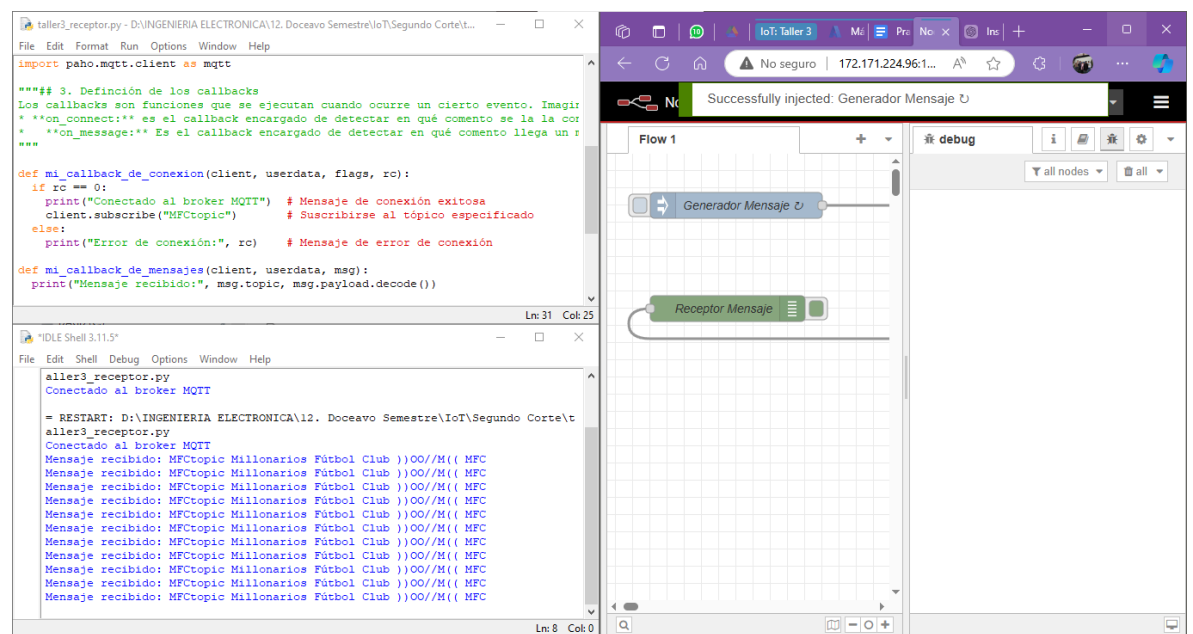
Para el informe. Tarea 4.

Muestre abajo el resultado de la implementación del “Suscriptor” remoto del mensaje.

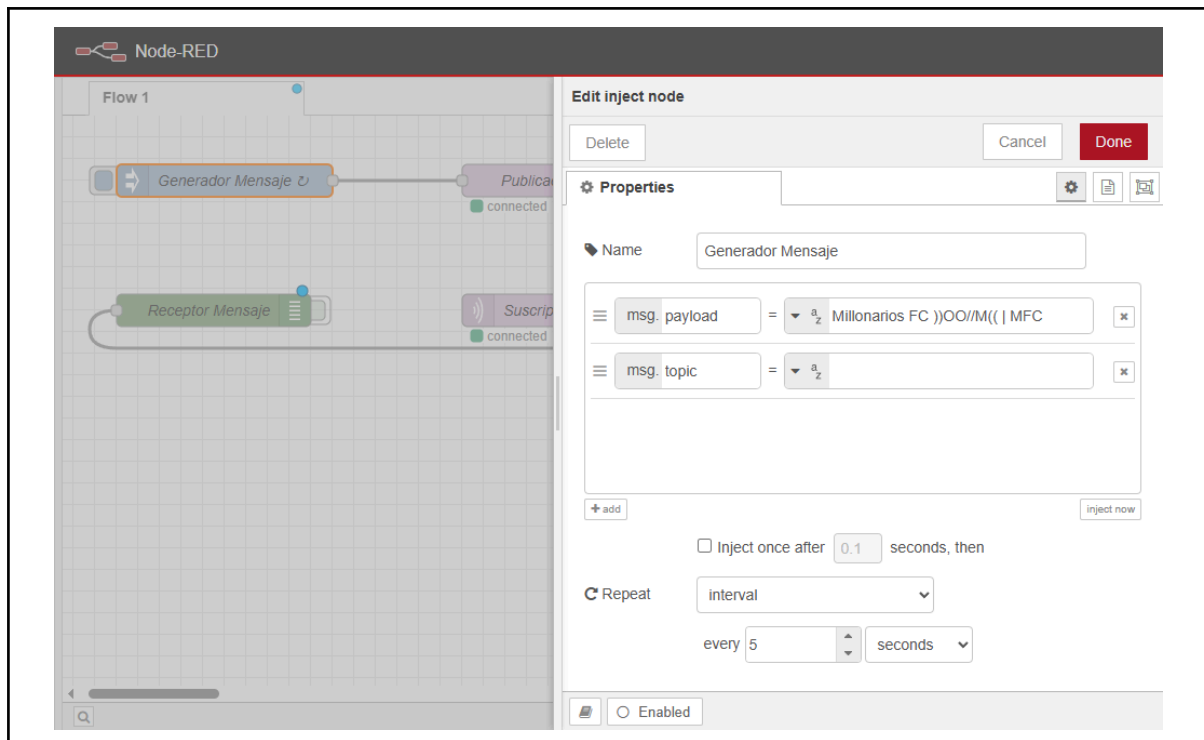


En este caso se reemplazó el servidor **localhost:1883** por la IP de la VM junto con el puerto 1883, es decir **http://172.171.224.96:1883/**

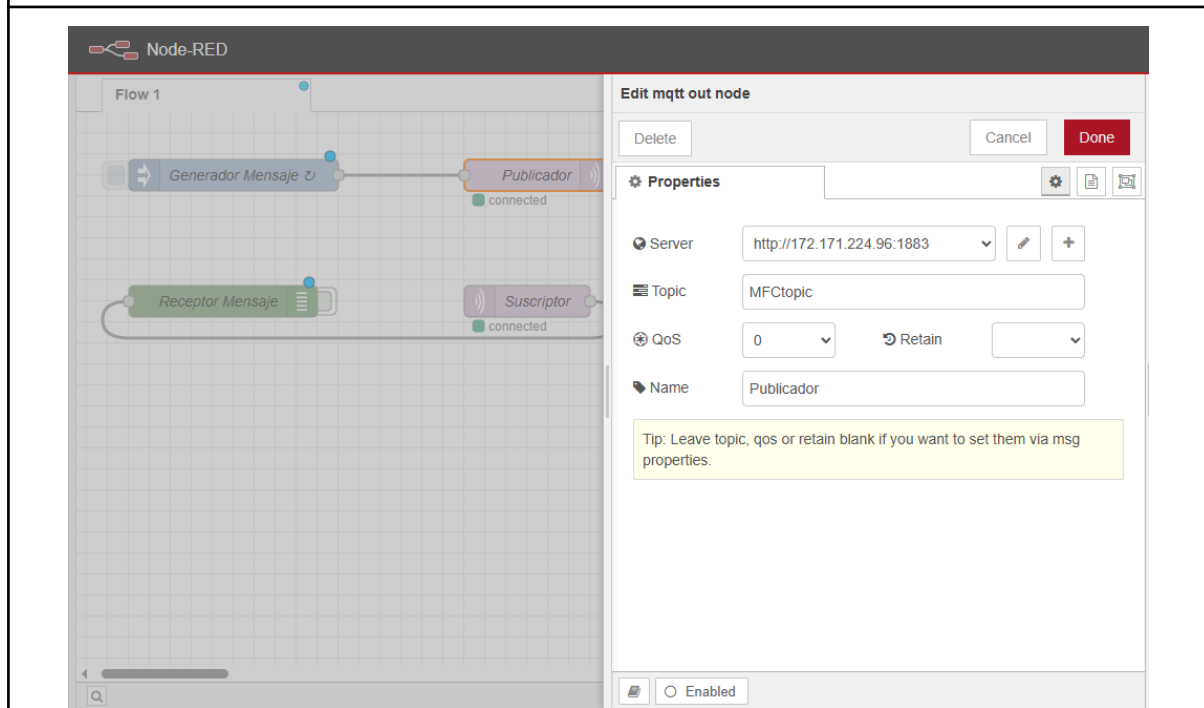
Muestre abajo el resultado de la implementación del “Receptor Mensaje” del mensaje.



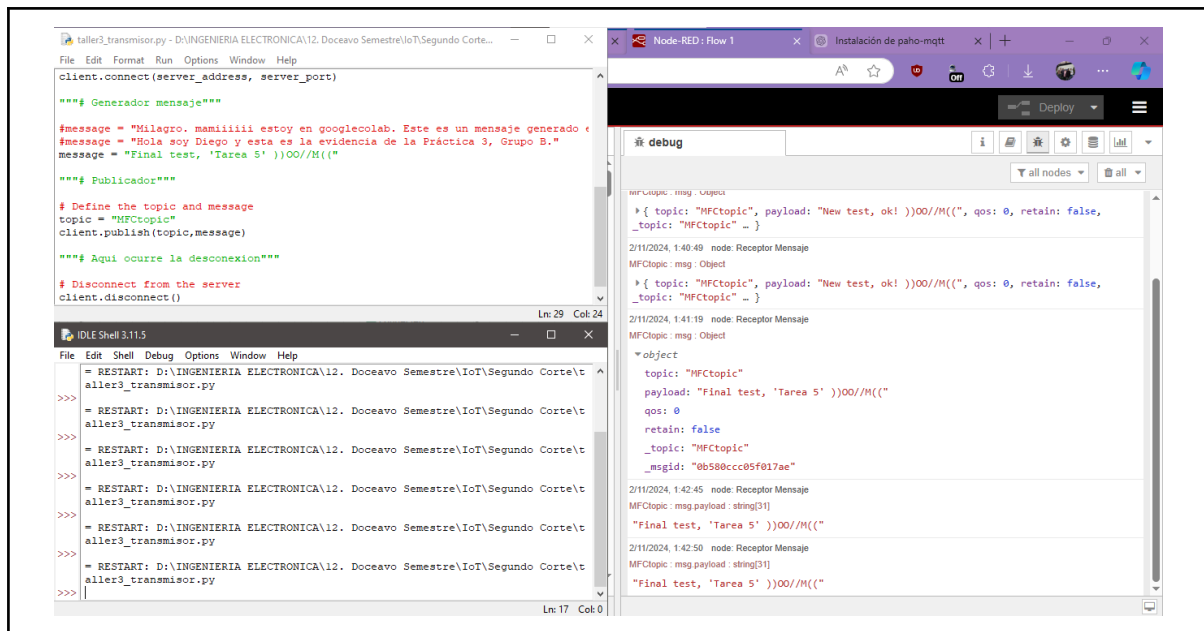
Muestre una comparación entre el mensaje en el “generador del mensaje” y lo que se obtiene en el Receptor en Google Colab



Muestre abajo el resultado de la implementación del publicador del mensaje.




Muestre una comparación entre el mensaje en el “generador del mensaje” y lo que se obtiene en Node-RED

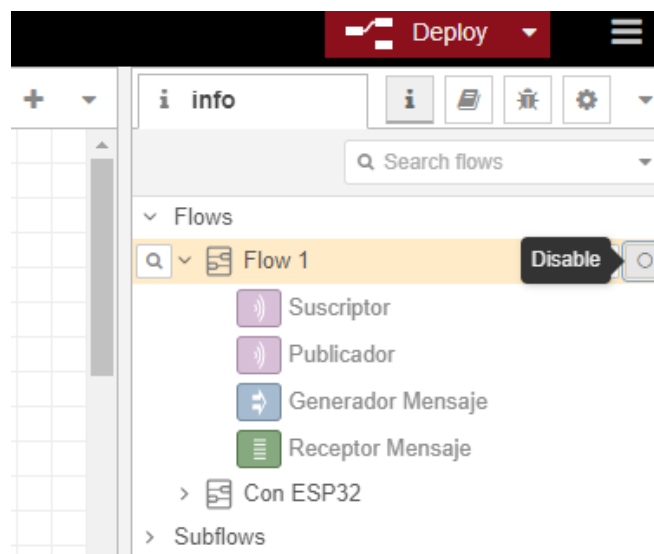


Tarea 6. Conquista del sistema de manejo de flujo-nodos y archivos

Paso 1. Probar lo más básico

- Analiza si se conservan los cambios de un flujo-nodo, cuando no oprimes Deploy. Para esto, haz unos cambios al flujo-nodo, copia la URL de tu trabajo, pasa a otra ventana del navegador, pega la URL y mira si observas los nuevos cambios o solo lo que había antes de un comando Deploy.
- Descubra cómo cambiarle el nombre a los nodos (flujo-nodos)
- Pruebe ocultar un flujo-nodo y luego hacerlo aparecer nuevamente
- Pruebe desactivar/activar uno de los flujo-nodos así:

Menu lateral derecho >  seleccionas en el panel izquierdo el flujograma > disable



El resultado es que el flujograma aparece punteado



Paso 2. Aprender a guardar de manera segura los trabajos hechos en Node-RED



Supongamos que tienes un flujograma que no vas a usar en el momento, pero que deseas conservar para usos futuros. ¿Cómo lo guardarías? ¿Cómo llamarías un flujograma guardado?. Responder estas preguntas es muy importante ya que tenemos amargas experiencias de principiantes que han realizado un fuerte trabajo, pero luego, por alguna razón lo pierden todo.

Realiza la siguiente práctica

- Guarde en su computador el flujograma usado. Menú principal > Export > Seleccionas el flujograma de interés > Download. Nota: es más profesional guardarlo en un repositorio como GitHub.
- Crea un flujo nodo de pruebas. Borra del Node-RED el flujo nodo (OJO porque estas instrucciones borran sin advertir el flujograma donde estés parado): Te paras en el flujograma de interés > Menú principal > Flows > Delete
- Recupere en Node-RED el flujograma a partir del archivo guardado en el computador Menú principal > Import > New flow > Select a file to import > seleccionas el archivo de interés (observa que existe la opción de subir un archivo de tu computador (Clipboard), de tu VM (localhost), ejemplos que Node-RED ofrecer (Examples))> primer Import > Import copy

Paso 3. Conquista de Menús de consulta

- Compruebe qué se aprecia con: Menú lateral derecho >
- Compruebe qué se aprecia con: Menú lateral derecho >
- Compruebe si las ayudas se adaptan al bloque que se selecciona: Resalte uno de los nodos del flujo-nodo > Menú lateral derecho >
- Crea un nuevo flujo-nodo sin modificar el anterior, ponlo a correr y analiza si los dos flujo-nodos quedan corriendo o solo uno

- Compruebe la posibilidad de filtrar datos con  >  por ejemplo para mostrar sólo datos de un flujo-nodo o de varios.

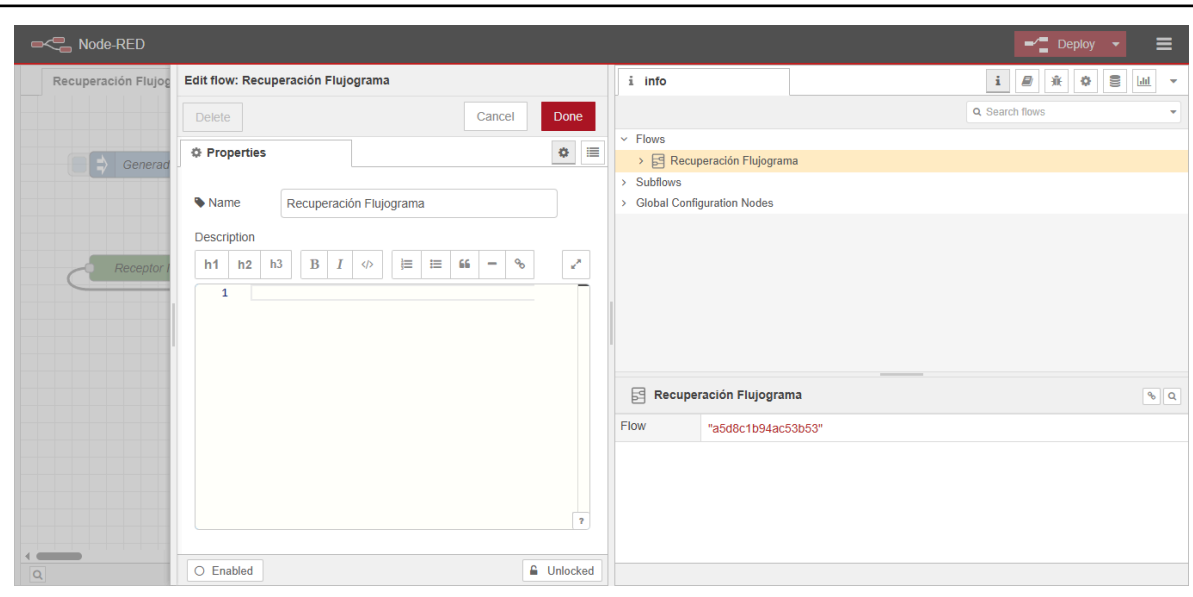
Paso 4. Otras conquistas

- Analice si otras personas puede editar su flujo-nodo, desde otro navegador

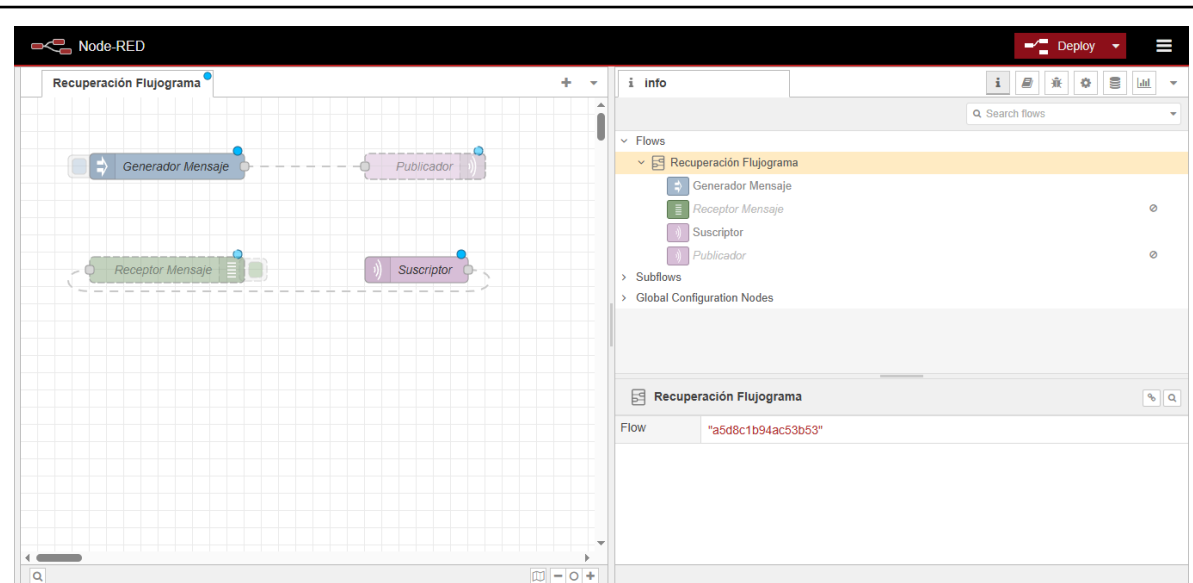
Paso 5. Llene la tabla que va como informe

Para el informe. Tarea 6. Tabla de resultados de la conquista de herramientas de Node-RED

Captura de pantalla que demuestra que has podido cambiarle el nombre a un flujo-nodo

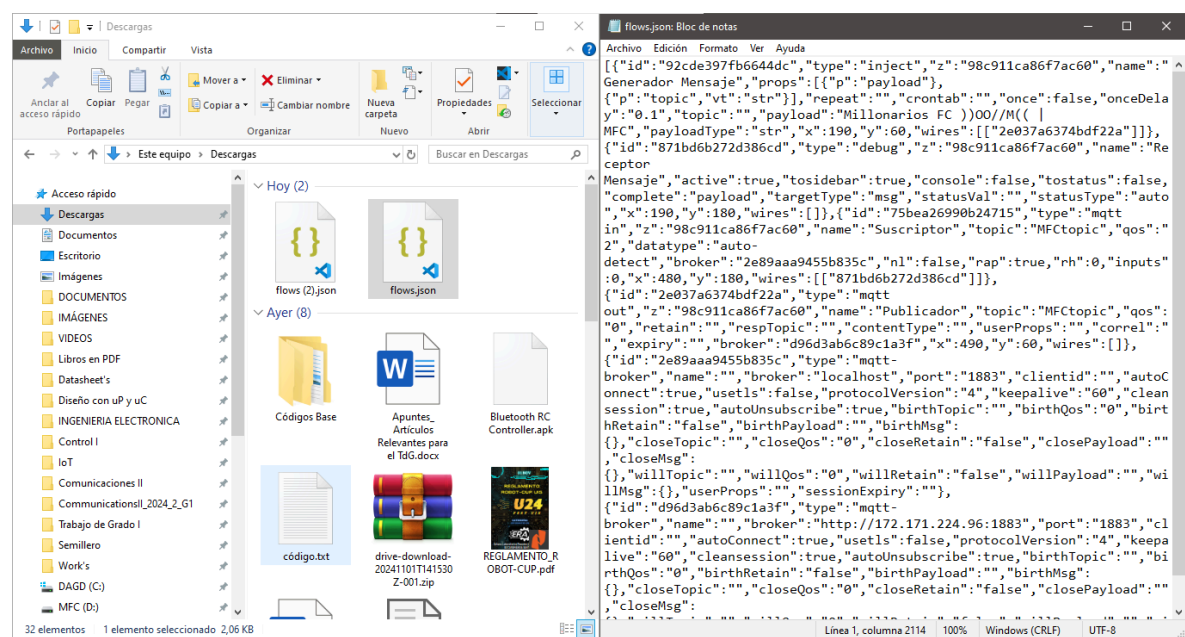


Captura de pantalla que demuestra que has podido desactivar un flujo-nodo

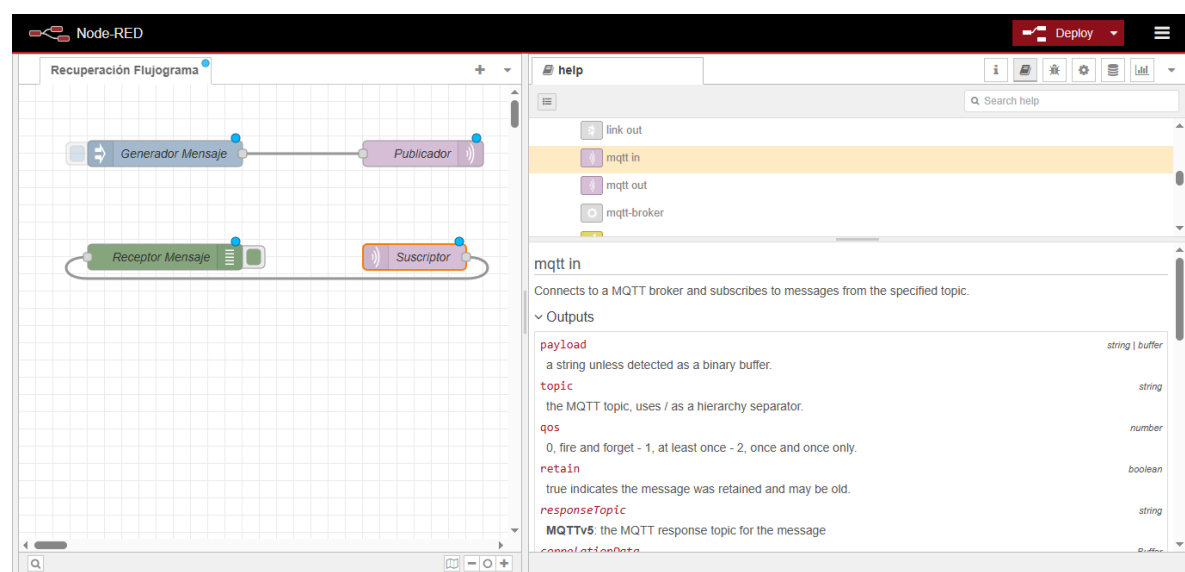


Pantallazo que muestra como es el archivo de un flujo-nodo cuando lo has bajado a tu

computador



Muestra un pantallazo para demostrar que puedes ver las ayudas de un nodo seleccionado



Explique si, con la configuración que tiene actualmente Node-RED, otros usuarios podrían, desde otro navegador modificar tu flujo-nodo.

Si, otros usuarios pueden acceder desde otro navegador si se tiene acceso a la VM, la IP de la misma y el puerto de Node-RED. Se recomienda habilitar una autenticación y permisos de usuario en un archivo settings.js, esto con el fin de que alguien pueda acceder se deben cumplir determinadas condiciones:

1. **Acceso a la Red:** Si Node-RED se está ejecutando en una VM o servidor accesible públicamente (como una VM en Azure) y el puerto adecuado (en este caso, el 1880) está abierto, otros usuarios pueden acceder a la interfaz de Node-RED ingresando la dirección IP y el puerto en un navegador web.

2. **Autenticación y Seguridad:** Por defecto, Node-RED no tiene activada la autenticación, lo que significa que cualquier persona que conozca la dirección IP y el puerto de Node-RED podría ver y modificar los flujos. Para proteger el acceso, se recomienda configurar el archivo *settings.js* para habilitar usuarios y contraseñas, permitiendo así un control sobre quién puede ver o editar los flujos en Node-RED.
3. **Permisos de Edición:** Aunque otros usuarios tengan acceso a la interfaz de Node-RED, necesitarán permisos específicos para realizar ediciones en los flujos. Al habilitar la autenticación, se puede restringir la capacidad de edición a usuarios autorizados, protegiendo los flujos contra cambios no deseados.