INFORME - PROYECTO FINAL
INTERNET DE LAS COSAS

Bustos Mendez Jorge Fernando
Burbano Danny Alexander
Diaz Anama Jonatan Nemesio
Narvaez Arturo Nicolas

Universidad de Nariño

2024

Resumen

En este informe se presenta la implementación de un sistema de transmisión de datos basado en el microcontrolador ESP32 y el servidor Portenta X8. Utilizando Docker como plataforma de virtualización, se configuró un entorno para ejecutar Mosquitto, un broker MQTT que facilita la comunicación entre dispositivos IoT. El ESP32 actúa como nodo emisor, enviando datos en tiempo real al servidor Portenta X8, que los recibe y procesa mediante el protocolo MQTT. Este desarrollo destaca por su modularidad y escalabilidad, demostrando un modelo eficiente para la conectividad y transmisión de datos en entornos IoT.

1. Introducción

En la actualidad, el Internet de las Cosas (IoT, por sus siglas en inglés) se ha consolidado como una de las principales tecnologías para impulsar la automatización, la conectividad y la eficiencia en diversos sectores. La comunicación entre dispositivos IoT requiere de sistemas que garanticen la transmisión de datos de manera confiable, rápida y eficiente, factores cruciales en aplicaciones que involucran desde la domótica hasta la industria 4.0. En este marco, el presente trabajo se centra en el desarrollo e implementación de un sistema de transmisión de datos entre un microcontrolador ESP32 y un servidor Portenta X8, utilizando tecnologías avanzadas como Docker y el protocolo MQTT.

El ESP32, ampliamente utilizado en proyectos IoT por su capacidad de procesamiento y conectividad inalámbrica, se configuró como un nodo que recolecta y envía datos. Por su parte, el Portenta X8, un dispositivo versátil con capacidad para ejecutar múltiples contenedores de software, actúa como un servidor que recibe, organiza y procesa la información. Para gestionar esta comunicación, se utilizó Docker, una plataforma que permite la virtualización ligera de aplicaciones, configurándose un entorno optimizado para ejecutar Mosquitto, un broker MQTT.

El protocolo MQTT fue seleccionado por sus características de ligereza y alta eficiencia, especialmente diseñadas para entornos donde los recursos de red son limitados y la latencia debe ser mínima. Esta elección permitió establecer una comunicación robusta entre los dispositivos, asegurando la transmisión de datos en tiempo real con un consumo mínimo de ancho de banda.

A lo largo del documento se describen en detalle los pasos seguidos para la configuración y la implementación del sistema, desde la instalación y configuración del entorno Docker en el Portenta X8, hasta la programación del ESP32 para enviar datos mediante MQTT. También se analizan los resultados obtenidos, destacando los beneficios de esta integración tecnológica, como la modularidad, escalabilidad y eficiencia energética, así como sus posibles aplicaciones en sistemas conectados más amplios. El desarrollo presentado en este informe constituye un ejemplo práctico de cómo integrar hardware y software para crear soluciones IoT modernas y escalables. Este enfoque no solo valida el potencial de estas tecnologías, sino que también abre camino para futuras implementaciones en entornos donde la conectividad y la automatización son esenciales.

2. Marco Teórico

2.1. MQTT (Message Queuing Telemetry Transport)

El protocolo MQTT (Message Queuing Telemetry Transport) es un estándar abierto y ligero de mensajería, diseñado para comunicaciones eficientes en

redes con limitados recursos de ancho de banda o alto nivel de latencia, características comunes en el ámbito del Internet de las Cosas (IoT). MQTT sigue una arquitectura de publicación/suscripción, donde los dispositivos (conocidos como clientes) envían ("publican") mensajes a un broker MQTT y otros dispositivos se suscriben a los topics o temas de interés para recibir esos mensajes. La simplicidad de este protocolo y su bajo consumo de recursos lo hacen ideal para entornos con dispositivos de bajo poder de procesamiento y redes con limitaciones, como es el caso del ESP32, y dispositivos más potentes como el Portenta X8.

Este protocolo se basa en un modelo de publicación-suscripción, en el cual los dispositivos (çlientes") envían mensajes a un servidor centralizado llamado "broker". Los clientes se suscriben a "topics"(temas) específicos y reciben solo los mensajes relacionados. Esto minimiza el uso de datos y facilita la comunicación asíncrona entre múltiples dispositivos.
El uso de MQTT con Mosquitto en el Portenta X8 permite un flujo de datos eficiente y fiable entre dispositivos IoT, manteniendo la flexibilidad en la implementación y la posibilidad de integrar diversos sensores, actuadores y dispositivos adicionales. Esto facilita la creación de sistemas IoT más inteligentes y adaptables, al permitir que los datos sean centralizados y visualizados de forma eficiente en plataformas como Grafana, mientras que el Portenta X8 gestiona toda la infraestructura necesaria para soportar las conexiones y el procesamiento de información.

2.2. ***Grafana***

Grafana es una herramienta de código abierto diseñada para la visualización, monitoreo y análisis de datos en tiempo real provenientes de diversas fuentes. Su capacidad para integrar múltiples data sources, como bases de datos, servidores y protocolos como MQTT, la convierte en una solución versátil en proyectos de IoT. Grafana permite crear paneles personalizados donde los datos se presentan de manera gráfica e intuitiva, lo que facilita la comprensión y toma de decisiones basadas en información procesada. En el contexto de dispositivos como el Portenta X8, Grafana se destaca al ofrecer una interfaz visual potente que complementa la transmisión de datos gestionada por contenedores como Mosquitto, brindando a los usuarios una herramienta robusta para monitorear sistemas IoT de forma eficiente y accesible.

2.3. ***Docker***

Docker es una plataforma de contenedorización que permite desarrollar, distribuir y ejecutar aplicaciones en entornos aislados, conocidos como contenedores. Estos encapsulan el código y todas las dependencias necesarias para garantizar que las aplicaciones funcionen de manera consistente en cualquier entorno. A diferencia de las máquinas virtuales, los contenedores son ligeros, ya que comparten el núcleo del sistema operativo del host, lo que permite un uso eficiente de recursos y un inicio rápido. Las imágenes Docker, que actúan como plantillas para los contenedores, contienen configuraciones predefinidas y facilitan la portabilidad y la modularidad en el desarrollo y despliegue de software. La plataforma es impulsada por Docker Engine, que utiliza tecnologías como namespaces y cgroups para garantizar el aislamiento. Docker fomenta un enfoque basado en microservicios, dividiendo aplicaciones en componentes independientes que se ejecutan en contenedores separados, lo que facilita el escalado y el mantenimiento. Además, se integra con herramientas como Kubernetes para gestionar múltiples contenedores en clústeres. Amplia y versátil, Docker es una herramienta clave en proyectos de desarrollo, pruebas y despliegue, siendo especialmente útil en IoT para gestionar aplicaciones en dispositivos de bajo consumo.

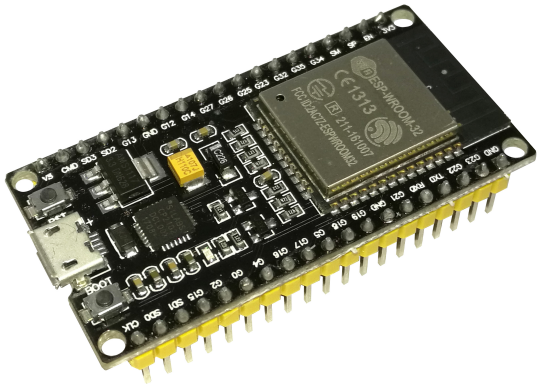
Un contenedor de Docker es una unidad de software que encapsula una aplicación y todas sus dependencias necesarias para ejecutarla de manera

uniforme en cualquier entorno. Este encapsulamiento incluye bibliotecas, configuraciones, herramientas de sistema y código, permitiendo que la aplicación se ejecute de forma aislada y sin conflictos con otras aplicaciones en el mismo sistema. Los contenedores son ligeros, ya que comparten el núcleo del sistema operativo del host en lugar de incluir un sistema operativo completo, lo que los hace más eficientes que las máquinas virtuales.

En el caso específico de Mosquitto, que es un broker MQTT ampliamente utilizado, un contenedor Docker ofrece un entorno optimizado y preconfigurado para ejecutar Mosquitto sin necesidad de instalar manualmente todas sus dependencias. Mosquitto, como broker MQTT, actúa como intermediario entre dispositivos IoT que envían y reciben mensajes basados en el protocolo MQTT, conocido por su eficiencia y bajo consumo de ancho de banda.

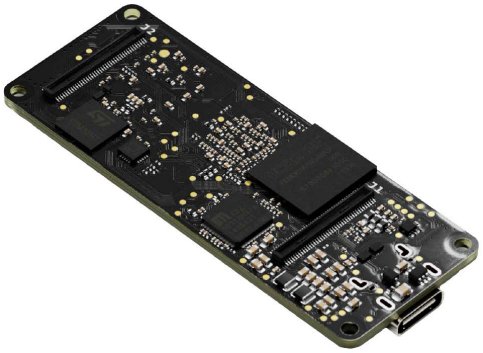
2.4. ***ESP32***

El ESP32 es un microcontrolador de 32 bits desarrollado por Espressif, ampliamente utilizado en aplicaciones IoT gracias a sus capacidades de conectividad Wi-Fi y Bluetooth, así como a su eficiencia energética. Este módulo es ideal para aplicaciones de transmisión de datos en la nube, como el envío y recepción de datos a AWS IoT mediante MQTT.

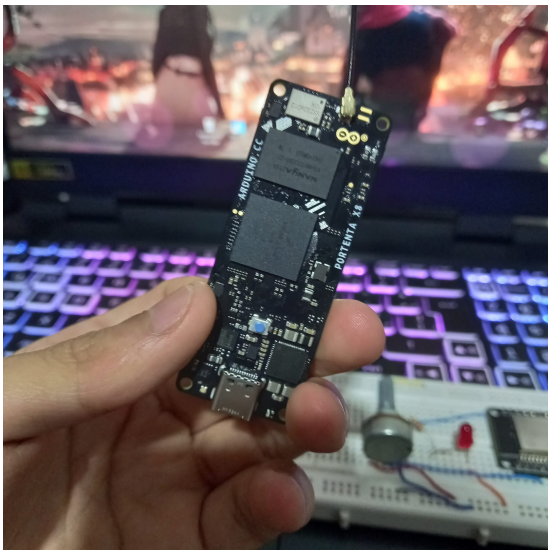


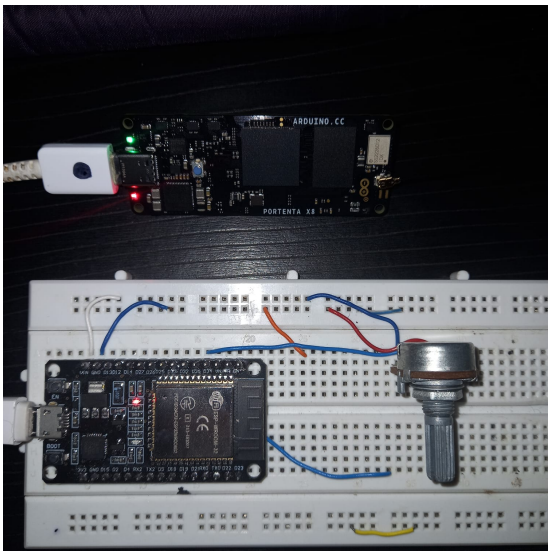
2.5. ***Portenta x8***

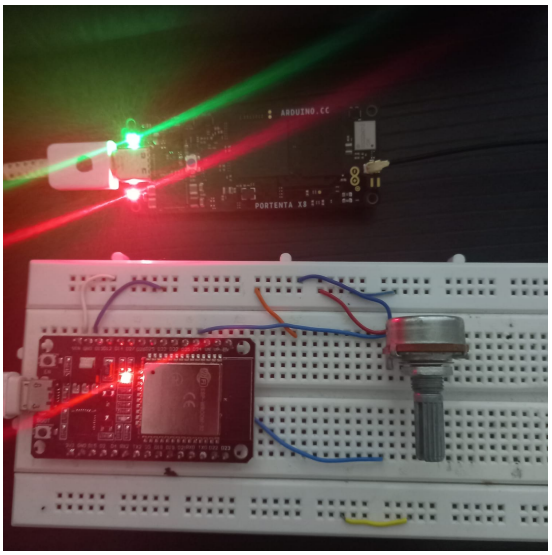
El Portenta X8 es una placa de desarrollo avanzada de Arduino diseñada específicamente para aplicaciones IoT y escenarios industriales. Su arquitectura híbrida combina núcleos ARM Cortex-A53, para ejecutar sistemas operativos como Linux, con núcleos Cortex-M7 y Cortex-M4, ideales para tareas en tiempo real y de bajo consumo. Esta versatilidad permite que la placa actúe como un potente gateway IoT capaz de gestionar la comunicación entre dispositivos, procesar datos localmente (edge computing) y ejecutar aplicaciones en contenedores mediante Docker, como brokers MQTT o bases de datos. Su conectividad Wi-Fi, Bluetooth y Ethernet, junto con características de seguridad avanzada, la hacen ideal para redes IoT confiables y escalables, mientras que su diseño robusto y capacidad de integración con el ecosistema Arduino facilitan su implementación en proyectos de automatización, monitoreo industrial y gestión de dispositivos inteligentes.



3. Desarrollo







3.1. *Configuracion de la board*

El primer paso para trabajar con el Portenta X8 fue configurar el entorno de desarrollo en el Arduino IDE, comenzando por instalar la versión más reciente del software para garantizar la compatibilidad con la placa. Dentro del IDE, se accedió al gestor de placas mediante el menú Herramientas ¿Placa ¿Gestor de Placas, donde se buscó y descargó el paquete .ªrduino Mbed OS Portenta Boards"necesario para habilitar su uso. Una vez instalado, se seleccionó la Portenta X8 desde el menú Herramientas ¿Placa ¿Arduino Portenta X8 y se conectó al sistema utilizando un cable USB-C, configurando el puerto adecuado en Herramientas ¿Puerto. Finalmente, se cargó un sketch de prueba, como el clásico "Blink", para verificar la correcta comunicación entre la board y el IDE, asegurando que el entorno estuviera listo para el desarrollo de aplicaciones IoT avanzadas.

3.1.1. *Reconocimiento y configuracion del Portenta x8*

Una vez instalado el paquete de la board en el Arduino IDE, el siguiente paso fue abrir Windows PowerShell y acceder a la ubicación correspondiente mediante el comando:

```
cd "C:\Users\J Ferb\AppData\Local\
    Arduino15\packages\arduino\tools\adb
    \32.0.0"
```

Una vez en este directorio, se introdujo el comando adb devices, el cual permite verificar si el sistema está reconociendo correctamente el dispositivo Portenta X8. Este comando devuelve una lista de dispositivos conectados, lo que confirma si la conexión entre el ordenador y la board ha sido establecida de forma adecuada.

3.1.2. *Configuracion del entorno Linux del Portenta x8*

Una vez confirmado que el dispositivo Portenta X8 ha sido reconocido correctamente, el siguiente paso fue configurar el entorno de Linux en la board. Para ello, en la misma ubicación de PowerShell, se introdujo el siguiente comando:

```
.\adb shell
```

Este comando permite acceder al sistema operativo Linux del Portenta X8. Al ingresar al entorno de Linux, se procede a cambiar la contraseña del usuario para poder acceder a los permisos necesarios y gestionar la configuración del dispositivo de manera adecuada. Este paso es esencial para asegurar el acceso y control total sobre el sistema operativo del Portenta X8, facilitando la ejecución de tareas administrativas y el despliegue de aplicaciones.

3.1.3. *Conexion y configuracion WIFI e IP*

A continuación, se procedió a conectar el Portenta X8 a una red Wi-Fi para poder acceder de forma remota al dispositivo y facilitar la gestión de los contenedores y la conexión SSH. Para ello, se configuró la conexión inalámbrica directamente desde el sistema operativo Linux del Portenta X8, utilizando los comandos adecuados para establecer la red Wi-Fi, como:

```
nmcli dev wifi list
nmcli dev wifi connect "SSID" password "
    contrase a"
nmcli connection show
```

de esta forma verificamos las redes disponibles, luego conectamos a la red, y verificamos el estado de la conexion.
Una vez conectados, se verificó que el dispositivo obtuviera una dirección IP válida mediante el siguiente comando:

```
ip addr show
ifconfig
```

cualquiera de estos dos comandos es valido para obtener la IP

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.11  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::eea6:3a3b:e92f:2892  prefixlen 64  scopeid 0x20<link>
        ether 04:c4:61:d5:bf:e2  txqueuelen 1000  (Ethernet)
        RX packets 55  bytes 3936 (3.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 65  bytes 9261 (9.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Esto asegura que el dispositivo esté correctamente conectado a la red. Con la conexión establecida, se procedió a salir del entorno bash o del sistema operativo del Portenta X8 y volver a PowerShell. Desde allí, se procedió a acceder de manera remota al Portenta X8 utilizando SSH, mediante el siguiente comando:

```
ssh fio@"ip"
```

Al intentar conectarse por SSH, el sistema solicitará la contraseña que fue configurada al principio del proceso para poder acceder al dispositivo. Esta conexión no solo permite la gestión de la dirección IP, sino también el acceso SSH, lo que facilita la administración de contenedores y la ejecución de tareas de manera eficiente desde cualquier terminal de la red.

3.2. Configuraciones de Docker

Una vez establecido el acceso SSH y confirmada la conexión del Portenta X8 a la red, el siguiente paso fue configurar los entornos de Docker y sus componentes necesarios. Este proceso incluiría la instalación de Docker, Mosquitto y Grafana. Sin embargo, durante esta configuración, se observó un aspecto importante: el Portenta X8 desactiva los contenedores utilizados cada vez que se reinicia o se desconecta y vuelve a conectarse a una red. Dado que no se manejan direcciones IP estáticas, sino dinámicas, es necesario actualizar constantemente las configuraciones de red, ya que cada vez que el dispositivo se conecta a una red, su dirección IP puede cambiar. Por lo tanto, tanto la configuración del entorno Docker como la conexión SSH deberán repetirse cada vez que el dispositivo se reinicie o se conecte a una nueva red, asegurando que los contenedores se activen correctamente y las direcciones IP se actualicen en los códigos correspondientes.
Antes de configurar los contenedores individuales, es importante verificar qué contenedores están activos y cuáles están instalados en el Portenta X8. Para ello, se pueden ejecutar los siguientes comandos para obtener el estado de los contenedores:
Para ver los contenedores activos:

```
docker ps
```

```
fio@portenta-x8-10111209dabc240b:~$ docker ps
CONTAINER ID   IMAGE                                                    COMMAND
  CREATED       STATUS         PORTS                                    NAMES
1bd5feec4d16   hub.foundries.io/arduino/arduino-ootb-python-devel       "/entrypoint.sh"
  2 years ago   Up 4 minutes                                            x8-devel
e3603f1c3906   hub.foundries.io/arduino/arduino-iot-cloud-provisioning  "/entrypoint.sh"
  2 years ago   Up 4 minutes                                            x8-provisioning
1cf08a8ee276   hub.foundries.io/arduino/arduino-ootb-webapp             "/entrypoint.sh"
  2 years ago   Up 4 minutes   0.0.0.0:80->1323/tcp, :::80->1323/tcp    x8-webapp
```

Para ver los contenedores instalados:

```
docker ps -a
```

```
fio@portenta-x8-10111209dabc240b:~$ docker ps -a
CONTAINER ID   IMAGE                                                    COMMAND
       CREATED         STATUS                       PORTS
      NAMES
e062c9667ae3   eclipse-mosquitto                                        "/docker-entrypo
int.…"   18 hours ago   Exited (255) 18 hours ago   0.0.0.0:1883->1883/tcp, :::1883->1883
/tcp   mosquitto
555906ecc131   grafana/grafana                                          "/run.sh"
       18 hours ago   Exited (255) 18 hours ago   0.0.0.0:3000->3000/tcp, :::3000->3000
/tcp   grafana
6eadf402d1fc   influxdb:latest                                          "/entrypoint.sh
inf1…"   24 hours ago   Exited (255) 23 hours ago   0.0.0.0:8086->8086/tcp, :::8086->8086
/tcp   influxdb
bda62c49708a   telegraf                                                 "/entrypoint.sh
tele…"   24 hours ago   Exited (255) 23 hours ago   8092/udp, 8125/udp, 8094/tcp
      telegraf
1bd5feec4d16   hub.foundries.io/arduino/arduino-ootb-python-devel       "/entrypoint.sh"
       2 years ago    Up 5 minutes
      x8-devel
e3603f1c3906   hub.foundries.io/arduino/arduino-iot-cloud-provisioning   "/entrypoint.sh"
       2 years ago    Up 5 minutes
      x8-provisioning
1cf08a8ee276   hub.foundries.io/arduino/arduino-ootb-webapp              "/entrypoint.sh"
       2 years ago    Up 5 minutes                 0.0.0.0:80->1323/tcp, :::80->1323/tcp
      x8-webapp
```

3.2.1. Configuracion del Container Mosquitto

Una vez obtenida esta información, pasamos a trabajar específicamente con el contenedor Mosquitto. Para gestionar este contenedor, utilizamos los siguientes comandos para detenerlo, reiniciarlo y configurar su primer puerto de escucha para habilitarlo correctamente:
Para detener el contenedor Mosquitto:

```
docker stop mosquitto
```

Para reiniciar el contenedor Mosquitto:

```
docker rm mosquitto
```

Para configurar el primer puerto de escucha y habilitar el contenedor:

```
docker run -d --name mosquitto -p
    1883:1883 eclipse-mosquitto
```

Adicionalmente, el contenedor Mosquitto presenta otro problema relacionado con la autenticación, ya que por defecto restringe el acceso solo a usuarios autorizados. Para permitir que dispositivos desconocidos, como el ESP32, puedan conectarse y comunicarse a través de Mosquitto, es necesario modificar su configuración para habilitar el acceso sin autenticación. Para ello, se deben ejecutar los siguientes comandos para activar Mosquitto en modo de uso libre para usuarios desconocidos:
Para permitir el acceso sin autenticación:

```
sudo docker exec -it mosquitto sh
echo -e "listener 1883 0.0.0.0\
    nallow_anonymous true" > /mosquitto/
    config/mosquitto.conf
```

y ya por fuera del contenedor

```
sudo docker restart mosquitto
docker logs mosquitto
```

```
fio@portenta-x8-10111209dabc240b:~$ docker logs mosquitto
1733259311: mosquitto version 2.0.20 starting
1733259311: Config loaded from /mosquitto/config/mosquitto.conf.
1733259311: Starting in local only mode. Connections will only be possible from clients r
unning on this machine.
1733259311: Create a configuration file which defines a listener to allow remote access.
1733259311: For more details see https://mosquitto.org/documentation/authentication-metho
ds/
1733259311: Opening ipv4 listen socket on port 1883.
1733259311: Opening ipv6 listen socket on port 1883.
1733259311: Error: Address not available
1733259311: mosquitto version 2.0.20 running
1733259342: mosquitto version 2.0.20 terminating
1733259343: mosquitto version 2.0.20 starting
1733259343: Config loaded from /mosquitto/config/mosquitto.conf.
1733259343: Opening ipv4 listen socket on port 1883.
1733259343: mosquitto version 2.0.20 running
1733259345: New connection from 192.168.1.10:57831 on port 1883.
1733259345: New client connected from 192.168.1.10:57831 as ESP32Client (p2, c1, k15).
```

3.2.2. *Configuracion Container Grafana*

El proceso para gestionar el contenedor Grafana es más sencillo en comparación con el de Mosquitto, ya que no requiere modificaciones complejas en su configuración. Para trabajar con este contenedor, simplemente se debe detener, reiniciar y verificar su funcionamiento. Primero, si es necesario, se detiene el contenedor utilizando el comando adecuado. Luego, se reinicia para asegurar que cualquier cambio o actualización se aplique correctamente. Finalmente, se puede verificar que el contenedor esté funcionando correctamente con el comando para listar los contenedores activos, y si se desea, revisar los logs del contenedor para comprobar que no existan errores y que todo esté en orden. Este proceso asegura que Grafana esté operativo y accesible para su uso en el entorno.

```
docker stop grafana
docker rm grafana
docker run -d --name=grafana -p 3000:3000
    grafana/grafana
```

```
fio@portenta-x8-10111209dabc240b:~$ docker stop grafana
grafana
fio@portenta-x8-10111209dabc240b:~$ docker rm grafana
grafana
fio@portenta-x8-10111209dabc240b:~$ docker run -d --name=grafana -p 3000:3000 grafana/gra
fana
f669444828b64dc6b7229c2480a839864ef032612a47e3138ef8ee145240dc8b
```

3.2.3. *Creación del codigo esp32 y conexion mqtt*

Con los contenedores ya activos y listos para su uso, pasamos a la creación del código para el ESP32. En este paso, implementamos el siguiente código, en el cual primero establecemos la conexión a Internet. Luego, configuramos el protocolo MQTT utilizando las credenciales del Portenta X8, creando un topic en el que el ESP32 podrá publicar datos. Es importante destacar que el listener por defecto utilizado por Mosquitto es el puerto 1883, lo cual debe ser configurado en ambos dispositivos para asegurar la comunicación correcta. Posteriormente, será necesario que el Portenta X8 se suscriba a este topic para recibir los datos enviados por el ESP32. Finalmente, estos datos podrán ser visualizados en Grafana para su monitoreo y análisis en tiempo real.

```
#include <WiFi.h>
#include <PubSubClient.h>

// Configuraci n de red WiFi
const char* ssid = "FLIA BUSTOS";
    // Nombre de la red WiFi
const char* password = "Jydli12345"; //
    Contrase a de la red WiFi

// Configuraci n del broker MQTT
const char* mqtt_server = "192.168.1.11";
    // Direcci n IP del broker MQTT
const int mqtt_port = 1883;
    // Puerto del broker MQTT (por defecto
     1883)
const char* mqtt_topic = "potentiometer";
     // Tema de MQTT al que se publicar n
     los datos

WiFiClient espClient;
PubSubClient client(espClient);

// Configuraci n del pin del
    potenci metro
const int potentiometerPin = 34; // Pin
    anal gico para el potenci metro (
    cambiar seg n el pin utilizado)

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  // Conectar a la red WiFi
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...")
    ;
  }
  Serial.println("Conectado a la red WiFi"
    );

  // Configurar el cliente MQTT
  client.setServer(mqtt_server, mqtt_port)
    ;
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Leer el valor del potenci metro
  int potentiometerValue = analogRead(
    potentiometerPin);

  // Enviar el valor del potenci metro al
     broker MQTT
  char msg[50];
  snprintf(msg, 50, "%d",
    potentiometerValue);
  client.publish(mqtt_topic, msg);

  delay(1000); // Enviar cada segundo
}

// Funci n para reconectar al broker MQTT
void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando conectar al
    broker MQTT...");
    if (client.connect("ESP32Client")) {
```

```
      Serial.println("Conectado al broker MQTT");
    } else {
      Serial.print("Fall , reintentando en 5 segundos");
      delay(5000);
    }
  }
}
```

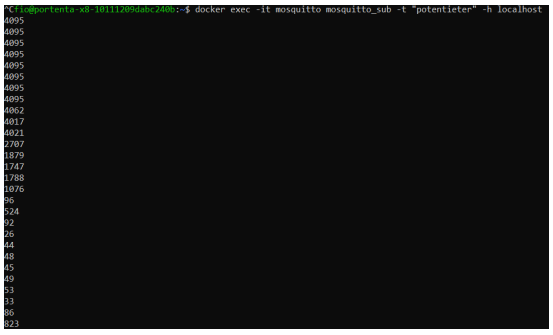
Una vez configurado el código y cargado en el ESP32, es hora de regresar al Portenta X8 y suscribirnos al topic que hemos creado en el ESP32. Para lograr esto, utilizaremos el siguiente comando:

```
docker exec -it mosquitto mosquitto_sub -t "#" -h localhost
```

Este comando nos permitirá suscribirnos a todos los topics disponibles en el canal MQTT del Portenta X8. Sin embargo, si queremos suscribirnos a un topic en particular, podemos utilizar el siguiente comando, que es más específico para un topic determinado:

```
docker exec -it mosquitto mosquitto_sub -t "potentiometer" -h localhost
```

se pone "potentiometerçomo en el codigo del esp32, el cual fue el nombre del topico creado.

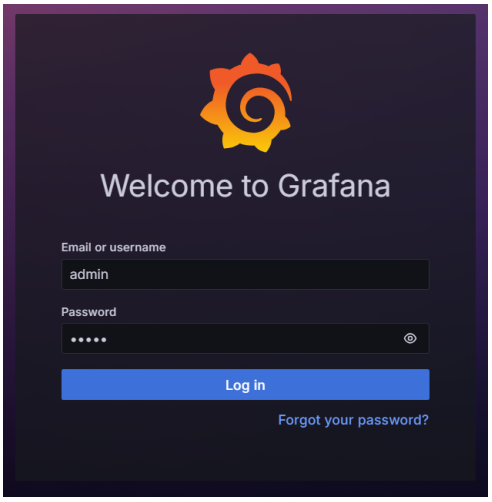


3.3. Visualizacion en Grafana

Una vez suscritos y con acceso a los datos enviados por el ESP32 a través del Portenta X8, es hora de abrir Grafana para realizar las configuraciones finales y poder visualizar estos datos. El primer paso es iniciar nuestro navegador de preferencia y acceder a la interfaz de Grafana escribiendo en la URL:

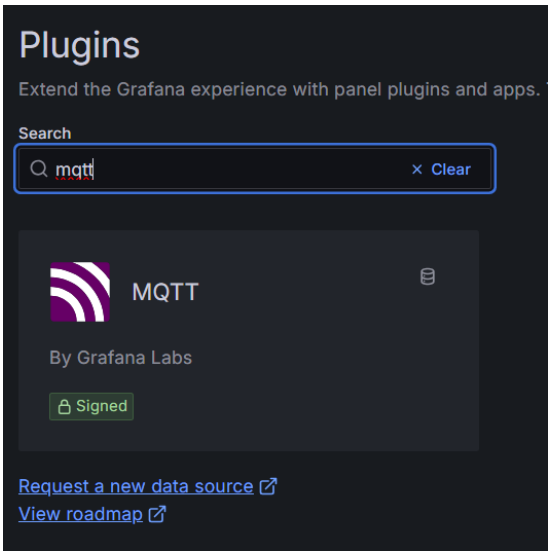
```
http://"IP del Portenta X8":3000
```

Al ingresar, se nos pedirá configurar un usuario y contraseña para proteger el acceso y garantizar la seguridad de nuestros datos. Esta configuración inicial es fundamental para asegurar que solo los usuarios autorizados puedan interactuar con la plataforma y acceder a la información sensible.



3.3.1. Configuracion de plugins, data sources y lanzamiento

Una vez en este punto, estamos muy cerca de finalizar el proceso. Ahora, procedemos a configurar los plugins dentro de Grafana, comenzando por buscar e instalar el plugin MQTT.

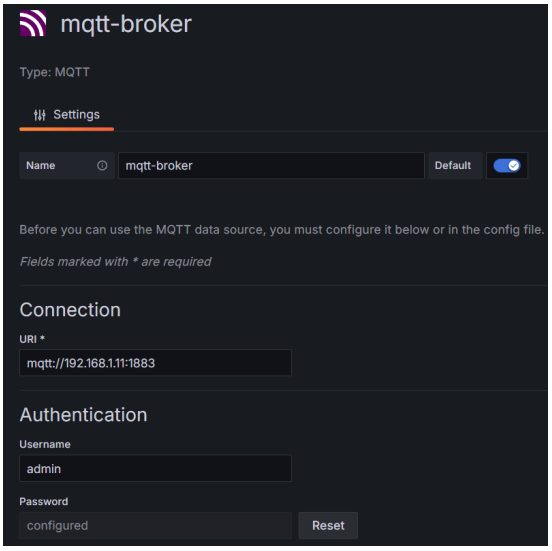


Cabe aclarar que este es un proceso que debe repetirse cada vez que el Portenta X8 se apague y vuelva a encender. Además, la visualización de los

datos será solo accesible en la red local. Tras haber instalado el plugin, avanzamos a la configuración del Data Source. En esta sección, seleccionamos MQTT como la fuente de datos y configuramos la URI de la siguiente manera:

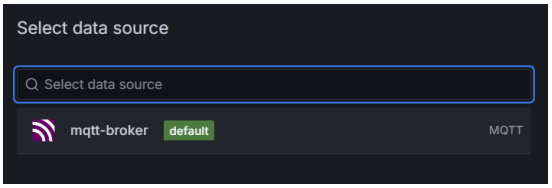
```
mqtt://"IP del Portenta X8":1883
```

A continuación, configuramos la autenticación utilizando el usuario y contraseña que creamos al primer ingreso en Grafana. Finalmente, hacemos clic en guardar y con eso, nuestra configuración estará completa, permitiéndonos visualizar los datos enviados desde el ESP32 en Grafana.

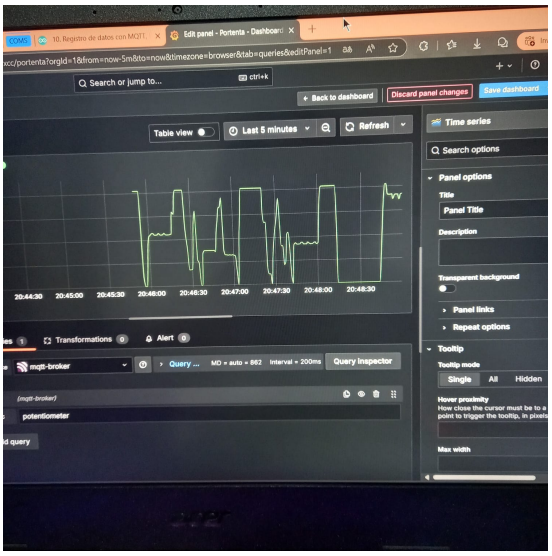


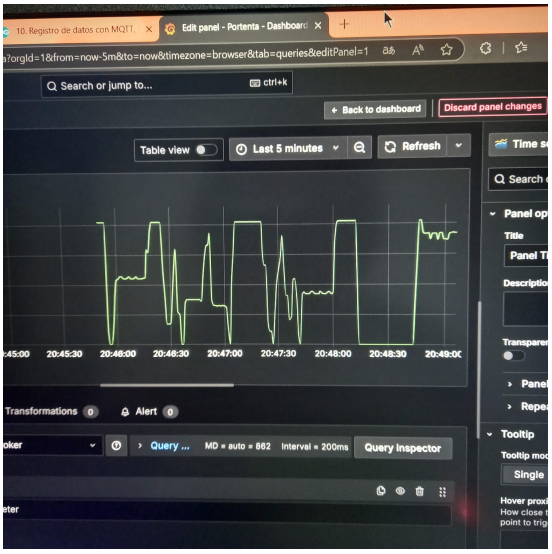
*3.3.2. **Creacion del dashboard***

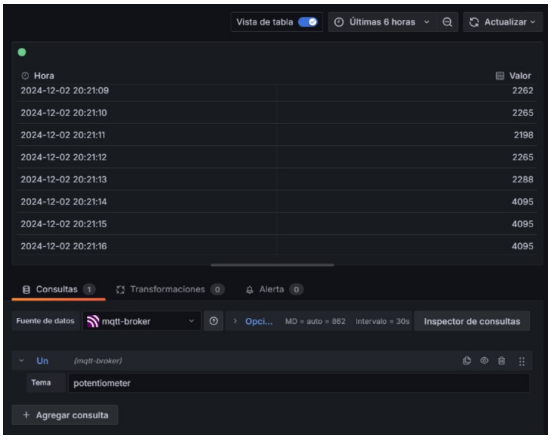
Finalmente, creamos el Dashboard en Grafana. Al abrirlo, conectamos el Data Source previamente configurado, seleccionando el nombre que asignamos en el paso anterior.

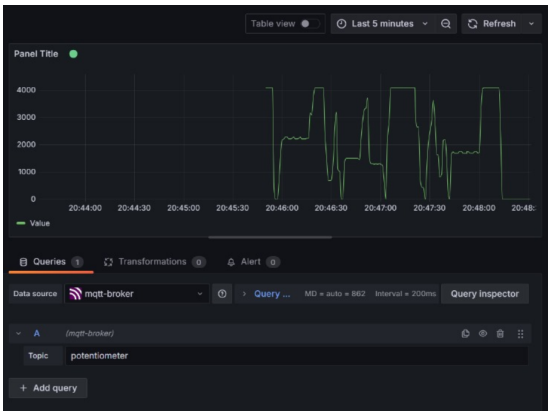


Además, vinculamos el topic al cual nos suscribimos, de modo que los datos enviados desde el ESP32 sean correctamente mostrados en el Dashboard. Para optimizar la visualización, ajustamos algunos parámetros, desactivando la opción de autorefresh y estableciendo el menor intervalo posible entre actualizaciones de datos. Esto asegura una visualización más fluida y precisa de la información en tiempo real. Con estos ajustes, el Dashboard estará listo para mostrar los datos de manera efectiva.









4. Análisis y Resultados

El proceso de desarrollo para establecer la conexión entre el ESP32 y el Portenta X8 utilizando el protocolo MQTT a través de Docker fue en general satisfactorio, aunque se presentaron ciertos desafíos técnicos que fueron superados a lo largo del proceso. A pesar de estos obstáculos, se logró establecer una comunicación exitosa entre ambos dispositivos IoT, permitiendo la transmisión de datos en tiempo real, lo cual era el objetivo principal.
Uno de los principales problemas enfrentados fue la gestión de las conexiones y la configuración del entorno de red, especialmente debido a la naturaleza dinámica de las IPs en el Portenta X8. Cada vez que el dispositivo se reiniciaba o se conectaba a una nueva red, era necesario actualizar las configuraciones, lo que generó algunos inconvenientes en los primeros intentos. Sin embargo, una vez configurados correctamente los entornos de Docker y MQTT, la comunicación entre el ESP32 y el Portenta X8 se estabilizó y funcionó correctamente.
El uso de Mosquitto en Docker fue crucial para facilitar la transmisión de mensajes entre dispositivos a través de MQTT, y la configuración del puerto 1883 para la escucha y la autorización anónima permitió que el ESP32 pudiera publicar datos sin necesidad de autenticación. Este paso fue clave, ya que permitió una integración rápida y sencilla con el Portenta X8.
Al integrar los datos en Grafana, se completó la parte final del proceso, logrando visualizar en tiempo real los datos enviados por el ESP32. La creación del Dashboard fue exitosa, permitiendo monitorear de manera efectiva los valores recibidos. Se realizaron ajustes en la visualización, como la desactivación del auto-refresh y la reducción del intervalo de actualización de datos, lo que optimizó el rendimiento y mejoró la experiencia de usuario.

5. Conclusiones

El uso de diversos dispositivos en un entorno IoT, como el ESP32 y el Portenta X8, demuestra una clara ventaja al permitir la recolección, transmisión y visualización de datos de manera eficiente y flexible. La capacidad de estos dispositivos para trabajar en conjunto habilita soluciones más completas, donde ESP32 puede ser responsable de capturar datos de sensores, mientras que el Portenta X8 actúa como el servidor para gestionar la transmisión y procesamiento de la información, y Grafana ofrece una potente plataforma de visualización en tiempo real. Esta integración de diferentes dispositivos no solo optimiza el flujo de datos, sino que también facilita la escalabilidad del sistema al poder incorporar más sensores y dispositivos en el futuro.
No obstante, un problema notable con el Portenta X8 es la necesidad de reconfigurar gran parte de la infraestructura cada vez que el dispositivo se apaga o reinicia. Esto se debe a que el dispositivo no maneja direcciones IP estáticas, lo que obliga a actualizar las configuraciones de red y contenedores tras cada reinicio. Este desafío puede generar inconvenientes y aumentar el tiempo de configuración, lo que puede afectar la eficiencia del sistema si no se gestionan adecuadamente los parámetros dinámicos. A pesar de esto, la solución puede encontrarse en la automatización de las configuraciones, aunque sigue siendo un aspecto a mejorar para optimizar el uso del Portenta X8.
En cuanto al protocolo MQTT, su implementación demostró ser fundamental para establecer una comunicación robusta, eficiente y de bajo consumo entre dispositivos. La capacidad de transmitir mensajes de manera asíncrona, con una estructura de publicación/suscripción, permite una gestión eficaz de los datos a medida que estos se generan y se distribuyen entre múltiples dispositivos sin necesidad de establecer conexiones directas. Esto mejora la escalabilidad y la flexibilidad del sistema, asegurando que los datos sean transmitidos y recibidos de manera segura, incluso con dispositivos de bajo consumo como el ESP32.
Finalmente, la potencia de los dispositivos utilizados en este proyecto, especialmente el Portenta X8 y el ESP32, ha demostrado ser crucial para lograr una solución IoT eficiente. Ambos dispositivos, con su capacidad de procesamiento y conectividad, brindan una plataforma sólida para el desarrollo de aplicaciones IoT, permitiendo ejecutar contenedores, gestionar redes, y realizar análisis de datos en tiempo real sin comprometer el rendimiento. Esta combinación de hardware potente y tecnología avanzada abre las puertas a un sinfín de aplicaciones en el campo de la automatización, mo-

nitoreo y control en entornos industriales, agrícolas y urbanos.

Referencias

[1] Portenta x8 User Manual,https://docs.arduino.cc/tutorials/portenta-x8/user-manual/#introduction.

[2] Docker,https://www.docker.com/.

[3] Grafana,https://grafana.com/.

[4] ChatGpt,https://chatgpt.com/.

[5] Paessler the monitoring experts, Que es MQTT,https://www.paessler.com/es/it-explained/mqtt#:~:text=MQTT%20son%20las%20siglas%20de,cuanto%20al%20ancho%20de%20banda..