Integración de tópicos MQTT y Kafka con HiveMQ

Tecnología de Software para Electrónica

Trabajo de Investigación

Autor:
Jordy Bayas Escobar (alumno)
Joshua Reyes Jurado (alumno)
Roger Espinoza Naranjo (alumno)
Tutor/es:
Darwin Alulema Flores (tutor1)

10 de Julio de 2019

Integración de tópicos MQTT y Kafka con HiveMQ

Subtítulo del proyecto

Autor/es
Jordy Bayas Escobar (alumno)
Joshua Reyes Jurado (alumno)
Roger Espinoza Naranjo (alumno)

Tutor/es
Darwin Alulema Flores (tutor1)
Departamento de Eléctrica y Electrónica

Tecnología de Software para Electrónica

Quito, 10 de Julio de 2019

Índice general

Índice de figuras

1 Planteamiento del Problema

Integración de tópicos MQTT y Kafka con HiveMQ

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 recurrente, el cual beneficiara en el trafico de los datos almacenados, entre ellos se encuentra HiveMQ y Apache Kafka los cuales son corredores de MQTT que funcionan basicamente como un servidor para los mismos.

2 Objetivos

2.1 Objetivo General

Desarrollar la integración de topicos MQTT y Kafka con HiveMQ

2.1.1 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 distitntos 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]

Al verificar los mecionado cuales son las distintas funciones que obtenemos ocupando las pescies de registros o servidores, tambien podemos interpretar el trabajo realizado por Monika Kashyap, Vidushi Sharma, Neeti Gupta donde nos menciona reiteradamente sobbre el Internet of Things (IoT) el cual nos permite la conexión entre dispositivos que utilizan internet con la capacidad de recopilar e intercambiar datos. Estos dispositivos suelen estar conectados con microcontroladores como Arduino, sensores, actuadores y conectividad a Internet. En este contexto, El protocolo de transporte de telemetría de Message Queue Server (MQTT) juega un papel importante para intercambiar los datos o la información entre Los dispositivos en IoT sin conocer las identidades entre sí. Este artículo presenta diferentes modelos de servicio para la comunicación. en Internet de las cosas (IoT). El Modelo A presenta el uso de USB en serie como medio de transmisión, mientras que el Modelo B usa el Mensaje Protocolo de transporte de telemetría de colas (MQTT) que implementa un módulo Wi-Fi (ESP8266-12) para conectar el sistema a Internet. Para la comunicación, se utiliza el concepto de editor y suscriptor. Los mensajes se publican o se suscriben con la ayuda de un intermediario. o servidor. Este agente se encarga de dispersar los mensajes a los clientes intencionados según la elección del tema de un mensaje. Broker en MQTT también se llama servidor. Algunos agentes utilizados en MQTT son: -Mosquitto, Adafruit, hiveMQ. [3]
Entonces teniendo en cuenta distintos articulos para la recoleccion de datos, tambien debemos tener en cuenta el articulo de Ghyzlane Cherradi,Adil El Boulziri y Azedine Boulmakoul, el cual nos menciona en palabras parciales sobre la recogida y organización de los datos los cuales son una parte integral y crítica del proceso de evaluación de riesgos. Por ello se apunta hacia monitorear el riesgo, que puede ayudar a los gerentes a producir acciones, con el fin de reducir el riesgo. Actualmente podemos compartir datos desde dispositivos conectados enn internet gracias a las tecnologías de IoT. En su trabajo podemos apreciar una propoción referente a una arquitectura de recopilación de datos inteligente basada en MQTT (Mensaje Protocolo de transporte de cola de telemetría), con el fin de construir una rápida y la plataforma escalable de IoT, que puede recopilar y transportar datos. Por ultimo se puede tener en cuenta el Articulo del Dr Sames Joloudi donde propone la utilización del protocolo de comunicaciones MQTT en las aplicaciones City Smart en el internet de las cosas, con una aplicación práctica típica y propponiento un modelo de red., basado protocolo MQTT en un escenario desarrollado.Donde se dependera que las aplicaciones sean medianas a pequeñas, con base en la nube para obtener buenos resultados, también de una Comunicación orientada a mensajes basados en eventos.
[4] Como se puede observar el tema de la investigación tiene diversas ámbito de aplicacion, artículos que poseen cierta relación con lo que se está investigando, pero este trabajo se diferencia de los demás en que se quiere integrar el módulo ESP8266 de manera que interactue con el sensor colocado, para que con ello mediante la conexión WI-FI, se puede comunicar con el Borker MQTT con Hive MQ como el servidor de la comunicación, y a suvez ocupar Apache Kafka como un plataforma para la manipulación en tiempo real

4 Marco Teórico

4.1 HiveMQ

HiveMQ es un corredor MQTT. Un intermediario es básicamente la parte del servidor en la comunicación MQTT. Además de la funcionalidad de Common Broker, HiveMQ proporciona una funcionalidad ampliada como la creación de clústeres (alta disponibilidad), una integración profunda a su infraestructura de TI e aplicación y una seguridad mejorada.

Figura 4.1: HiveMQ

4.2 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) .

Figura 4.2: MQTT

4.3 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.

4.3.1 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.

Figura 4.3: Apache 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.

Figura 4.4: NODEMCU

5 Diagramas

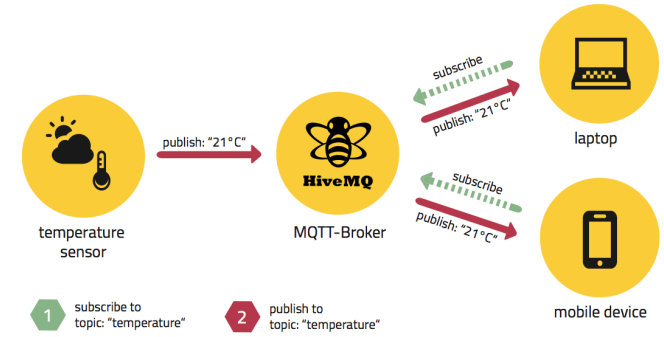


Figura 5.1: Conexión HiveMQ

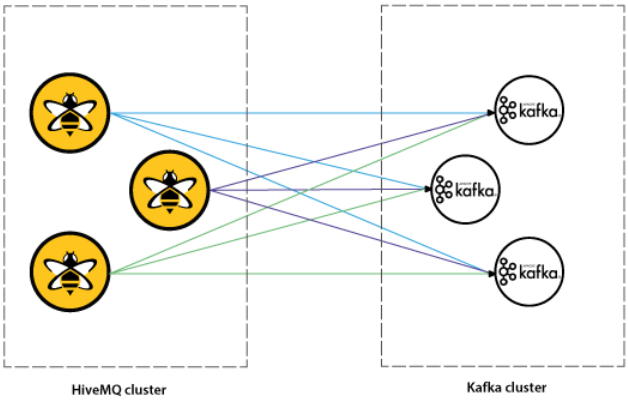


Figura 5.2: Comunicaciones entre HiveMQ y Apache Kafka

6 Lista de Componentes

6.1 HiveMQ

HiveMQ facilita la transferencia de datos hacia y desde los dispositivos conectados de manera eficiente, rápida y confiable. Hacemos posible la construcción de productos conectados que permitan nuevos negocios digitales.

Figura 6.1: HiveMQ

6.2 HiveMQ+Apche Kafka

HiveMQ es una plataforma empresarial MQTT que hace posible mover rápidamente los datos de los dispositivos IoT conectados. La nueva extensión empresarial de HiveMQ para Kafka ofrece una implementación nativa del protocolo Kafka integrado en el agente HiveMQ MQTT. Esto permite una integración transparente y escalable de flujos de datos MQTT entre millones de dispositivos IoT y varios clústeres Kafka.

Figura 6.2: HiveMQ+Apache Kafka

6.3 ESP8266

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.

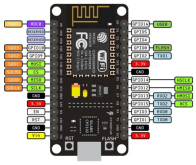


Figura 6.3: NODEMCU ESP8266

6.4 Arduino IDE

Es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirla en una salida: activar un motor, encender un LED y publicar algo en línea.

6.5 Sensor Ultrasónico HC-SR04

El HC-SR04 es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm.

Figura 6.4: HC-sr04

7 Mapa de Variables

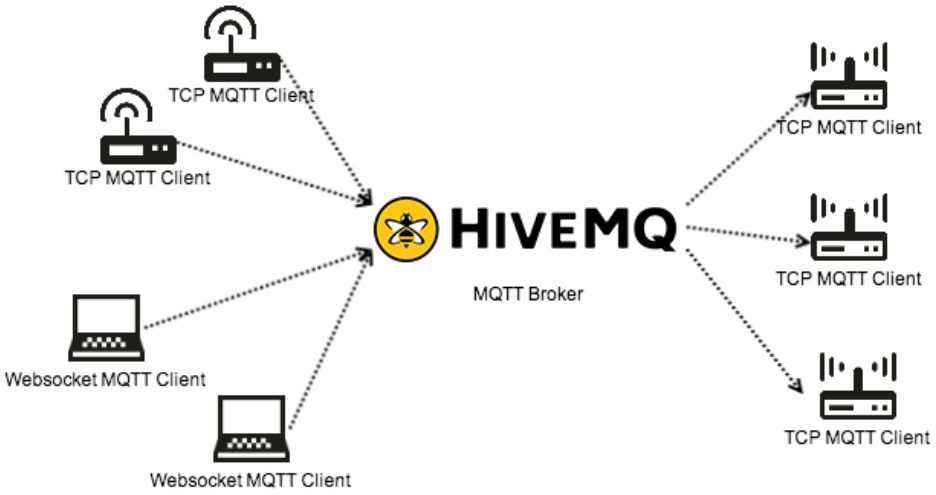


Figura 7.1: Conexión MQTT-HiveMQ

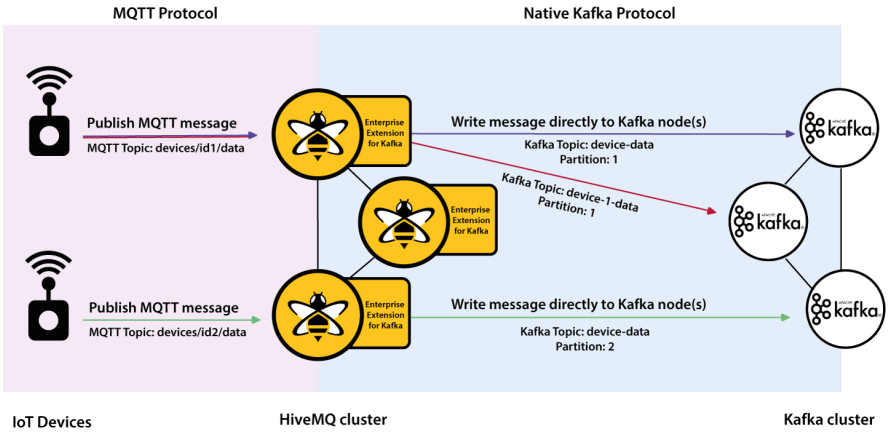


Figura 7.2: Conexión HiveMQ+Apache Kafka

8 Explicación

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 s
```

```
// 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);
}
```

9 Descripción de Prerrequisitos y Configuración

Para la realizacion del sistema se necesita una lista de programas y ya previamente instalados como:

- HiveMq o Hive-Dashboard
- Extensions Apache Kafka
- Librería Node MCU

9.0.1 HiveMQ+extensions Apache Kafka

Se debe contar HiveMQ Enterprise Extension para Kafka hace posible integrar sin problemas los mensajes MQTT con los clusters de Kafka. La extensión implementa el protocolo Kafka nativo, por lo que HiveMQ actúa como un cliente Kafka de primera clase. Esto permite que los temas de MQTT se asignen directamente a los temas de Kafka y reenvíen los mensajes directamente a varios clústeres de Kafka de una manera confiable, escalable y de alto rendimiento

9.0.2 Librería NodeMCu

Se debe tener en cuenta la librería WiFi para ESP8266 ha sido desarrollada basándose en el SDK de ESP8266, usando nombres convencionales y la filosofía de funcionalidades generales de la librería WiFi de Arduino. Con el tiempo, la riqueza de las funciones WiFi del SDK de ESP8266 pasadas a ESP8266/Arduino superan a la librería WiFi de Arduino y se hizo evidente que tenemos que proporcionar documentación por separado sobre lo que es nuevo y extra.

Esta documentación lo guiará a través de varias clases, métodos y propiedades de la librería ESP8266WiFi. Si eres nuevo en C++ y Arduino, no te preocupes. Comenzaremos por conceptos generales y luego pasaremos a la descripción detallada de los miembros de cada clase en particular, incluidos los ejemplos de uso.

El alcance de la funcionalidad que ofrece la biblioteca ESP8266WiFi es bastante extensa y por lo tanto, esta descripción se ha dividido en documentos separados marcados con :arrowright:.

10 Aportaciones

Utilizacion de HiveMQ websocket-client para comprobaciones de nuestra aplicacion, se lo puede realizar desde la pagina web:
http://www.hivemq.com/demos/websocket-client/. Además como otra aportacipón se agrego tambien el envio de datos de un sensor ultrasónico en tiempo real, esto permite de una manera mas extacto aplicaciones como obtener de manera confiable el nivel de agua que existe dentro de un tanque.

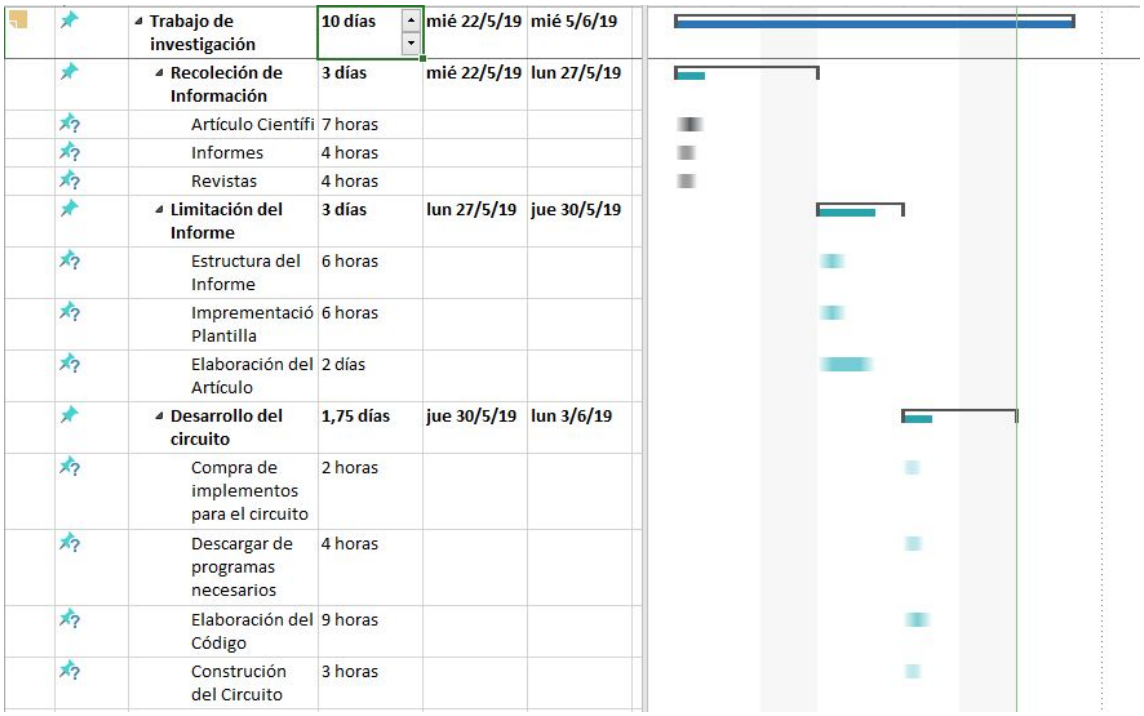
11 Conclusiones

- 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 realizada.

12 Recomendaciones

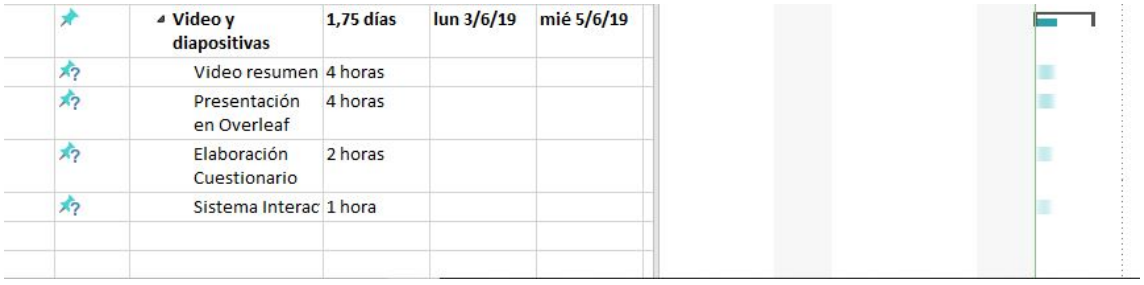
- Tener conocimientos básicos de sobre MQTT las bases para ser ocupado mediante el ESP8266.
- Comprender las ventajas que posee Hive MQ respecto a otros programas como servidor
- Se recomienda verificar detalladamente las distintas opciones que posee apache Kafka además de determinar sus funciones básicas.

13 Cronograma



	Trabajo de investigación	10 días	mié 22/5/19	mié 5/6/19
	Recolección de Información	3 días	mié 22/5/19	lun 27/5/19
	Artículo Científi	7 horas		
	Informes	4 horas		
	Revistas	4 horas		
	Limitación del Informe	3 días	lun 27/5/19	jue 30/5/19
	Estructura del Informe	6 horas		
	Impremantació Plantilla	6 horas		
	Elaboración del Artículo	2 días		
	Desarrollo del circuito	1,75 días	jue 30/5/19	lun 3/6/19
	Compra de implementos para el circuito	2 horas		
	Descargar de programas necesarios	4 horas		
	Elaboración del Código	9 horas		
	Construción del Circuito	3 horas		

Figura 13.1: Cronograma de actividades parte 1



	Video y diapositivas	1,75 días	lun 3/6/19	mié 5/6/19
	Video resumen	4 horas		
	Presentación en Overleaf	4 horas		
	Elaboración Cuestionario	2 horas		
	Sistema Interac	1 hora		

Figura 13.2: Cronograma de actividades parte 2

14 Repositorio

14.1 Git Hub

https://github.com/Jhordyb3/Producto_unidad_2/blob/master/Trabajo_Investigacion_3.zip

14.2 Links Overleaf

https://es.overleaf.com/9464852251hdbcgcnjzsts
https://es.overleaf.com/read/wmcypgtyftdc
https://es.overleaf.com/5485313766qfwbmyjgctwp

15 Anexos

15.1 Manual de usuario

1.Primeramente se debe tener instaladas todas las librerías necesarias tanto para lo sensores, como para el NODEMCU.

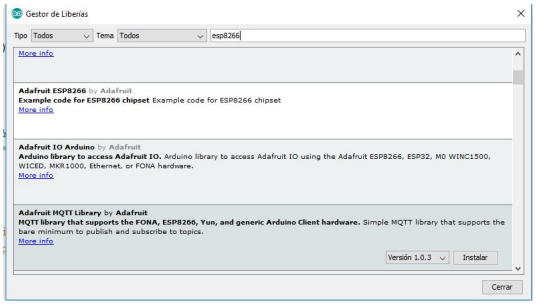


Figura 15.1: Librería ESP

Figura 15.2: Librería ESP

2. Instalar HiveMQ+ExtensionApche Kafka

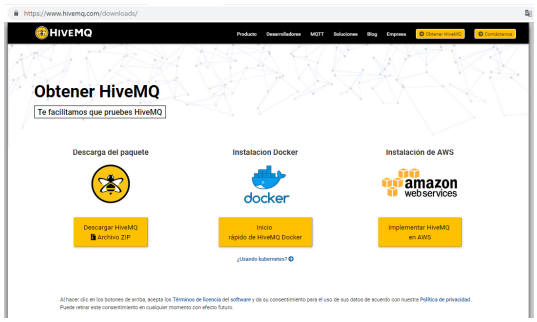


Figura 15.3: Instalación HiveMQ

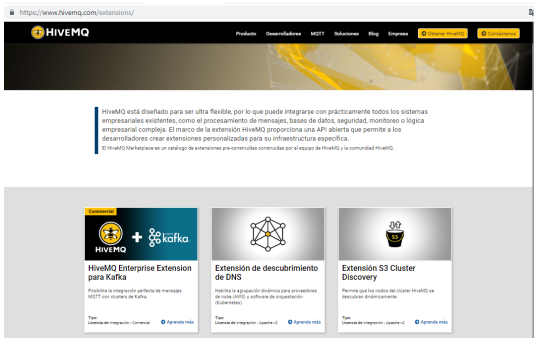


Figura 15.4: Extensión Apache Kafka

3.Ingresar en la carpeta de HiveMQ (Version que poseas), dentro de la misma carpeta se encontrara extensión entonces se extrae el .zip, descargado de la extensión apache kafka.

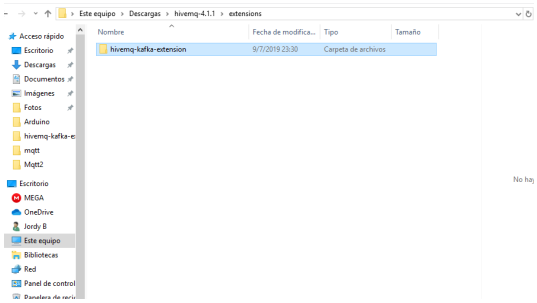


Figura 15.5: Instalación e la extensión

4. Configuración del archivo xml

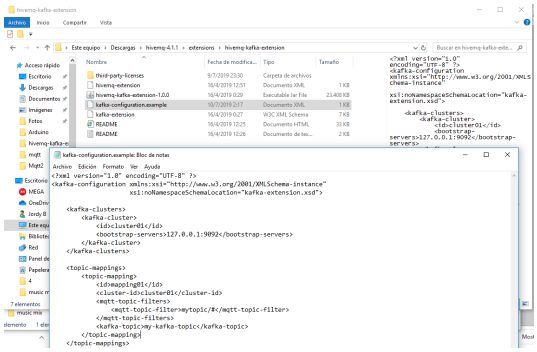


Figura 15.6: Configuración del archivo xml

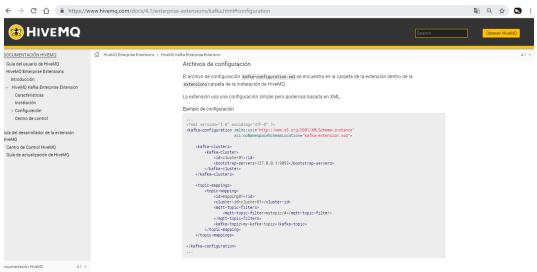


Figura 15.7: Configuración del archivo xml-2

5. Ejecución del Hive MQ
El archivo ejecutable se encuentra en la carpeta bin, y se ingresa a run.

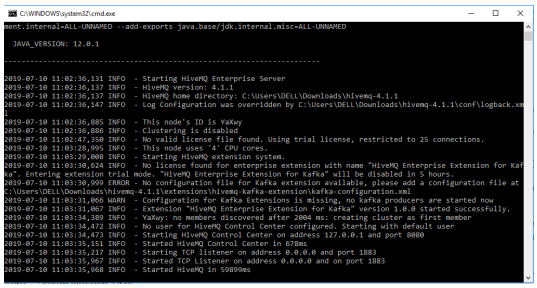


Figura 15.8: Inicialización de HiveMQ

6.Ingreso al dashboard de HiveMQ

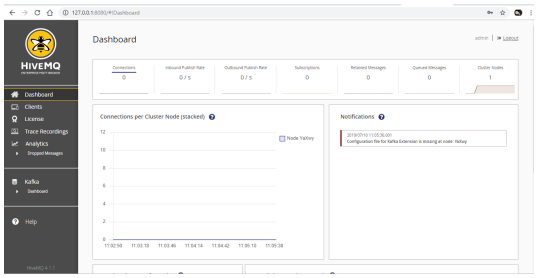
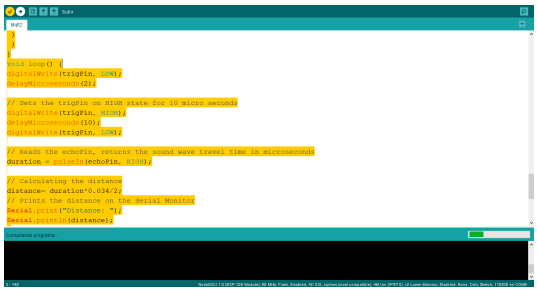


Figura 15.9: Dashboard HiveMQ

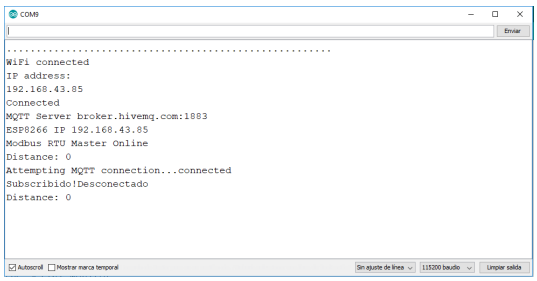
7. Ingreso del codigo al nodemcu



```
}
}
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);
```

Figura 15.10: Cargado del codigo

8. Conexión establecida con el servidor HiveMQ



```
COM9
Enviar
.....................................................
WiFi connected
IP address:
192.168.43.85
Connected
MQTT Server broker.hivemq.com:1883
ESP8266 IP 192.168.43.85
Modbus RTU Master Online
Distance: 0
Attempting MQTT connection...connected
Subscribido!Desconectado
Distance: 0
Autoscroll  Mostrar marca temporal
Sin ajuste de línea  115200 baudio  Limpiar salida
```

Figura 15.11: Conexión

9.Verificación del dato obtenido en HiveMQ-Dashboard Client

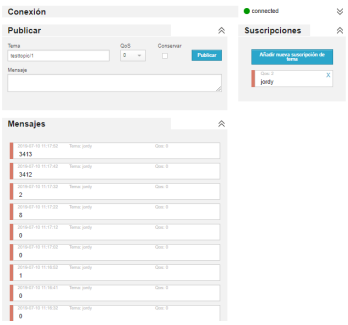


Figura 15.12: Datos en DashboardCLient

Bibliografía

[1] Sumit. Pal, Sourav. Ghosh, and Sarasij. Bhattacharya. Study and Implementation of Environment Monitoring System Based on MQTT. *Environmental and Earth Sciences Research Journal*, 4(1):23–28, 2017. doi: 10.18280/eesrj.040105.

[2] Hein Ph. Apache Kafka: Next Generation Distributed Messaging System KHIN ME ME THEIN. 03(47):9478–9483, 2014. URL `www.twitter.com/`.

[3] Monika Kashyap, Vidushi Sharma, and Neeti Gupta. Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. *Procedia Computer Science*, 132 (Iccids):1611–1618, 2018. ISSN 18770509. doi: 10.1016/j.procs.2018.05.126. URL `https://doi.org/10.1016/j.procs.2018.05.126`.

[4] S Jaloudi. MQTT for IoT-based Applications In Smart Cities. (April), 2018.

[5] Ghyzlane Cherradi, Adil El Bouziri, and Azedine Boulmakoul. Smart Data Collection Based on IoT Protocols. *Jdsi'16, Issn 2509-2103*, (November 2017), 2016. URL `https://www.researchgate.net/publication/320865914{_}Smart{_}Data{_}Collection{_}Based{_}on{_}IoT{_}Protocols`.