Integración de tópicos MQTT y Kafka con HiveMQ

Jhordy Bayas[1,2 L00365906], Joshua Reyes[1,3 L00372677], and Roger Espinoza[1,4 L00366058]

[1] Universidad de las Fuerzas Armadas E.S.P.E, Sangolqui, Ecuador
[2] jhordyb3@gmail.com
[3] jsreyes@espe.edu.ec
[4] rjespinoza@espe.edu.ec

Abstract. In this document, we will present the connection between a nodemcu v1.0 with the public broker hivemq using MQTT, where our nodemcu will publish in a topic and with a web service we will subscribe to a topic to visualize what was sent by the nodemcu, an ultrasonic sensor and a led will be used to perform the tests..

Keywords: MQTT · Topic · publish · subscribe

1 Introducción

Conforme a la evolución de la tecnología basada en IoT, donde se puede apreciar que existen multitud de protocolos de comunicaciones con recursos muy limitados, es por ello que se plantea el uso del protocolo MQTT, el cual es un protocolo simple que permite una gran versatilidad, y poco consumo de recursos, ademas de soportar multiples conexiones.

2 Objetivos

2.1 Objetivo General

Desarrollar la integración de topicos MQTT y Kafka con HiveMQ

2.2 Objetivos Específicos

- Interpretar independientemente los beneficios, y utilidades que pueden ofrecer HiveMQ, y Apache Kafka.
- Verificar la implementaciones de trafico, mediante el usos de un sensor que proporciones lo datos necesarios, para verificar su comportamiento.
- Determinar los componentes necesarios para el correcto funcionamiento de Apache Kafka y HiveMQ.
- Establecer la diferencia que se obtiene al utilizar el protocolo de comunicación MQTT, referente a otros protocolos existentes.

3 Estado del Arte

Al revisar los distintos usos de MQTT en los proyectos investigados, se puede encontrar temas acordes para tener una guía del uso de este protocolo, con la utilización de HiveMQ, por el cual se llegó al artículo realizado por Sumit Pal, Sourav Ghosh, Sarasij Bhattacharya el cual menciona que la tecnología se ha desarrollado mucho para simplificar nuestras vidas. Con la evolución de Internet y la tecnología móvil junto con el desarrollo paralelo de una variedad de sistemas integrados, el enfoque hacia un mundo inteligente ha cobrado impulso y nos ha dado un nuevo concepto, el Internet de las cosas. Como resultado, gran parte de la tecnología de hoy en día está automatizada y los desarrolladores están desarrollando sistemas que recopilan datos de varios sistemas de sensores que pueden enviarse a cualquier parte del mundo a través de Internet y pueden usarse para una gran variedad de propósitos, incluido el control de dispositivos. A su vez mencionan que en su articulos han intentado estudiar uno de estos protocolos de Internet que hace posible tal comunicación, el protocolo MQTT. Usando un sistema de monitoreo ambiental, determinaremos la viabilidad de dicho protocolo para la transmisión de datos de sensores y luego usaremos los mismos datos para controlar dispositivos electrónicos. También comparamos el protocolo MQTT con el protocolo HTTP tradicional e intentamos descubrir cuál es el mejor, por ello se verificaron, como tratar de cual es el protocolo mas eficiente, ya sea de manera convencional, u ocupando el protocolo MQTT.[1]
Entonces teniendo en cuenta el uso de este protocolo en el monitoreo ambiental se puede apreciar, que debe tener otras opciones variadas como es en el articulo de Khin Me Me Thein, el cual nos habla sobre las referencia del uso de Apche Kafka donde menciona que es un mensaje de publicación-suscripción implementado como un registro de confirmación distribuido, adecuado tanto para fuera de línea como para Consumo de mensajes online. Es un sistema de mensajería desarrollado inicialmente en LinkedIn para recopilar y entregar volúmenes de eventos y datos de registro con baja latencia. La publicación de mensajes es un mecanismo para conectar varias aplicaciones con la ayuda de mensajes que se enrutan entre ellos, por ejemplo, por un intermediario de mensajes como Kafka. Actúa como una especie de registro de escritura que registra los mensajes en un almacén persistente y permite a los suscriptores leer y aplicar estos cambios a sus propias tiendas. En un marco de tiempo apropiado del sistema. Los suscriptores

comunes incluyen servicios en vivo que hacen agregación de mensajes u otro procesamiento flujos, así como Hadoop y tuberías de almacenamiento de datos que cargan prácticamente todos los feeds para procesamiento orientado a lotes.[2]

4 Marco Teórico

4.1 MQTT

Es una norma ISO (ISO / IEC PRF 20922) protocolo de mensajería basado en publicación-suscripción . Funciona sobre el protocolo TCP / IP . Está diseñado para conexiones con ubicaciones remotas donde se requiere una "huella de código pequeño" o el ancho de banda de la red es limitado. El patrón de mensajería de publicación-suscripción requiere un intermediario de mensajes . En 2013, IBM presentó MQTT v3.1 al cuerpo de la especificación de OASIS con un estatuto que aseguraba que solo se podían aceptar cambios menores en la especificación. MQTT-SN es una variación del protocolo principal dirigido a dispositivos integrados en redes que no son TCP / IP, como Zigbee .Históricamente, el "MQ" en "MQTT" provino de la línea de productos de cola de mensajes de IBM MQ (entonces 'MQSeries') .Sin embargo, no se requiere que la cola en sí sea compatible como una característica estándar en todas las situaciones. El middleware alternativo orientado a mensajes incluye el Advanced Message Queuing Protocol (AMQP) , Streaming Text Oriented Messaging Protocol (STOMP) , el IETF Constrained Application Protocol , XMPP ,DDS ,OPC UA y el Protocolo de Mensajería de Aplicación Web (WAMP) .

4.2 Apache Kafka

Es un proyecto de intermediación de mensajes de código abierto desarrollado por la Apache Software Foundation escrito en Java y Scala. El proyecto tiene como objetivo proporcionar una plataforma unificada, de alto rendimiento y de baja latencia para la manipulación en tiempo real de fuentes de datos. Puede verse como una cola de mensajes, bajo el patrón publicación-suscripción, masivamente escalable concebida como un registro de transacciones distribuidas,lo que la vuelve atractiva para las infraestructuras de aplicaciones empresariales.El diseño tiene gran influencia de los registros de transacción.

4.3 Rendimiento de Kafka

Debido a su capacidad de escalar masivamente y a su uso en estructuras a nivel de aplicaciones empresariales, el seguimiento del rendimiento de Kafka se ha convertido en un tema cada vez más importante. Actualmente existen varias plataformas de código abierto, como Burrow de Linkedin, y plataformas de pago como Datadog,1 que permiten hacer el seguimiento del desempeño de Kafka.

4.4 NODEMCU

Es una plataforma IoT de código abierto.Incluye el firmware que se ejecuta en el SoC Wi-Fi ESP8266 de Espressif Systems y el hardware que se basa en el módulo ESP-12. El término "NodeMCU" se refiere al firmware en lugar de a los kits de desarrollo. El firmware utiliza el lenguaje Lua. Se basa en el proyecto eLua y se basa en el SDK no operativo de Espressif para el ESP8266. Utiliza muchos proyectos de código abierto, como lua-cjson, y spiffs.

5 Verificacion funcionamiento

En base a lo que se puede utilizar, las distintos codigos se obtiene primeramente, debemos crear metodos publish y suscribe para cada uno de los nodemcu, por ello tambien se necesita establecer la coneción con el tópico que existe en Hive MQ, además para obtener información del dashboar se debe tener en cuenta el topico que enmarca el ESP, para que con ello podramos recibir lo datos colocados en la consola y actualizar de manera real los valores del sensor colocado

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>       // including the library of DHT11 temperature and humidity sensor




// Update these with values suitable for your network.
const int trigPin = 2;  //D4
const int echoPin = 0;  //D3
long duration;
int distance;
const char* ssid = "Jhordy";
const char* password = "19971997";
const char* mqtt_server = "broker.hivemq.com"; /// MQTT Broker
int mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT);
pinMode(LED_BUILTIN, OUTPUT); // Initialize the BUILTIN_LED pin as an output
 Serial.begin(115200);
```

```
 // Start up the library




 setup_wifi();
 client.setServer(mqtt_server, mqtt_port);
 client.setCallback(callback);

 Serial.println("Connected ");
 Serial.print("MQTT Server ");
 Serial.print(mqtt_server);
 Serial.print(":");
 Serial.println(String(mqtt_port));
 Serial.print("ESP8266 IP ");
 Serial.println(WiFi.localIP());
 Serial.println("Modbus RTU Master Online");



}

void setup_wifi() {

 delay(1000);
 // We start by connecting to a WiFi network
 Serial.println();
 Serial.print("Conectandose a ");
 Serial.println(ssid);

 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
 }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Message arrived [");
```

```
 Serial.print(topic);
 Serial.print("] ");
 for (int i = 0; i<length; i++) {
 Serial.print((char)payload[i]);
 }
 Serial.println();
 // Switch on the LED if an 1 was received as first character
if ((char)payload[0] == '1') {
digitalWrite(LED_BUILTIN, LOW); // Turn the LED on
//(Note that LOW is the voltage level
// but actually the LED is on; this is because
// it is acive low on the ESP-01)
} else {
digitalWrite(LED_BUILTIN, HIGH); // Turn the LED off
//by making the voltage HIGH
}

}
void reconnect() {
 // Loop until we're reconnected
 while (!client.connected()) {
 Serial.print("Attempting MQTT connection...");
 // Attempt to connect
 if (client.connect("Esp")) {

 Serial.println("connected");
 client.publish("jordy","Connected!");
 Serial.print("Subscribido!");
 client.subscribe("tec");
 } else {
 Serial.print("failed, rc=");
 Serial.print(client.state());
 Serial.println(" try again in 5 seconds");
 // Wait 5 seconds before retrying
 delay(5000);
 }
 }
}
void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);

// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);

 //char temperaturenow [15];
 //dtostrf(distance,2, 1, temperaturenow); //// convert float to char
 char cadena[16];
sprintf(cadena, "%d", distance);
 client.publish("jordy", cadena); /// send char

 if (!client.connected()) {
 reconnect();
 Serial.println("Desconectado");

 }
 client.loop();

 delay(10000);
}
\end{verbataim}
Se recibira en la base de datos lo sieguiente://
\begin{figure}[h]
    \centering
    \includegraphics[width=6cm]{imagenes/sql.jpeg}
    \caption{Vista de base de datos .}
    \label{fig:2}
\end{figure}
También, tendremos que declarar los métodos ocupados en Java:
\begin{figure}[h]
    \centering
    \includegraphics[width=6cm]{imagenes/50.jpg}
    \caption{Código en Eclipse}
    \label{fig:2}
\end{figure}
```

6 Conclusiones

– El ESP8266 es muy versátil para el desarrollo de proyectos ya que es fácil de trabajar y acoplar sus módulos con lo cual nos da una amplia capacidad

para realizar sistemas, aplicaciones en basadas en servicios IOT ademas de la integración de protocolo MQTT.
- Mediante Apache Kafka realiza la misma acción con los datos, así los puede persistir de forma duradera y facilita su lectura de forma ordenada y determinista con el extra de que al proporcionar distribución obtiene mecanismos de escalabilidad y de recuperación frente a fallos.
- Se puede aseverar que HiveMQ posee la capacidad de poder disgregar la comunicación de los clientes en diferentes Tópicos de publicación, pudiendo tener diferentes redes de sensores con diferentes propósitos sobre un mismo canal.
- Las distintas conexiones con el nodeMCU se puede concluir que la ventaja de este módulo es poder incluir este tipo de conexiones en servicios de internet, y también encontrar dichas librerías para una óptima utilización de la programación rea

7 Recomendaciones

- Tener conocimientos básicos de sobre MQTT las bases para ser ocupado mediante el ESP8266.
- Comprender las ventajas que posee Hive MQ respecto a otros programas como servidor
- Se recomienda verificar detalladamente las distintas opciones que posee apache Kafka además de determinar sus funciones básicas.

References

[1] Sumit. Pal, Sourav. Ghosh, and Sarasij. Bhattacharya. "Study and Implementation of Environment Monitoring System Based on MQTT". In: *Environmental and Earth Sciences Research Journal* 4.1 (2017), pp. 23–28. DOI: 10.18280/eesrj.040105.

[2] Hein Ph. "Apache Kafka: Next Generation Distributed Messaging System KHIN ME ME THEIN". In: 03.47 (2014), pp. 9478–9483. URL: www.twitter.com/.