UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

ELECTRÓNICA Y AUTOMATIZACIÓN

RPC-SO-19-No.277-2018

OPCIÓN DE TITULACIÓN:

PROYECTOS DE DESARROLLO

TEMA:

DESARROLLO DE UNA PLATAFORMA BASADA EN INTERNET DE LAS COSAS PARA EL MONITOREO Y REGISTRO DE INFORMACIÓN DE DISPOSITIVOS INDUSTRIALES

AUTOR:

FERNANDO PATRICIO VÉLEZ IÑIGUEZ

DIRECTOR:

EDISON GEOVANNY NARANJO ANDRADE

CUENCA - ECUADOR

2022

COHORTE 2018 - 2020

Autor:

Fernando Patricio Vélez Iñiguez
Ingeniero en Sistemas.
Candidato a Magíster en Electrónica y Automatización, Mención en Informática Industrial por la Universidad Politécnica Salesiana - Sede Cuenca.
fvelezi@est.ups.edu.ec

Dirigido por:

Edison Geovanny Naranjo Andrade
Ingeniero en electrónica y control.
Master Universitario en Tecnología Educativa y Competencias Digitales.
edigeo12@hotmail.com

DESARROLLO DE UNA PLATAFORMA BASADA EN INTERNET DE LAS COSAS PARA EL MONITOREO Y REGISTRO DE INFORMACIÓN DE DISPOSITIVOS INDUSTRIALES

Índice general

Índice de Figuras

Índice de Tablas

Resumen

El presente trabajo de titulación consiste en construir una plataforma IOT desde su diseño, desarrollo e implementación, para ofrecer un producto de calidad a un precio más accesible a la realidad local. Existen muchas opciones en el mercado, siendo estos productos comerciales y de libre distribución, para implementar soluciones de la Industria 4.0, pero estos productos normalmente tienen el limitante de tener un costo muy elevado, y los productos freeware tienen restricciones tanto en la comunicación como el número de dispositivos soportados.

El prototipo que se propone mediante este trabajo, supera estos inconvenientes y representa una opción viable para empresas locales cuyo presupuesto es limitado.

Abstract

The present degree work consists of building an IOT platform from its design, development and implementation, to offer a quality product at a price that is more accessible to the local reality. There are many options on the market, these being commercial and free distribution products, to implement Industry 4.0 solutions, but these products usually have the limitation of having a very high cost, and freeware products have restrictions in both communication and use, limiting the number of supported devices.

The prototype that is proposed through this work, overcomes these drawbacks and represents a viable option for local companies whose budget is limited.

Capítulo 1

Introducción

1.1. Descripción general del problema

Hoy en día las necesidades por las que atraviesa la industria en general, obligan a cambiar la percepción de cómo manejar el proceso de la producción, con la finalidad de ser más eficientes, adoptar mecanismos que eleven la producción industrial especialmente la manufacturera, por lo que nos estamos dirigiendo hacia la transformación industrial para la explotación de nuevos potenciales con la ayuda de la tecnología. Con esto nos referimos al concepto de industria 4.0.

Este término se refiere a la cuarta revolución industrial, misma que se basa en el desarrollo de Tecnologías de Información y Comunicaciones, con lo cual se lleva a un nivel superior la automatización inteligente de sistemas ciber-físicos con control descentralizado y conectividad avanzada[Rojko, 2017].

La implementación de industria 4.0 implica necesariamente el desarrollo tecnológico e innovación, con lo cual se mejorará el sistema de manufactura, gestión y forma de realizar negocios, permitiendo ofrecer una propuesta de valor adicional a los clientes. Este desarrollo tecnológico va enfocado hacia la denominada "computación en la nube" o su término en inglés "Cloud Computing", que implementa infraestructura y servicios en internet.

La finalidad es proveer una herramienta que por medio de la utilización de "Computación en la Nube" facilite la exposición de datos esenciales de procesos industriales, de tal manera se logre reducir la brecha digital hacia la ya popular industria 4.0, que en nuestro país aún no ha sido adoptada a gran escala.

1.2. Alcance

Realizar el diseño, desarrollo, implementación, pruebas y comparativas de resultados de un servidor IoT, que permita supervisar variables de procesos de dispositivos medidores involucrados en el proceso de producción de empresas industriales a través de la WEB, mediante la captura de datos mediante el protocolo MQTT, HTTP; almacenamiento de valores y la posterior generación de reportes.

Las medidas comparativas para verificar resultados se las realizará frente a herramientas comerciales existentes en el mercado al momento de desarrollar este proyecto.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un software como servicio en la nube para el monitoreo de dispositivos IoT, almacenamiento de información y generación de reportes; que represente una solución de bajo costo frente a otras opciones comerciales en el mercado.

1.3.2. Objetivos específicos

- Analizar los protocolos más apropiados para el monitoreo de dispositivos IoT.
- Diseñar e implementar una interfaz de usuario para la configuración de dispositivos IoT.
- Generar reportes de lecturas obtenidas por dispositivos IoT.
- Realizar un análisis comparativo de costos de la solución propuesta frente a opciones comerciales.

1.4. Contribuciones

Este proyecto de titulación está orientado a proveer un servicio de publicación de variables del funcionamiento de procesos en una industria, contribuyendo así con un producto de bajo costo para estimular a la industria local a adecuar sus procesos hacia la mencionada Industria 4.0.

1.5. Organización del documento

El siguiente capítulo presenta el marco teórico donde se realiza una introducción general a los servicios que en la actualidad se ofrecen en computación en la nube, así como una descripción de los tipos de nube, y los principales protocolos utilizados en IoT organizado por capas basadas en el modelo OSI. Además, se presenta una descripción más profunda del protocolo MQTT, el cual es el protocolo utilizado en este proyecto de titulación por varias razones que motivaron el uso del mismo.

El tercer capítulo realiza un estudio de las plataformas comerciales más utilizadas en la actualidad para solucionar el manejo de lectura de datos de dispositivos IIoT.

En el cuarto capítulo se documenta la construcción del servidor IoT, que es el tema central de esta tesis.

El quinto capítulo realiza un análisis económico del producto creado frente a productos comerciales, así como métricas comparativas entre las mencionadas plataformas. Finalmente, se presentan conclusiones y recomendaciones que justifican si esta plataforma alcanzó los objetivos plateados.

Capítulo 2

Marco teórico

2.1. Modelos de servicio

La arquitectura de los ambientes de Computación en la Nube está dividida en cinco capas principales que son: Infraestructura física, Infraestructura virtual, Plataforma, Aplicación y Red. En base a estas capas se han definido modelos de servicios, como se muestra en la Figura 2.1.

Los servicios constan al lado izquierdo de la figura, las capas involucradas se muestran en la parte central, y a la derecha se muestra el agente involucrado en la provisión del servicio.

2.1.1. Software como un Servicio (SaaS)

En este modelo, un proveedor de software licencia una aplicación y la ofrece como un servicio bajo demanda. Dicho servicio corre en la nube y múltiples usuarios pueden acceder por medio de un navegador web o aplicaciones en teléfonos móviles. Este modelo elimina la necesidad de instalar y ejecutar las aplicaciones en forma local [Rani and Ranjan, 2014].

Ejemplos de esta plataforma son Google apps, Zoho que es un conjunto de aplicaciones que incluyen cliente de correo electrónico, así como software CRM (Customer Relationship Management).

2.1.2. Plataforma como un Servicio (PaaS)

Este modelo ofrece un ambiente de desarrollo como un servicio, donde las aplicaciones son construidas usando un conjunto de lenguajes de programación y otros aplicativos[Rani and Ranjan, 2014]. Los servicios ofrecidos pueden incluir herramientas para el desarrollo, integración y

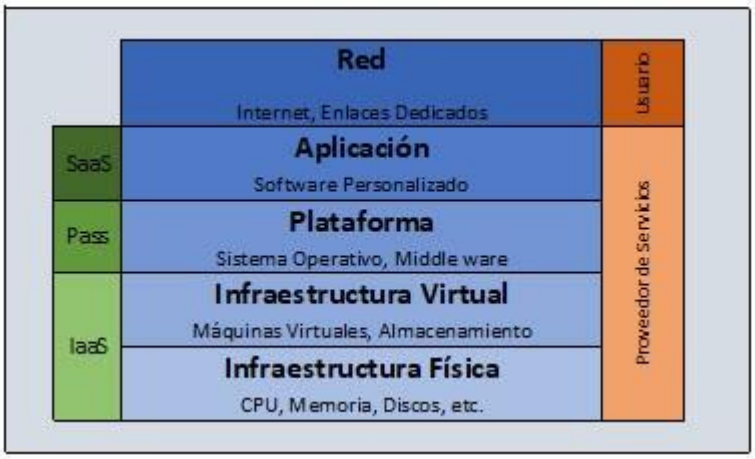


Figura 2.1: Modelos de computación en la nube y capas. Fuente: [Al-Masri et al., 2020] Elaboración propia

pruebas de ejecución. Como ejemplos de esta plataforma tenemos Microsoft Azure, Heroku, Aneka.

2.1.3. Infraestructura como un Servicio (IaaS)

La provisión de servicios IaaS, se refiere a la implementación y mantenimiento de infraestructura, que permite brindar hardware físico o virtual a clientes, por ejemplo, servicios de almacenamiento, máquinas virtuales, entre otros[Sala-Zárate and Colombo-Mendoza, 2012]. Con esta provisión de servicios se ofrece al cliente una eliminación conveniente de instalaciones en el sitio, brindando las mismas en forma remota. Los proveedores más conocidos de este tipo de servicio son Amazon Web Services (AWS), DigitalOcean, Vultr, Cisco Metapod, y otros.

2.2. Tipos de nube

En computación en la nube además de modelos de servicios existen modelos de despliegue, los que definen la forma de acceso a la nube, entre los cuales constan: Privada, Comunitaria, Pública e Híbrida. Estos tipos varían en función de quien es dueño y quien administra los servicios[Sala-Zárate and Colombo-Mendoza, 2012].

2.2.1. Privada

La Nube Privada, o nube interna, hace referencia cuando se ofrece los servicios de computación en la nube a negocios u organizaciones cuya preferencia es mantener los datos en un ambiente controlado y privado[Jin et al., 2010].

2.2.2. Comunitaria

Nube Comunitaria, es una nube compartida y administrada por un conjunto de compañías u organizaciones que tienen requerimientos de servicios en común, y se desenvuelven en un dominio que las relaciona.

2.2.3. Pública

La Nube Pública, o nube externa, ofrece los servicios de computación en la nube a un nivel más amplio, donde usuarios en general podrán acceder a los beneficios de estos servicios, sin restringir únicamente a segmentos privados como en el caso anterior[Jin et al., 2010][Moghaddam et al., 2015].

2.2.4. Híbrida

La Nube Híbrida está compuesta de muchas nubes públicas, privadas y/o comunitarias. Esta ofrece más flexibilidad que las anteriores descritas. En una nube híbrida, parte de los servicios podrían provenir de las nubes privadas, y otros de redes públicas y comunitarias[Moghaddam et al., 2015].

2.3. Industria 4.0 e Internet de las Cosas - IoT

El término IoT (Internet of Things) o su correspondiente en español (Internet de las cosas) ha sido introducido hace varios años para representar la comunicación entre los elementos físicos denominados “cosas” y un servidor centralizado que se encuentra disponible en la nube permitiendo un acceso generalizado. Las “cosas” son cualquier aparato conformado por electrónica embebida que es capaz de transferir datos por medio de una red sin la interacción manual de humanos [Rojko, 2017].

Debido a la introducción de la Industria 4.0 ha surgido la necesidad de implementar IoT para el control de procesos industriales, produciéndose de esta manera la adopción del concepto IIoT (Industrial Internet of Things), lo que posibilita el monitoreo de dispositivos desde un servidor

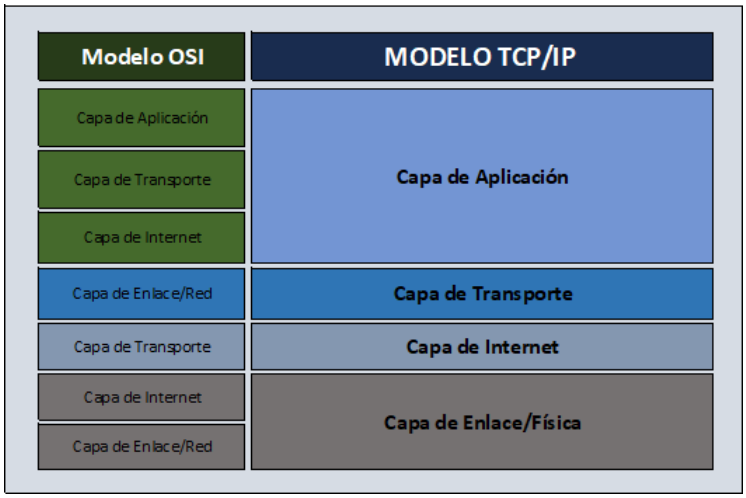


Figura 2.2: Correspondencia entre modelo OSI y modelo TCP/IP. Fuente: [Rani and Ranjan, 2014] Elaboración propia

en la nube, es decir, permite la comunicación entre elementos y agentes de control [Munirathinam, 2020].

2.4. Protocolos de comunicación utilizados en IoT

El modelo OSI divide el proceso de comunicación en siete capas, con lo que realiza la implementación de protocolos dedicados a cada capa, como lo muestra la figura 2.2. Este modelo ayuda al entendimiento de la comunicación IoT y los protocolos utilizados.

Una revisión del modelo OSI permite comprender los protocolos dedicados a IOT, ya que esta última tiene su propia pila de protocolos, la cual está organizada en varias capas que tiene su correspondencia con el modelo OSI, estas son la capa de Aplicación, Transporte, Internet y capa de Enlace/Física. Para un mejor entendimiento se ha organizado los protocolos IoT en función del modelo TCP/IP, el cual agrupa varias capas del modelo OSI, para enfocarse en cuatro principales, como lo muestra la figura 2.3.

Los usuarios interactuarán directamente con la capa de aplicación, de aquí su gran importancia ya que garantiza el intercambio de mensajes entre interfaces y herramientas por medio de la web. La capa de transporte se encarga de establecer la comunicación y transmisión entre nodos de una red.

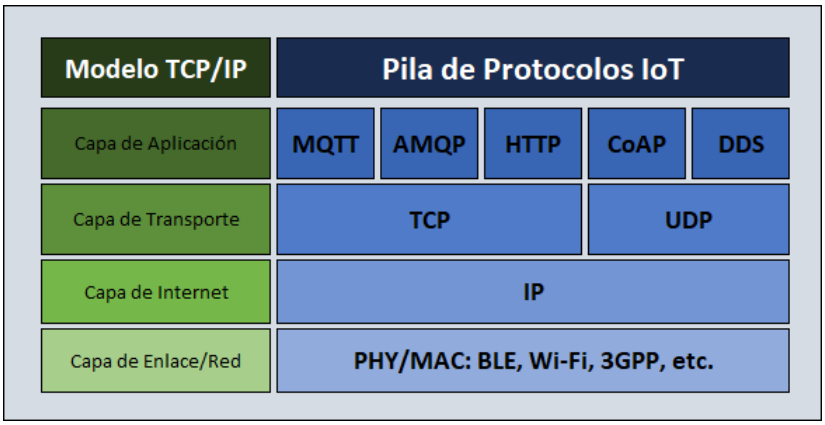


Figura 2.3: Opciones de la pila de protocolos IoT. Fuente: [Elgazzar, 2015] Elaboración propia

Esta garantiza que los paquetes lleguen en orden y sin error para lo cual utiliza técnicas como la confirmación de recepción de datos para garantizar la fiabilidad de las comunicaciones.

La capa de Internet representa la segunda capa en el modelo TCP/IP. La principal tarea que realiza es el envío de paquetes desde cualquier red, así como la recepción en el destino sin importar la ruta tomada.

La capa de enlace/red es la de más bajo nivel responsable de la transferencia de información a través del medio físico. Este define como los datos deben ser enviados por la red, y es responsable de la transmisión de datos entre dos dispositivos en la misma red.

El conjunto de protocolos que se describen abajo, son los más utilizados hoy en día.

2.4.1. Protocolo CoAP

Constrained Aplicación Protocol, es un protocolo basado en la arquitectura REST, el cual facilita la transferencia de información mediante el uso de la web, posibilitando el uso de dispositivos inteligentes en nodos y redes restringidas[Thota and Kim, 2016]. El protocolo está diseñado para comunicaciones (M2M – Machine to Machine), especialmente en aplicaciones de energía inteligente y automatización de edificios.

Este protocolo provee un modelo de interacción entre las aplicaciones finales de usuarios, provee servicios de exploración para descubrimiento

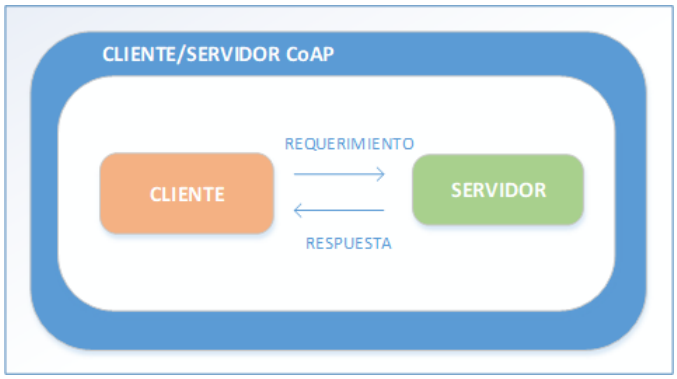


Figura 2.4: Comunicación Cliente-Servidor - CoAP Fuente: [Thota and Kim, 2016] Elaboración propia

integrado de servicios. Este interactúa fácilmente con HTTP para la integración con la web.

CoAp utiliza URI (Uniform Resource Identifier) para la identificación de dispositivos, provee métodos REST como GET, POST, PUT y DELETE. Por otra parte, es basado en el protocolo "liviano" UDP, a diferencia de MQTT que es basado en el protocolo TCP. Permite multicasting IP, que es utilizado para la comunicación de grupos de dispositivos.

Modelo de comunicación

El modelo de comunicación consiste en un intercambio de mensajes entre el cliente y el servidor, se lo realiza a partir de la solicitud inicial, utilizando métodos predefinidos con funciones específicas, como lo muestra la figura 2.4.

Los métodos soportados son los siguientes:

- GET – Para recibir información.
- POST – Para transmitir información.
- PUT – Para actualizar recursos en el servidor.
- DELETE – Para eliminar recursos del servidor.

Estructura del mensaje

Un mensaje del protocolo CoAP tiene una longitud de 4 bytes, donde consta información sobre la versión, el tipo de mensaje (CON, NO, ACK,

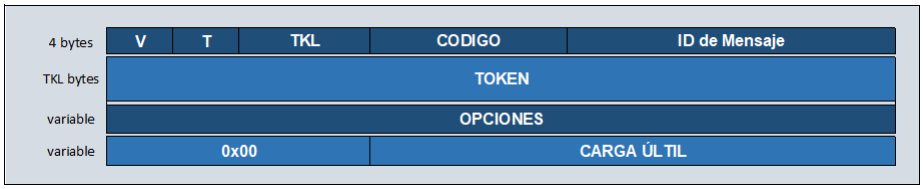


Figura 2.5: Estructura del mensaje - CoAP Fuente: [Shelby et al., 2014] Elaboración propia

RST), un id de identificación del mensaje, opciones utilizadas es CoAP como el número de puerto, host o cadena de consulta, y la carga de información del mensaje.

Características

Las características del protocolo CoAP son muy similares a HTTP, sin embargo este no debe ser considerado como una versión limitada de este último, ya que es específicamente optimizado para IoT, en particular para comunicaciones M2M.

Las principales características de este protocolo son las siguientes.

- Protocolo web restringido que cumple con los requerimientos M2M.
- Estándar de seguridad DTLS (Datagram Transport Layer Security).
- Intercambio de mensajes asíncronos.
- Baja sobrecarga en encabezado (paquetes pequeños).
- Soporte de URI (Uniform Resource Identifier).
- Uso de proxy y capacidades de captura simple.
- Enlace fiable UDP con capacidad de difusión simple y múltiple.
- Acceso a recursos CoAP con interfaz HTTP.

2.4.2. Protocolo MQTT

Message Queuing Telemetry Transport (MQTT), es un protocolo M2M de la capa de aplicación que transporta mensajes entre dispositivos en base

al esquema publicador/subscriptor. Es muy liviano y por el uso de tópicos de identificación facilita una implementación sencilla.

Este protocolo realiza la transmisión de datos por medio del protocolo TCP, a diferencia de otros que están basados en UDP. Esto lo convierte en un protocolo con características diferentes, que a pesar que UDP es más rápido, simple y más eficiente, la simplicidad de MQTT en alguna forma compensa este aspecto, además de proveer un control de calidad de servicio (QoS) mejor que los protocolos basados en UDP, es decir, la confiabilidad es mayor por el hecho que TCP soporta retransmisión de paquetes perdidos[Hejazi et al., 2018].

La información es organizada en niveles jerárquicos mediante el uso de tópicos, lo cual facilita la difusión de mensajes particulares como a grupos.

Este protocolo es uno de los más utilizados en el ámbito IoT debido a su ligereza, sencillez y confiabilidad. Muchas aplicaciones IoT la han adoptado para proveer soluciones industriales robustas.

Modelo de comunicación

El esquema de comunicación de este protocolo se basa en el uso de dos tipos de entidades, Bróker y Dispositivos.

El "Bróker" es un software que se ejecuta en un dispositivo, normalmente un servidor, el cual se encarga del filtrado de mensajes y el envío de los mismos a los subscriptores correspondientes registrados.

Los "Dispositivos" son los clientes que pueden recibir y enviar mensajes al Bróker para su posterior difusión a otros clientes.

En la figura 2.6 se muestra el esquema general de comunicación del protocolo MQTT.

Para filtrar los destinatarios de los mensajes, este protocolo utiliza "Tópicos" los cuales son strings UTF-8 que el bróker utiliza para identificar los clientes subscritos. Un tópico consiste en niveles, cada nivel es separado con el carácter "/".

- Nivel1/Nivel2/Nivel3/Nivel4

Los niveles jerárquicos proveen la funcionalidad de difundir un mensaje a un nivel específico, o mediante el uso de comodines es posible enviar mensajes a todos los subscriptores cuyos niveles están más profundo que el nivel del mensaje enviado.

Los comodines pueden ser de dos tipos, (carácter numeral), que difunde mensajes multinivel inferior, y '+' (carácter de adición) que convierte un nivel en comodín, es decir a un nivel único.

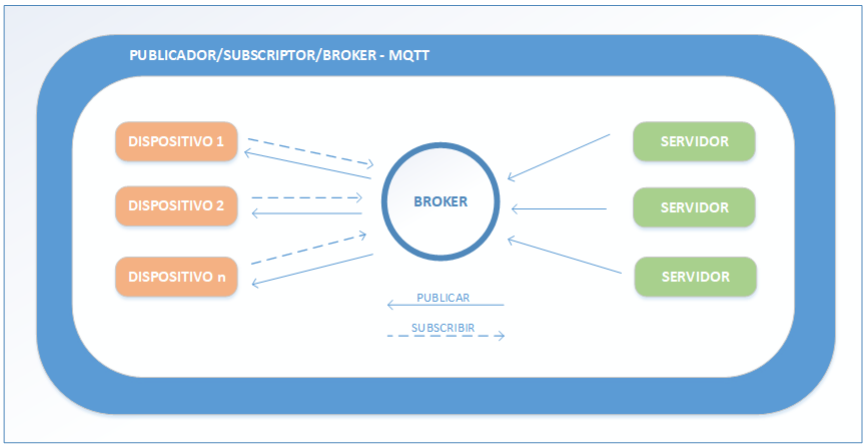


Figura 2.6: Comunicación Publicador/Subscriptor - MQTT. Fuente: [Hejazi et al., 2018] Elaboración propia

A continuación, se muestra dos ejemplos del uso de comodines multinivel y simple nivel (+).

- Edificio/Sala1/EquiposControl/Temperatura/
- Edificio/+/EquiposControl/Temperatura

En el primer ejemplo se subscribirá a los mensajes cuyos destinos se encuentran en el Edificio, Sala1, EquiposControl dentro del ámbito equipos de medición de temperatura y todos sus subniveles.

En el segundo ejemplo se subscribirá a los mensajes cuyos destinos tienen los equipos de medición de Temperatura que se encuentran en el Edificio, en cualquier sala pertenecientes a los Equipos de Control.

Estructura del mensaje

Los mensajes constan de tres partes: Cabecera fija, Cabecera variable y Contenido.

Cabecera fija: Sirve para la identificación del tipo de mensaje enviado, además de la longitud del mismo. Se utilizan 2 bytes, cuyos 7 primeros bits corresponden a la cabecera, y el último para control de continuidad.

Cabecera variable: Es de uso opcional para situaciones especiales en las que se requiere del manejo de información adicional.

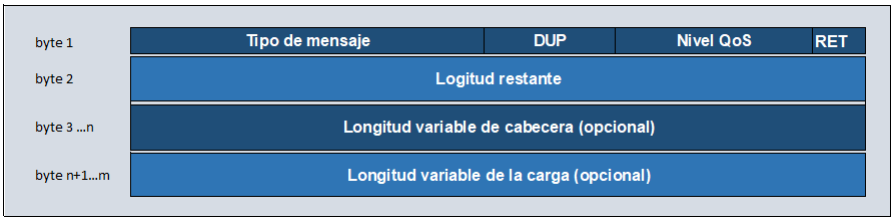


Figura 2.7: Estructura del mensaje - MQTT. Fuente: [Yassein et al., 2017] Elaboración propia

Contenido: Es el contenido del mensaje, el cual puede almacenar hasta 256Mb. En aplicaciones reales se maneja capacidades entre 2 a 4 kB.

Características

Las principales características de este protocolo se resumen a continuación.

- Protocolo abierto que cumple con los requerimientos M2M.
- Seguridad a nivel de capa de transporte usando SSL/TLS.
- Intercambio de mensajes asíncronos.
- Difusión múltiple de subscripción y publicación independiente de la aplicación.
- Flujo de datos optimizado para reducir el tráfico de red.
- Soporte de comunicación Publicadores-Subscriptores.
- Soporta tres niveles de servicio QoS.
- Notificaciones de desconexiones inesperadas.

Niveles QoS

Los clientes MQTT tienen la posibilidad de seleccionar el nivel de calidad de servicio que se utilizará para las acciones de Publicación y Subscripción. Esta tarea es exclusiva del cliente donde este debe de forma obligatoria seleccionar uno de los tres niveles de calidad a utilizar durante el envío-recepción de los mensajes. Los niveles soportados son:

- QoS 0: Los mensajes son enviados como máximo una vez; no se garantiza la recepción, y no son reenviados. Este es el modo menos confiable pero el más rápido, y se suele utilizar en procesos que envían información regular, en el que uno o dos mensajes que no sean recibidos, no influirá significativamente en el resultado final.

- QoS 1: Los mensajes son enviados por lo menos una vez; es más fiable que el nivel QoS 0 pero es más lento. Aquí el publicador vuelve a enviar un mensaje cada cierto tiempo mientras no reciba un acuso de recibo (ACK) proveniente del subscriptor. Este nivel de calidad es usualmente utilizado en configuraciones realizadas sobre equipos, en el que se setea un valor específico.

- QoS 2: Los mensajes son entregados al receptor exactamente una vez. Este esquema es utilizado cuando se desee entregar solamente una copia del mensaje al subscriptor.

2.4.3. Protocolo AMQP

Advanced Message Queuing Protocol, es otro protocolo diseñado para comunicaciones M2M, generalmente utilizado en ambientes corporativos. Es utilizado en aplicaciones de envío y recepción de mensajes, soporta el modelo publicador-subscriptor[Vinoski, 2006].

La característica que diferencia a este protocolo es la capacidad de almacenar mensajes en una cola, permitiendo la comunicación asíncrona, es decir, el transmisor y receptor no deben actuar al mismo tiempo. El punto fuerte de este protocolo es la interoperabilidad ya que trabaja a nivel muy bajo en la comunicación, el cual envía datos en forma de flujo de bytes, ofreciendo una comunicación punto a punto mucho más simple.

La diferencia entre el bróker de MQTT Y AMQP es que en este último está compuesto de un Intercambiador y de colas de tópicos.

Modelo de comunicación

Este protocolo proporciona el modelo de comunicación que se presenta en la figura 2.8. Un cliente llamado “Productor” publica un mensaje al intercambiador, luego este recibe el mensaje y es ahora responsable de enrutar el mensaje. La vinculación entre el intercambiador y las colas se lo realiza en función del número de colas disponible y reglas definidas. El mensaje permanece en las colas hasta que sean manejadas por el subscriptor.

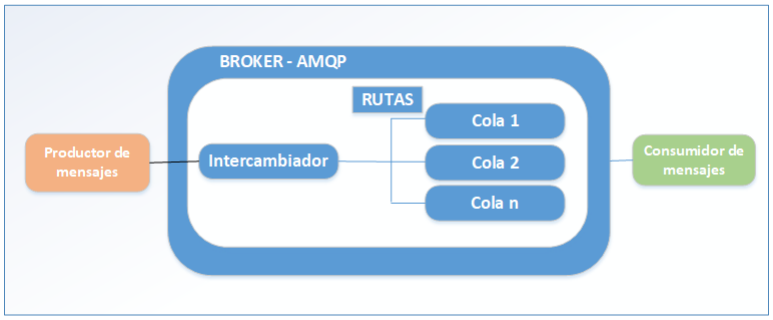


Figura 2.8: Modelo de comunicación - AMQP. Fuente: [Priyadarshi and Behura, 2018] Elaboración propia

Las reglas que aplica el intercambiador se realiza dependiendo del tipo y la llave de ruteo contenida en el mensaje.

Los tipos de intercambiador que soporta este protocolo son los siguientes:

- Intercambiador directo: Envía mensajes a las colas basados en una llave de ruteo contenida en el propio mensaje. Esta llave es agregada por el productor y está contenida en la cabecera.
- Intercambiador por defecto: Es un intercambiador pre-declarado el cual no tiene nombre y es referido por un string vacío. Al utilizar este intercambiador, los mensajes son enrutados a la cola cuyo nombre es igual al nombre contenido en la llave de ruteo del mensaje.
- Intercambiador basado en tópicos: Este enruta los mensajes basado en coincidencias de comodines entre la llave de ruteo y el patrón de ruteo, lo cual especifica la vinculación de las colas que recibirán el mensaje. El uso de este tipo de intercambiador hace que el modelo de comunicación de este protocolo se asemeje al utilizado en MQTT (basado en tópicos).
- Intercambiador abanico: Realiza copias del mensaje y las difunde a todos las colas registradas, sin realizar ningún tipo de filtrado, independientemente de la llave de ruteo o coincidencia de patrones. Este intercambiador es útil para realizar difusiones de información a todas las colas.
- Intercambiador cabecera: Enruta los mensajes basados en argumentos o cadenas de texto que contienen cabeceras y valores opcionales. Este

es muy similar al intercambiador de tópico con la diferencia que realiza el ruteo basado en valores de cabecera en lugar de llaves de ruteo.

- Intercambiador de letra muerta: Si no existe una cola de coincidencia para un mensaje, el mensaje es eliminado.

Colas de mensajes: Una cola es una estructura de datos secuencial donde se puede realizar dos operaciones principales: Agregar y Obtener mensajes. Las colas pueden utilizar dos mecanismos de almacenamiento y captura que son primero en llegar primero en salir (FIFO), o último en llegar primero en salir (LIFO). Lo más común es el uso FIFO en este protocolo. Las colas tienen nombre para que puedan ser identificadas por aplicaciones e intercambiadores.

Estructura del mensaje

La estructura del mensaje en el protocolo AMPQ consta de las secciones que se describen a continuación: Cabecera: Es un conjunto de datos relacionados con el envío de los mensajes, donde consta el tiempo de vida, durabilidad y prioridades.

Anotaciones de envío: Esta sección es utilizada para registrar información que no está definida en el estándar, la cual no es posible colocar en la cabecera. Es información utilizada por implementaciones de protocolo para enviar información característica de dicha implementación.

Anotaciones de mensaje: Aquí se puede almacenar algunas propiedades personalizadas de los mensajes, utilizados principalmente en la conversión del mensaje.

Propiedades: En esta sección se almacena propiedades estandarizadas definidas por ISO/IEC.

Propiedades de aplicación: Se refiere a propiedades de datos de la aplicación. Pie: Contiene detalles sobre el mensaje o el proceso de envío que puede ser calculado solo luego que el mensaje desnudo ha sido construido.

El formato de mensaje AMPQ se muestra en la figura 2.9.

Las secciones Propiedades, Propiedades-Aplicación y Cuerpo del Mensaje forman parte del denominado “Mensaje Desnudo” el cual contiene el mensaje exacto, es decir no puede cambiar, mientras que los otras secciones que en conjunto forman el mensaje anotado puede contener propiedades que pueden cambiar durante el proceso de envío.

Figura 2.9: Estructura del mensaje - AMQP. Fuente: [Vinoski, 2006] Elaboración propia

2.4.4. Estándar DDS

Data Distribution Service, es un estándar a nivel de la capa de aplicación, el cual es una especificación para dispositivos puente (middleware) que trabajan en el esquema publicador/subscriptor. Es un estándar abierto para aplicaciones de tiempo real [Yang et al., 2012].

Trabaja en la capa de aplicación y utiliza UDP como su método de transporte, lo cual proporciona velocidad y ligereza durante el envío de mensajes.

Esta especificación define tanto la Interfaz de Aplicación (API), como la semántica de comunicación que se encarga de la calidad de servicio, así como del desempeño de alto rendimiento en tiempo real de la comunicación Máquina-a-Máquina.

El propósito general de esta especificación puede ser resumida en el envío eficiente y robusto de la información correcta al momento correcto.

A diferencia del protocolo MQTT que es optimizado para la colecta y análisis de datos centralizado, la especificación DDS es optimizada para el procesamiento distribuido, por ejemplo, conectar directamente sensores, dispositivos y aplicaciones entre sí, sin la dependencia de una infraestructura centralizada.

La especificación está diseñada para permitir una separación entre los publicadores y receptores; así, un proceso de una aplicación que esté del lado del publicador, solo participará en tareas de publicación, similarmente un proceso que esté en el lado del subscriptor solamente participará en tareas relacionadas al subscriptor.

Modelo de comunicación

La especificación DDS utiliza dos niveles de interfaces, DCPS (Data-Centric Publish-Subscribe) que tiene como tarea realizar un reparto de la información en forma balanceada a los receptores apropiados, y DLRL

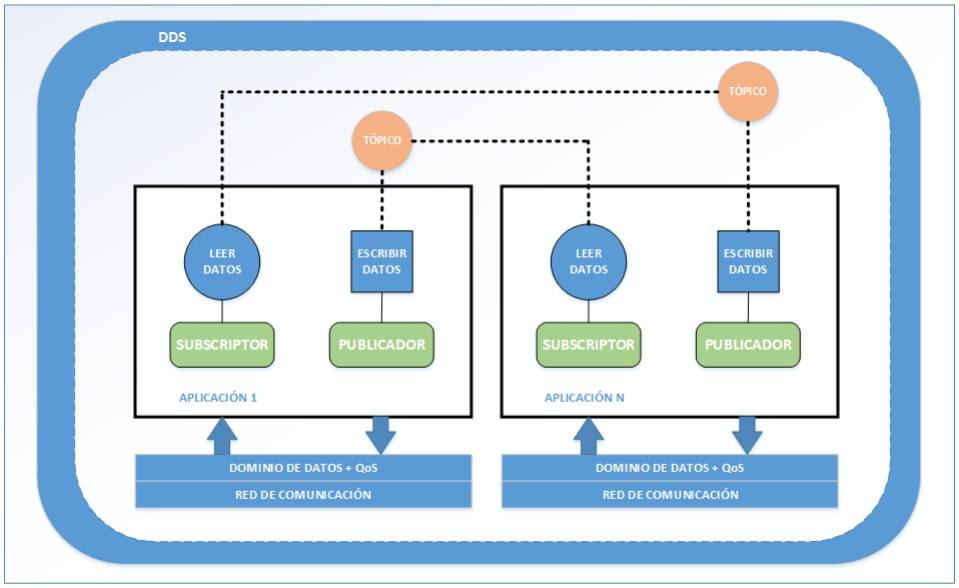


Figura 2.10: Modelo de comunicación - DDS. Fuente:[Yang et al., 2012] Elaboración propia

(Data Local Reconstruction Layer), que permite una integración sencilla en la capa de aplicación.

Esta especificación es orientada a sistemas con gran número de nodos, ya que su arquitectura fue diseñada para soportar un sistema de comunicación optimizado para el manejo de gran número de elementos.

Se elimina el concepto de "Bróker" manejado en protocolos antes revisados, ya que considera que este representa un punto único de falla y agrega valor de latencia de acuerdo a la carga del bróker.

DDS soluciona este problema introduciendo el uso de un bus de Dominio de Datos, al cual se conectan las Aplicaciones o puntos finales estableciendo enlaces lógicos a cada una de ellos.

Se establece un proceso de descubrimiento de elementos para realizar la conexión lógica con el bus, lo cual facilita significativamente el manejo de grandes cantidades de dispositivos conectados.

La forma en que se produce la comunicación es la siguiente: Los nodos luego de ser descubiertos por el sistema son catalogados como una aplicación, la cual mediante el uso de tópicos puede subscribir o publicar información dirigida a una o más aplicaciones. La o las aplicaciones destino podrán a

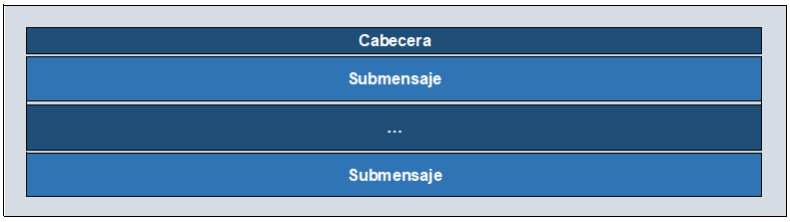


Figura 2.11: Estructura de un mensaje DDS. Fuente: [Corsaro, 2014] Elaboración propia

su vez enviar datos a la aplicación original o de ser necesario establecer una comunicación con otras aplicaciones diferentes para el intercambio de información.

Las aplicaciones podrán ser ejecutadas en el mismo sistema o en diferentes, es decir, se puede distribuir la comunicación entre varios elementos de la red con lo cual se elimina el enfoque centralizado sin depender enteramente de un solo nodo. Esto es posible mediante el uso de la red de comunicación, que trabaja en conjunto con el bus de datos, donde posibilita que el proceso de comunicación entre aplicaciones funcione de manera similar ya sea que éstas residan en el mismo hardware o estén en diferentes puntos de la red.

Debido a que DDS utiliza UDP en la capa de transporte, es necesario agregar la funcionalidad ACK para no dejar de lado la fiabilidad de la red. Esta funcionalidad también es tarea de la capa "Red de Comunicación" que implementa este protocolo, esto posibilita manejar acuso de recibo de paquetes (ACK) para realizar el reenvío de paquetes perdidos.

Estructura del mensaje

La estructura de los mensajes DDS constan de una cabecera y un número variable de submensajes RTPS [Corsaro, 2014], como se muestra en la figura 2.11.

En la figura 2.12 consta la estructura de la cabecera.

La cabecera RTPS tiene una longitud de 20 bytes conformado por los siguientes identificadores:

- RTPS: Contiene datos que identifican al mensaje como RTPS. Tiene una longitud de 4 bytes.
- Versión del protocolo: Indica la versión del protocolo RTPS utilizado. Tiene una longitud de 2 bytes.

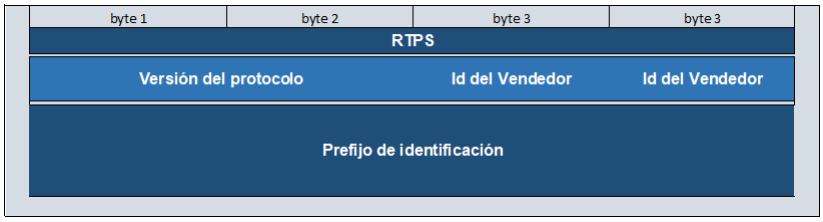


Figura 2.12: Estructura de la cabecera RTPS. Fuente:[Corsaro, 2014] Elaboración propia

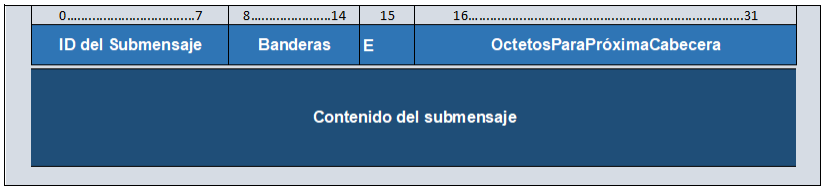


Figura 2.13: Estructura de un submensaje RTPS. Fuente:[Corsaro, 2014] Elaboración propia

- Id del Vendedor: Contiene la identificación del fabricante.
- Prefijo de identificación: Define un prefijo que identifica en forma única las entidades RTPS que aparecen en el mensaje.

El formato del submensaje consta de los siguientes campos:

- Id del submensaje: Identifica el tipo de submensaje.
- Banderas: Este campo guarda información correspondiente al formato utilizado para almacenar los datos.
- E: Nomenclatura que indica la forma en que se almacena la información de la bandera. E=0, indica que el bit más significativo se guarda primero; E=1, indica que el bit menos significativo se guarda primero.
- Octetos Para Próxima Cabecera: Indica el largo del submensaje, es decir,con esta información se conoce donde inicia la siguiente cabecera del siguiente submensaje.

Características

Las principales características de este estándar son las siguientes:

- Protocolo abierto que cumple con los requerimientos M2M.
- Basado en funcionalidad DataCentric, es decir, datos disponibles para aplicaciones interesadas en ellos.
- Intercambio de mensajes asíncronos y síncronos.
- Descubrimiento de nodos en forma automática.
- Infraestructura descentralizada, manejo de fallos y escalable.
- Soporte comunicación Publicadores-Subscriptores y QoS
- Manejo de publicadores y subscriptores en forma independiente.
- Define dos niveles de interface, DLR y DCPS.
- Implementa protocolo RTPS para proveer interoperabilidad con la capa de transporte.

2.4.5. Protocolo TCP

Transmision Control Protocol, en un protocolo orientado a la conexión y su función es garantizar que se produzca la comunicación en forma independiente de las capas inferiores del modelo TCP/IP.

Tiene un mecanismo avanzado de control de errores, utiliza la técnica de ventana deslizante para que los segmentos de los paquetes lleguen correctamente. Además, garantiza que los paquetes lleguen en el mismo orden en que fueron enviados

Permite realizar el control de flujo, esto es, la capacidad de adaptarse a distintas velocidades de transmisión y recepción, contribuyendo al manejo más eficiente del ancho de banda. En conjunto con el control de congestión que controla la pérdida de paquetes, aportan fiabilidad a este protocolo.

2.4.6. Protocolo UDP

User Datagram Protocol, a diferencia de TCP no es un protocolo orientado a la conexión, permite la transmisión sin conexión de datagramas. En este protocolo el envío de información es muy rápida, ya que no es necesario

establecer una comunicación con el receptor ni esperar una respuesta. Utiliza puertos para realizar la transmisión de datos a las aplicaciones del sistema. La transmisión mediante UDP no garantiza que los paquetes lleguen completos ni en el orden en que fueron enviados, tampoco garantiza protección frente a ataques externos.

La fortaleza de este protocolo es su sencillez y ligereza, que permite transmitir información en forma rápida en redes IP. Se usa principalmente en aplicaciones de video y multimedia, pero también se ha extendido su uso a otras aplicaciones que requieren fiabilidad de datos con la implementación de otros protocolos que cubran las falencias de fiabilidad de UDP.

2.4.7. Protocolo IP, ARP, ICMP

El principal protocolo utilizado es esta capa es IP, el cual implementa las direcciones lógicas de los elementos de red, denominada dirección IP. Esta dirección será utilizada por otras capas de nivel más alto para proveer el ruteo entre redes. Otra tarea del protocolo IP es establecer la comunicación Dispositivo a Dispositivo, para lo cual determina la ruta por la cual los datos deben ser transmitidos.

Otro protocolo de esta capa es ARP (Address Resolution Protocol), el cual es utilizado para encontrar la dirección física de los dispositivos a partir de su dirección IP.

El protocolo ICMP (Internet Control Message Protocol), el cual provee una funcionalidad que es utilizada por los equipos terminales y routers para enviar el origen y notificaciones sobre datagramas con problemas[RiLi, 2011].

Capítulo 3

Estudio de productos comerciales

Este capítulo tiene como objetivo analizar diversas opciones comerciales que existen en el mercado para ofrecer el uso de plataformas IoT. Este estudio servirá de punto de referencia, especialmente en costos, para ser comparada con la plataforma propuesta en este proyecto de titulación.

A continuación se muestran los principales productos comerciales que, al momento del desarrollo de este trabajo, son las más populares y por lo tanto nos darán un indicativo válido de referencia para la comparación.

En cada plataforma estudiada se obtendrá un cuadro estandarizadode costos en función del número de dispositivos terminales soportados. Se analizan soluciones de 10, 100, 200, 500, 1000 y 5000 dispositivos supervisados. Esto servirá para generar un reporte comparativo.

3.1. Google Cloud IoT

Es una plataforma que ha reunido las herramientas web que Google estaba ofreciendo por separado o como productos autónomos, con lo cual ha permitido la facilidad de proveer soluciones de distintos ámbitos a través de computación en la nube. Entre los productos ofrecidos consta Google Cloud IoT, que por medio de IoT Core provee un servicio completamente administrado que permite conectar dispositivos IoT para recopilar datos, procesarlos y obtener información relevante por medio de reportes y alertas. Google Cloud IoT tiene un esquema de facturación que va en función del volumen de datos usados en un período mensual. La figura 3.1 muestra los valores correspondientes.

Volumen de datos mensual	Precio por MB	Dispositivos registrados	Cargo mínimo*
Hasta 250 MB	$0.00	Ilimitado, dentro de la cantidad máxima de QPS	1,024 bytes
De 250 MB a 250 GB	$0.0045	Ilimitado, dentro de la cantidad máxima de QPS	1,024 bytes
De 250 GB a 5 TB	$0.0020	Ilimitado, dentro de la cantidad máxima de QPS	1,024 bytes
A partir de 5 TB	$0.00045	Ilimitado, dentro de la cantidad máxima de QPS	1,024 bytes

Figura 3.1: Costos del uso de plataforma Google Cloud IoT core. Fuente: [Cloud, 2021]

Cualquier mensaje de menos de 1024 bytes, se cobra como que tuviese ese tamaño.

Bajo este esquema, y para tener una idea más clara, en la página web de Google Cloud presenta unos ejemplos de facturación, los cuales se los va a citar.

- El primer ejemplo muestra el valor a cancelar por el uso de 10.000 dispositivos que no están transmitiendo lecturas, sino únicamente los mensajes PINGREQ que son utilizados para verificar que dichos dispositivos están en estado activo en la red. Es decir, este ejemplo muestra el escenario más básico.

  10.000 dispositivos * 96 PINGREQ por día * 30 dias * 1024 bytes = 27.47 GB.

  Si se considera un volumen de datos gratuitos de 250mb, el costo total para este escenario sería:

  (27.47GB – 250 MB) * 0,0045 =125.46

- El segundo ejemplo se contempla el envío de mensajes pequeños por parte de los dispositivos, con una carga de un mensaje pequeño de menos de 1024 bytes cada 30 minutos. Cabe indicar que el tamaño mínimo de un mensaje es considerado 1024 bytes, así estos ocupen una menor longitud.

  Se propone utilizar los mismos 10.000 dispositivos. El cálculo del costo mensual sería el siguiente:

  10.000 dispositivos * 48 mensajes/dispositivo por día * 30 dias * 1024 bytes = 14 GB

Tabla 3.1: Precios estándar plataforma Google Cloud IoT.
Elaboración propia.

Dispositivos	***Nivel1*** $*$	***Nivel2*** $*$	***Seleccionado***
10	$43,31	$225,56	$225,56
50	$225,56	$1.136,81	$1.136,81
100	$453,38	$2.275,88	$2.275,88
200	$909,00	$2.024,00	$2.024,00
500	$2.275,88	$5.061,50	$5.061,50
1000	$2.024,00	$10.124,00	$10.124,00
5000	$10.124,00	$11.390,40	$11.390,40

Si se considera un volumen de datos gratuitos de 250mb, el costo total para este escenario sería:

(421.87GB – 250Mb) * 0,0045 =1943.99

En resumen, para completar la tabla estándar de comparación se calculan los valores como lo muestra la tabla 3.1. Las columnas 'Nivel1' y 'Nivel2' denotan un mensaje cada cinco segundos y un mensaje cada segundo respectivamente.

3.2. AWS IoT

La plataforma Amazon Web Services (AWS), es una colección de servicios basados en computación en la nube, tienen un amplio portafolio de productos que pueden ir desde herramientas de análisis, aplicaciones financieras, herramientas para desarrolladores, juegos, almacenamiento, así como Internet de las Cosas.

La plataforma AWS IoT Core, permite el uso de dispositivos preparados con conectividad en la nube, los cuales son fabricados por proveedores que los preparan para hacerlos compatibles con esta tecnología. Hace uso de un Gateway, el cual funciona como un punto de entrada para los dispositivos compatibles con IoT que se conectan a AWS.

Dispone de un SDK (Software Development Kit) para dispositivos con AWS IoT, el cual permite conectar, autenticar e intercambiar mensajes con AWS IoT Core (Server) mediante el uso de los protocolos MQTT, HTTP con WebSockets. Esta librería está soportada en C, JavaScript y Arduino.

Número de dispositivos activados	Precio mensual por dispositivo activado
De 1 a 14 999	0,25 USD
De 15 000 a 74 999	0,20 USD
De 75 000 a 99 999	0,15 USD
100 000 y más	0,10 USD

Figura 3.2: Costos del uso de plataforma AWS IoT. Fuente: [Amazon Web Services, 2021]

Para realizar tareas de Publicación/Subscripción utiliza un agente de mensajes que realiza la transmisión de datos en forma segura hacia y desde aplicaciones y dispositivos IoT.

Tiene un esquema de pagar lo que se utiliza, es decir, se cobra en función del número de dispositivos activados que utilicen esta plataforma. Este puede ser administrada desde la consola de administración AWS, aplicación móvil o uso de API.

El esquema de precios se muestra en la figura 3.2, a partir de la cual se obtiene la tabla de costos estándar mostrada en la tabla 3.2.

Además del cargo de mensajería es necesario incluir cargos por la conectividad, la cual para Latinoamérica tiene un costo de 12 centavos de dólar por millón de minutos de conexión.

La tabla 3.2 muestra el cuadro estándar calculado con dos escenarios, el primero considerando la lectura de un mensaje cada 5 segundos por cada dispositivo, mientras que el segundo escenario considera la lectura de un mensaje por cada segundo. La columna selección es la que se utilizará en la comparativa final ya que se propone para este ejercicio una actualización de datos cada segundo.

3.3. IBM Watson IoT

En forma general IBM Watson es la propuesta de soluciones de Inteligencia Artificial para negocios ofrecida por la compañía IBM, lo cual ayuda a las organizaciones a predecir ganancias o pérdidas, facilita la automatización de procesos complejos, organizar las actividades de empleados, y todas aquellas aplicaciones relacionadas con inteligencia artificial.

Luego agregaron a este conjunto de herramientas el soporte de IoT Core,

Tabla 3.2: Precios estándar plataforma AWS IoT.
Fuente: Elaboración propia.

Dispositivos	***Nivel1*** $*$	***Nivel2*** $*$	***Seleccionado***
10	$13,01	$64,85	$64,85
50	$65,06	$324,26	$324,26
100	$130,12	$648,52	$648,52
200	$260,24	$1.297,04	$1.297,04
500	$650,59	$3.242,59	$3.242,59
1000	$1.301,18	$6.485,18	$6.485,18
5000	$6.505,92	$32.425,92	$32.425,92

Tabla 3.3: Precios estándar plataforma IBM Watson IoT.
Elaboración propia.

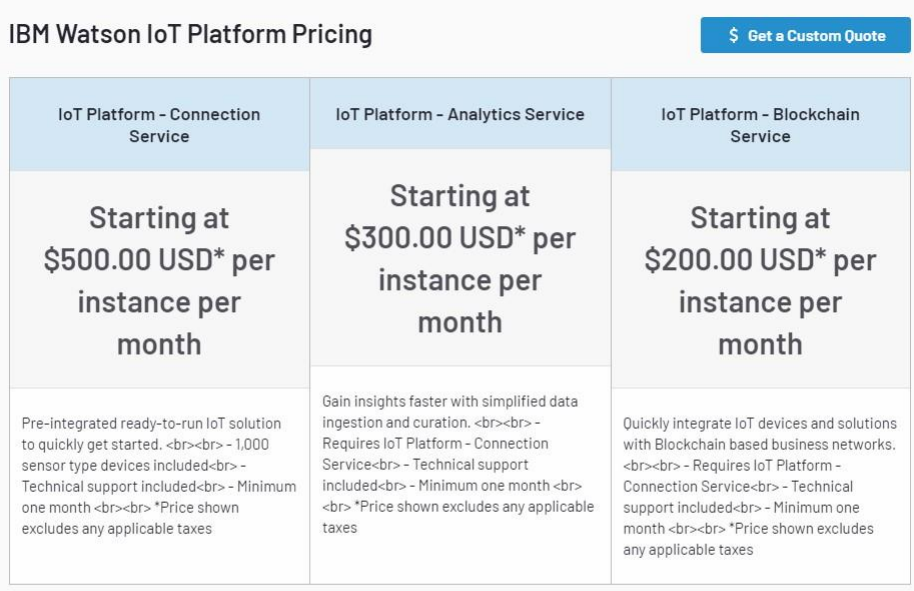
Dispositivos	***Nivel1*** $*$	***Nivel2*** $*$	***Seleccionado***
10	$310,00		$310,00
50	$350,00		$350,00
100	$400,00		$400,00
200	$500,00		$500,00
500	$800,00		$800,00
1000	$1.300,00		$1.300,00
5000	$5.300,00		$5.300,00

con lo cual permiten la administración de Internet de las Cosas, permitiendo a billones de dispositivos alrededor del mundo conectarse al Internet y proveer y compartir información, facilitando así el acceso a datos en vivo o históricos, con lo cual se puede realizar análisis para facilitar la toma de decisiones.

El esquema de costos para la plataforma IoT se la resume en la tabla 3.3.

3.4. Kaa IoT Platform

El Proyecto Kaa es una plataforma middleware que se basa en computación en la nube (PaaS), es decir, plataforma como un servicio. Esta plataforma es multi propósito y flexible de código abierto para la implementación de soluciones IoT, aplicaciones compartidas y productos inteligentes.



Cantidad de elementos	Precio unitario para todos los elementos
Nivel 1: 1 - 1000	$1 USD
Nivel 2: 1001 - 2000	$0,90 USD
Nivel 3: 2001 - 3000	$0,75 USD
Nivel 4: 3001 - 4000	$0,60 USD
Nivel 5: > 4000	$0,40 USD

Figura 3.3: Costos del uso de plataforma IBM Watson IoT. Fuente: [Helen, 2021]

La Figura 3.4 muestra el esquema de precios de la plataforma Kaa IoT, y en la Tabla 3.4 constan los precios estándar obtenidos para la posterior comparación.

3.5. Microsoft Azure IoT

Es una Plataforma de servicios a través de la red para conectar, supervisar y controlar actividades IoT. Esta plataforma tiene el siguiente esquema de facturación:

Los dos primeros dispositivos sin costo, luego aproximadamente 0,39 por cada dispositivo adicional.

Se ofrece una cuenta gratuita por 30 días.

Figura 3.4: Esquema de precios de la plataforma Kaa IoT. Fuente: [KaaIoT Technologies, 2021]

Tabla 3.4: Precios estándar plataforma Kaa IoT.
Elaboración propia.

Dispositivos	***Nivel1*** $*$	***Nivel1*** $*$ ***Seleccionado***
10	$14,99	$14,99
50	$44,99	$44,99
100	$79,99	$79,99
200	$175,00	$175,00
500	$325,00	$325,00
1000	$625,00	$625,00
5000	ND	ND

La Tabla 3.5 muestra los precios correspondientes a los distintos números de elementos supervisados.

3.6. Ubidots IoT

Es una plataforma IoT que permite implementar soluciones IoT de la industria en general. Utiliza computación en la nube bajo el esquema PaaS, mediante la cual obtiene, procesa y provee información de equipos terminales o sensores del cliente.

Tabla 3.5: Precios estándar plataforma Microsoft Azure IoT.
Elaboración propia.

Dispositivos	***Nivel1*** $*$	***Nivel1*** $*$	***Total***	***IoT HUB***	***T OT AL***
10	$0,62	$3,14	$5,49	$2.500,00	$2.505,49
50	$3,75	$18,82	$32,94	$2.500,00	$2.532,94
100	$7,65	$38,42	$67,25	$2.500,00	$2.567,25
200	$15,47	$77,62	$135,87	$2.500,00	$2.635,87
500	$38,90	$195,22	$341,73	$2.500,00	$2.841,73
1000	$77,95	$391,23	$684,83	$2.500,00	$3.184,83
5000	$390,39	$1.959,27	$3.429,63	$2.500,00	$5.929,63

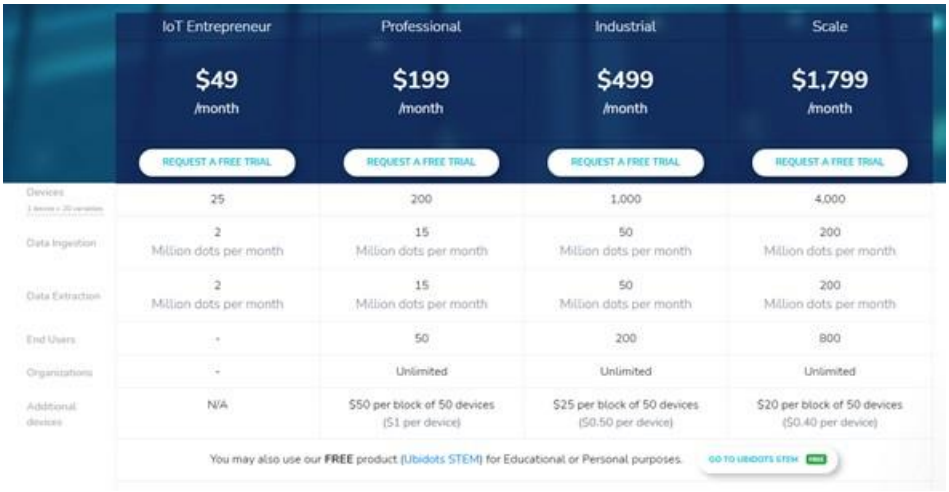


Figura 3.5: Costos del uso de plataforma Ubidots IoT. Fuente: [Ubidots, 2021]

La figura 3.5 indica los planes de precios ofrecidos, cuyo costo se incrementa en función del número de dispositivos soportados, mientras que la Tabla 3.6 muestra el esquema de precios estándar en función del número de elementos supervisados.

Tabla 3.6: Precios estándar plataforma Ubidots IoT.
Elaboración propia.

Dispositivos	***Nivel1*** ∗	***Nivel2*** ∗	***Seleccionado***
10	$49,00		$49,00
50	$199,00		$199,00
100	$199,00		$199,00
200	$199,00		$199,00
500	$499,00		$499,00
1000	$499,00		$499,00
5000	$2.199,00		$2.199,00

Tabla 3.7: Cuadro Comparativo de Precios - Plataformas IoT.
Elaboración propia.

Dispositivos	***GoogleCloud***	***AWS***	***IBMW atson***	***MicrosoftAzure***	***Kaa***	***Ubidots***
10 disp	$225,56	$64,85	$310,00	$2.505,49	$14,99	$49,00
50 disp	$1.136,81	$324,26	$350,00	$2.532,94	$44,99	$199,00
100 disp	$2.275,88	$648,52	$400,00	$2.567,25	$79,99	$199,00
200 disp	$2.024,00	$1.297,04	$500,00	$2.635,87	$175,00	$199,00
500 disp	$5.061,50	$3.242,59	$800,00	$2.841,73	$325,00	$499,00
1000 disp	$10.124,00	$6.485,18	$1.300,00	$3.184,83	$625,00	$499,00
5000 disp	$11.390,40	$32.425,92	$5.300,00	$5.929,63	ND	$2.199,00

3.7. Comparativas de costos de servicio

Una vez que se ha investigado sobre las principales plataformas IoT que se encuentran disponibles actualmente en el mercado, y al contar con una tabla de precios estandarizada de cada proveedor, se procede a realizar una tabla única para comparar los precios de servicio entre ellas.

La Tabla 3.7 muestra el resumen general de precios propuestos por las distintas soluciones.

Al analizar la Figura 3.6, se nota que a mayor cantidad de dispositivos soportados las diferencias de precios entre las distintas soluciones se amplían, en el caso de 5000 dispositivos el mejor costo ofrece la plataforma de IBM a 2000 mientras que el mayor costo corresponde a la plataforma AWS de Amazon con 32000.

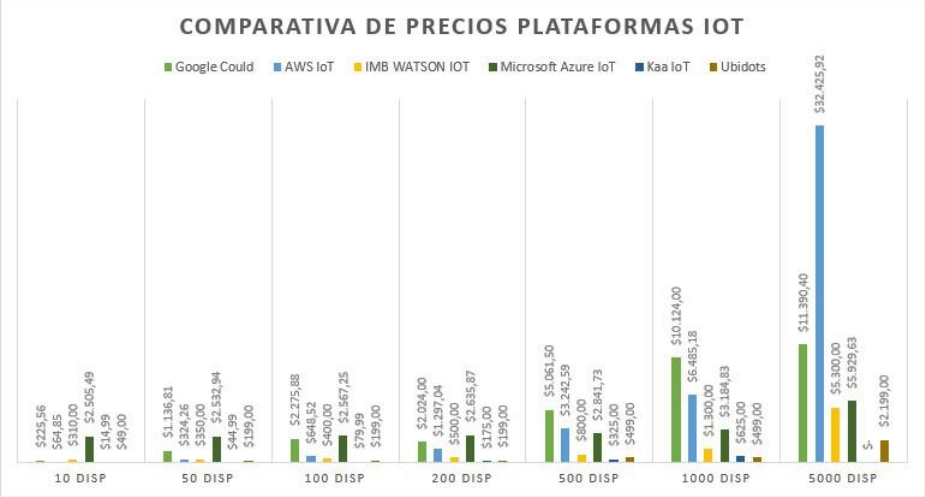


Figura 3.6: Gráfica comparativo de Precios – Plataformas IoT. Elaboración propia.

Capítulo 4

Construcción del servidor IoT

4.1. Alcance

El servidor IoT es el encargado de proveer al usuario un sistema para el monitoreo de variables correspondientes a dispositivos terminales que se encuentran en las instalaciones de los clientes, o industria. Estos dispositivos terminales pueden ser sensores, PLC, o cualquier otro dispositivo que contenga información de un proceso.

4.2. Estructura General

El modelo general de comunicación se muestra en la figura 4.1, en el que se especifica la distribución de los dispositivos que forman parte de esta solución, iniciando con los equipos terminales como sensores o PLC, hasta el servidor IoT.

- Equipos Cliente. - Es todo el equipamiento que conforma la red del cliente, el cual puede estar conformado por los siguientes elementos:

  Dispositivos terminales: Están compuestos por sensores, PLC o cualquier otro tipo de dispositivos de medición.

  Switch/Router: Son equipos que conforman la red LAN donde estarán conectados los PLC y otros dispositivos de red.

  DAQ (Data Adquisition): Es un dispositivo de adquisición de datos el cual realizará tareas de lectura de datos de dispositivos terminales, con la respectiva publicación de datos hacia el bróker MQTT que se encuentra funcional en el servidor IoT. El tipo de dispositivo que se

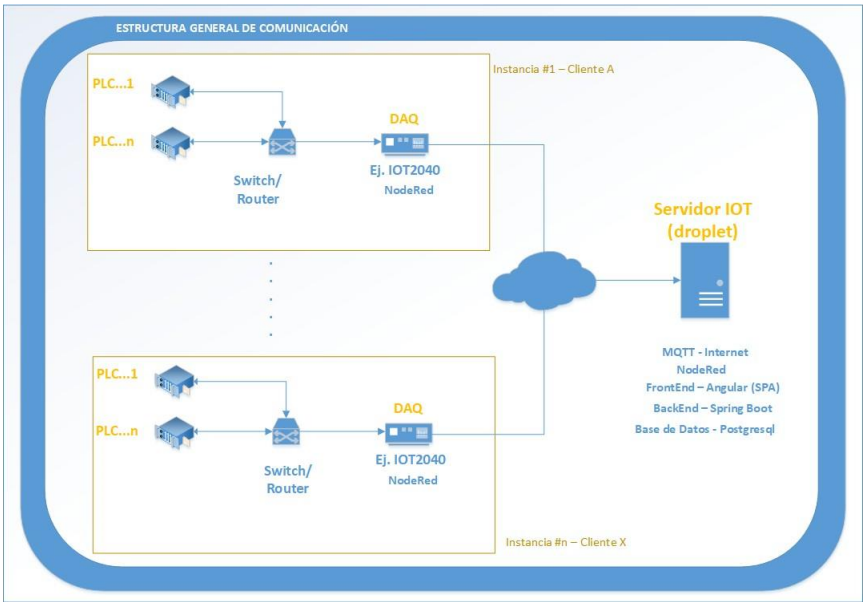


Figura 4.1: Esquema general de Comunicación. Elaboración propia

utilizó en la fase de pruebas es IoT 2040 de Siemens, el cual tiene las siguientes características.

- Tipo de procesador: Intel Quark

- Memoria RAM: 1 Gb.

- Sistema Operativo: Yocto Linux.

- Puertos de comunicación: Dos tarjetas Ethernet y 2 conectores RS232/485.

- Tipo de Programación: Puede ser de tres tipos, en la primera se utiliza una licencia de libre distribución de un IDE (Entorno de Desarrollo de Programación) de Eclipse, el cual permite programar el código en ambiente Java; el segundo tipo de programación hace uso de la herramienta "Node Red", el cual utiliza una interfaz web para programar mediante bloques funcionales las tareas de lectura, procesamiento y publicación de datos en la nube; finalmente, la tercera opción de programación utiliza un aplicativo llamado Mindsphere, el cual mediante el uso de una red privada de Siemens, identifica de manera única los elementos de lectura haciendo uso de una interfaz web también propia de Siemens.

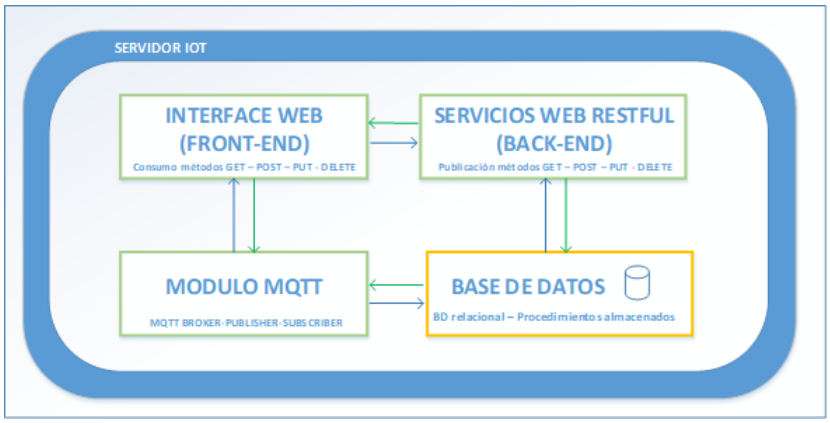


Figura 4.2: Módulos del Servidor IoT. Elaboración propia

- Captura de información: Se seleccionó el modo de programación con Node-Red para la publicación de mensajes MQTT hacia el servidor IoT en la nube.

- Los eventos que se utilizan corresponden a la lectura de datos de los sensores por intervalos de tiempo definidos. Para el caso de las pruebas se realizaron lecturas cada segundo.

- Servidor IoT.- Es el encargado de realizar la lectura de datos de dispositivos y presentar al usuario una interfaz web para una fácil manipulación e interpretación. Será el encargado de permitir el manejo de dispositivos y variables, así como la generación de reportes.

4.2.1. Diseño

El servidor IoT está compuesto de cuatro módulos fundamentales: Front-End, Back-End, MQTT, Base de Datos, como lo muestra la figura4.2.

4.2.2. Módulo Back-End

Este módulo realiza la publicación de servicios web RESTFUL, los cuales se encargan de implementar métodos que interactúan con la base de datos para implementar la capa de negocio del sistema, es decir, la manipulación de información con el respectivo almacenamiento en la base de datos. Para implementar este módulo se ha utilizado la herramienta Spring Boot, con

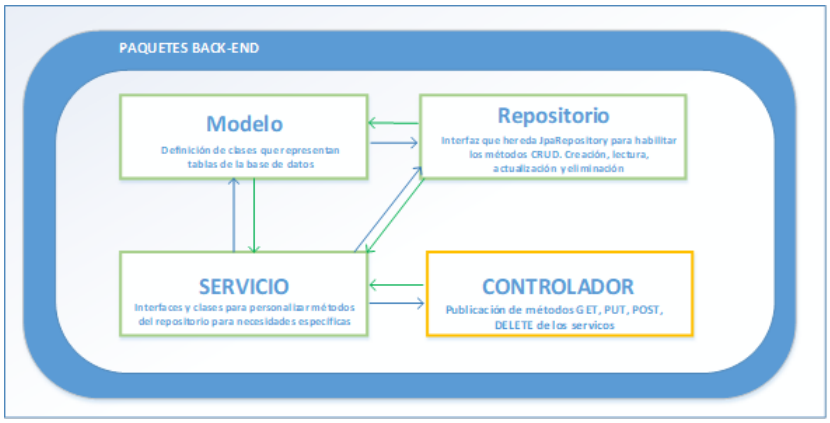


Figura 4.3: Paquetes del módulo Back-End. Elaboración propia

la cual se expone las funciones necesarias mediante la implementación de métodos web tipo GET, POST, PUT y DELETE. La figura 4.3 muestra un esquema general de paquetes que interactúan entre ellos para permitir la publicación y consumo de los diferentes métodos utilizados por otros módulos.

- Paquete Modelo.- Son clases que mapean las tablas de la base de datos, y luego mediante una funcionalidad JPA (Java Persistance API) realiza la interacción con la base de datos.
- Paquete Repositorio.- Son interfaces que implementan JPA para habilitar el modelo CRUD (Create, Read, Update, Delete), para interactuar con la base de datos.
- Paquete Servicio.- Aquí se implementan interfaces de Java, que extienden la funcionalidad del repositorio y agregan funciones específicas que requiera cada clase. Al no presentar acciones particulares, se implementarán los estándares definidos en JPA. Además, este paquete contiene clases de implementación de cada interfaz definida, dando la posibilidad al entorno de Spring Boot realizar la exposición de funciones en el ambiente Spring mediante el uso de la instrucción Autowired.
- Paquete Controlador.- Son clases que implementan las interfaces definidas en el paquete de servicio y lo publican mediante la

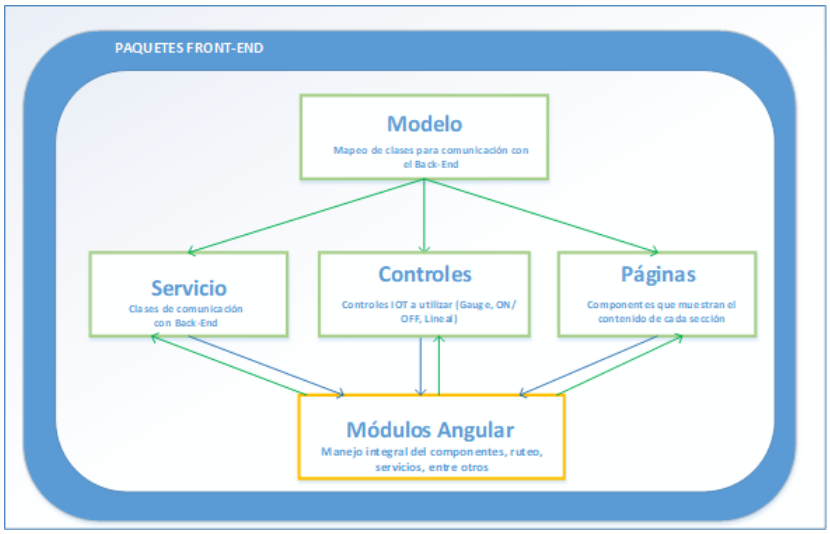


Figura 4.4: Componentes del módulo Front-End. Elaboración propia

implementación de métodos web (GET, POST, PUT, DELETE) disponibles a otros módulos o herramientas.

4.2.3. Front-End

Este módulo muestra la interfaz gráfica al usuario en ambiente web. Se ha utilizado la herramienta Angular que por medio de componentes facilita la visualización de eventos ocurridos durante la lectura y modificación de variables de elementos monitoreados.

Se ha propuesto una tecnología SPA (Single Page Application) que como su nombre lo indica, utiliza una sola página y varia su contenido interactuando entre los distintos componentes.

La figura 4.4 muestra el esquema de componentes utilizado.

- Modelo.- Contiene las clases en typescript que mapean las clases para establecer la comunicación con el Módulo Back-End, es decir instancian objetos que contendrán información proporcionada por el Back-End la cual es obtenida desde la base de datos.

- Servicio.- Funcionan como conectores que consumen los métodos proporcionados por el módulo Back-End. Estos contienen la lógica para llamar a los métodos web GET, POST, PUT, DELETE, los

cuales serán utilizados por componentes encargados de mostrar la información al usuario, específicamente aquellos agrupados dentro del paquete Páginas.

- Controles.- Son componentes que presentan en forma gráfica los distintos tipos de controles utilizados en el ambiente IoT, los cuales mostrarán valores de las variables que se están monitoreando en tiempo real. Se han implementado tres tipos de controles: Gauge, Boolean y Lineales.

- Páginas.- Estos controles simulan la visualización de páginas web, según la sección en la que se encuentre el usuario. Esta es una simulación ya que en realidad al ser una aplicación SPA, solamente se maneja una sola página, pero se va variando el contenido al hacer visibles los componentes del paquete Páginas.

- Módulos Angular.- Son distintos componentes que manejan funcionalidades específicas del Front-End, entre las cuales constan:

  Ruteo: Presentan URL (Unified Routing Language) para cada servicio a utilizar, por ejemplo, existen servicios para listar tableros de control mediante el método GET, cuyo URL podría ser: http://localhost:8080/tableros

  Registro de componentes: Existe una sección llamada app.module.ts en el que se registran los distintos componentes que se van a utilizar en la aplicación.

  Manejo de menús: Componentes que muestran la implementación del menú lateral izquierdo en la interfaz.

A continuación se muestran las secciones que comprenden la interfaz gráfica propuesta:

Sección Principal

Para acceder a la aplicación web será necesario ingresar las credenciales de acceso las cuales constan de un nombre de usuario y contraseña, así como se muestra en la figura 4.5.

La figura 4.6 muestra la sección principal del entorno web, la cual está compuesta por un menú principal de navegación en la parte izquierda, una barra superior donde permite cambiar el comportamiento del menú de navegación, y la parte derecha que muestra el contenido actual de la página,

Figura 4.5: Acceso al sistema. Elaboración propia

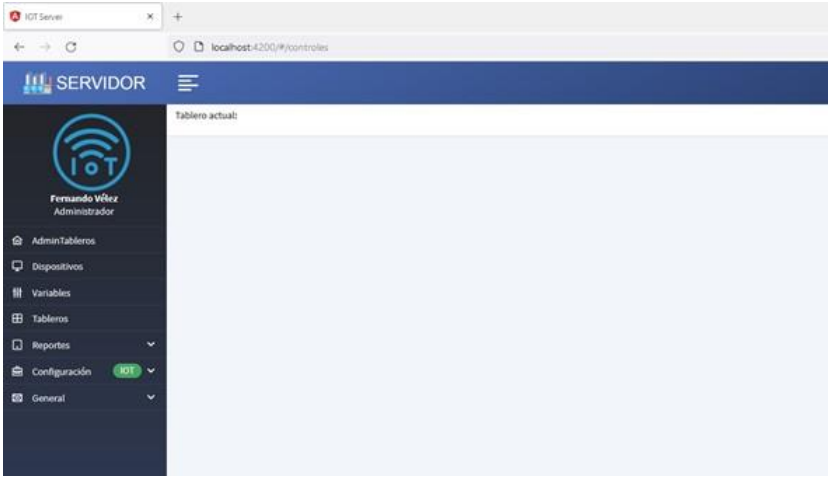


Figura 4.6: Sección principal. Elaboración propia

esta cambiará según la utilización del menú de navegación por parte del usuario.

Sección AdminTableros

Esta sección permite la creación de nuevos tableros de control, así como la selección de los existentes. Será necesario seleccionar un tablero para obtener la información correspondiente en otras secciones. Al seleccionar se mostrará una identificación del tablero en la segunda barra superior bajo el título de "Tablero actual". La figura 4.7 muestra esta sección.

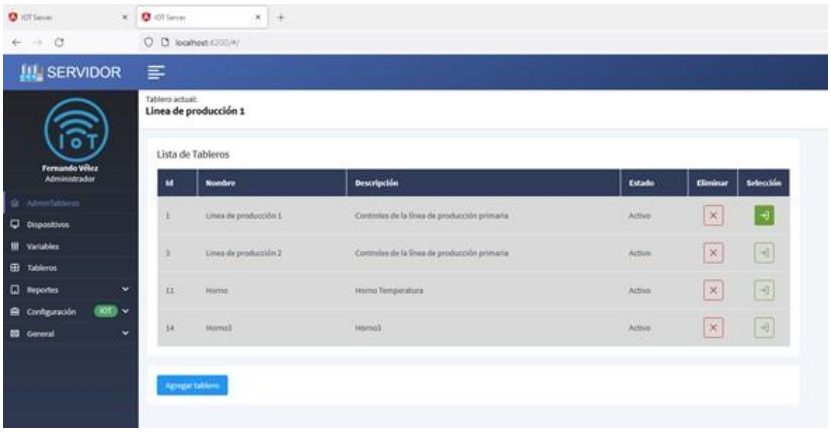


Figura 4.7: Sección AdminTableros. Elaboración propia

Figura 4.8: Sección Dispositivos. Elaboración propia

Sección Dispositivos

Los dispositivos agrupan varios elementos terminales sujetos a supervisión, y se comportará como elemento "Publicador" en el esquema de comunicación MQTT. Esta sección permite la creación de dispositivos en los que se debe indicar información relevante como la dirección IP, puerto y el tópico que utiliza para interactuar con el Bróker MQTT que se encuentra alojado en el servidor. La función de los dispositivos es publicar la información correspondiente a dispositivos finales utilizando la funcionalidad "Publish" del protocolo MQTT. La figura 4.8 muestra la sección dispositivos.

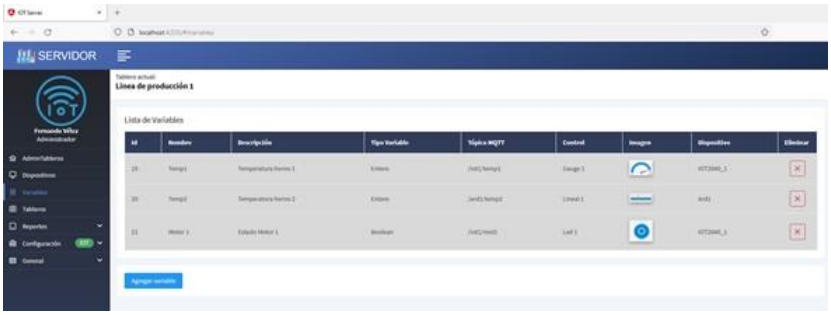


Figura 4.9: Sección Variables. Elaboración propia

Sección Variables

Los dispositivos registrados en la sección anterior contienen variables que representan valores obtenidos o enviados hacia dispositivos terminales (PLC, sensores, etc) que se encuentran manejados por la lógica del dispositivo. Las variables pueden manejar datos de distinto tipo, entre los cuales constan:

- Entero.
- Punto flotante.
- Booleano.

Al seleccionar el tipo de variable, el servidor utilizará el tipo correspondiente en la base de datos para almacenar y consultar información. En la figura 4.9 se puede apreciar la interfaz propuesta para el manejo de variables. La figura 4.10 indica los datos necesarios para crear una nueva variable.

Sección Tableros

Esta sección se encarga de la visualización de los tableros de control. Haciendo referencia a la estructura jerárquica de los dispositivos mostrados en el numeral 4.5, los tableros de control muestran todas las variables contenidas en dispositivos que son asociados al tablero seleccionado.

Para que esta sección muestre la información requerida, es necesario seleccionar el tablero correspondiente en la sección “AdminTableros” descrita en el numeral 4.6.2.2. Los valores de las variables mostradas en el tablero

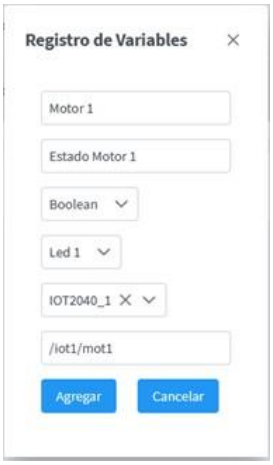


Figura 4.10: Creación de variables. Elaboración propia

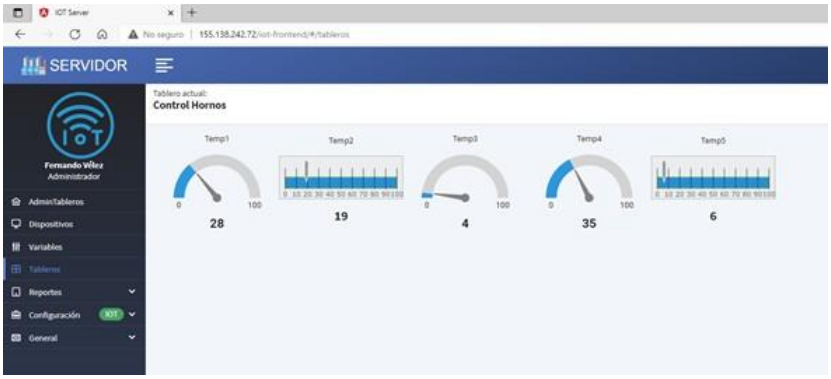


Figura 4.11: Sección Tableros de control. Elaboración propia

son actualizadas en tiempo real, según se reciba un dato enviado por el dispositivo publicador.

La lectura que se realiza a nivel de interfaz de usuario es solamente para fines visuales del estado actual de las variables, más no para almacenar la información en la base de datos, ya que esta tarea la realiza el módulo MQTT descrita en el numeral 4.6.3.

La figura 4.11 muestra un ejemplo de un tablero definido.

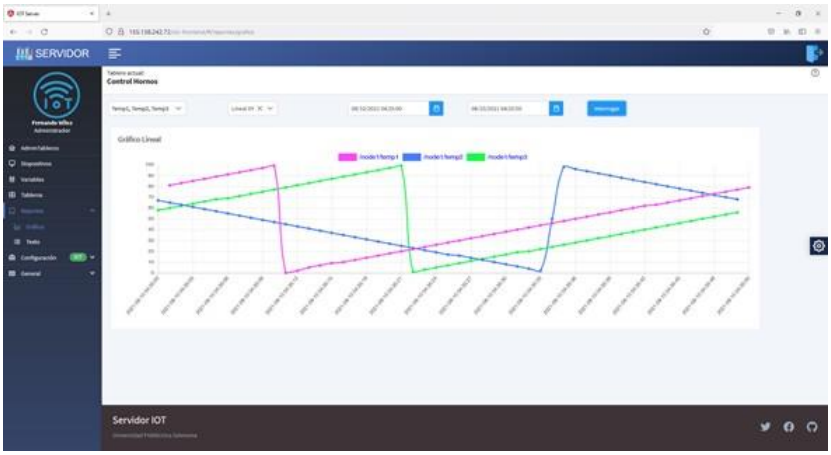


Figura 4.12: Sección Reporte Gráfico. Elaboración propia

Sección Reportes

Los reportes pueden ser visualizados en forma gráfica o listado de texto. Estos permiten la selección de un rango de fechas a ser interrogado en función de la o las variables seleccionadas.

En el caso de los reportes gráficos, es posible seleccionar el tipo de gráfico a desplegar, pudiendo ser LinealXY, Barras o Radar. El reporte de texto permite la misma interrogación, pero despliega la información en una grilla de texto, pudiendo la misma ser descargada en un formato CSV.

La figura 4.12 muestra un ejemplo de reporte gráfico, mientras que la figura 4.13 muestra un ejemplo de reporte de lista de texto.

Sección Configuración

Esta sección permite realizar configuraciones correspondientes al Servidor y al Protocolo MQTT.

Con respecto al servidor se puede especificar los datos de conexión de red como son dirección ip y nombre del host, lo cual será utilizado por los dispositivos publicadores de información.

La configuración a nivel de Protocolo MQTT se refiere a datos de conexión de red del bróker, que por lo general funcionará en el mismo servidor web, pero por diversas razones de capacidad o rendimiento se podría configurar en un servidor distinto.

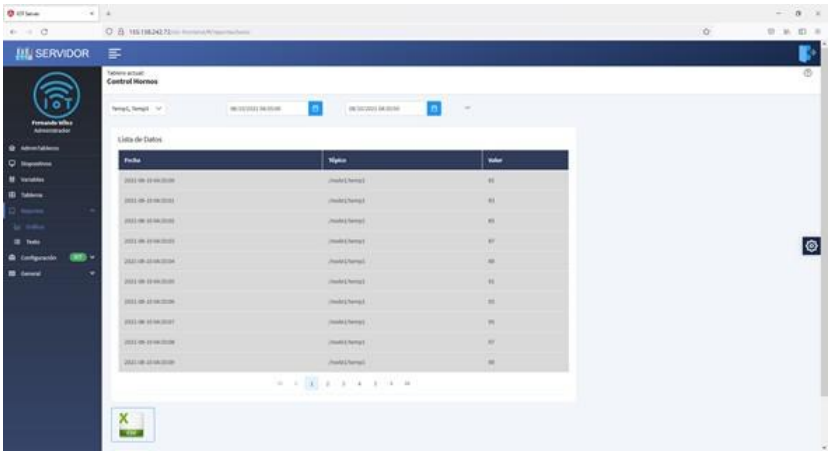


Figura 4.13: Sección Reporte de Texto. Elaboración propia

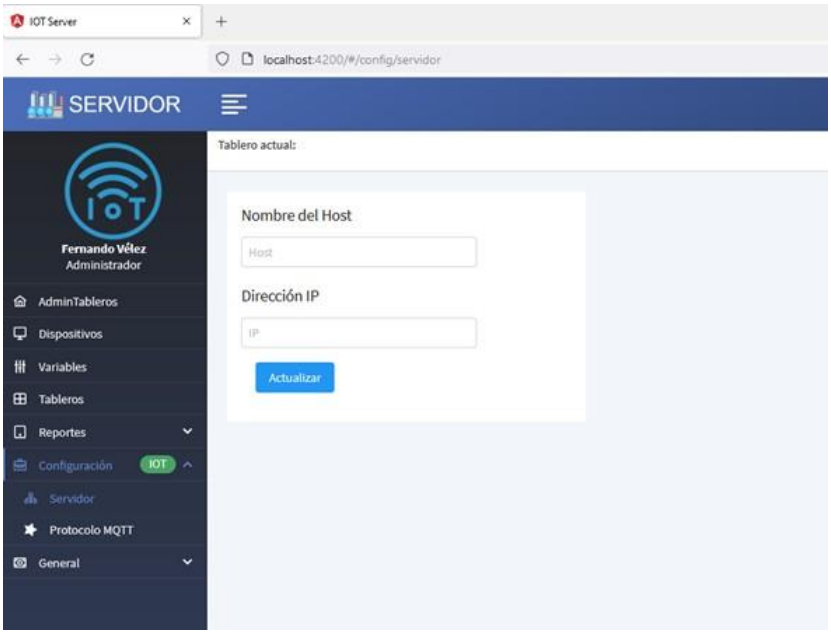


Figura 4.14: Configuración del Servidor. Elaboración propia

La figura 4.14 muestra la interfaz de configuración del servidor; la figura 4.15 se refiere a configuración del protocolo.

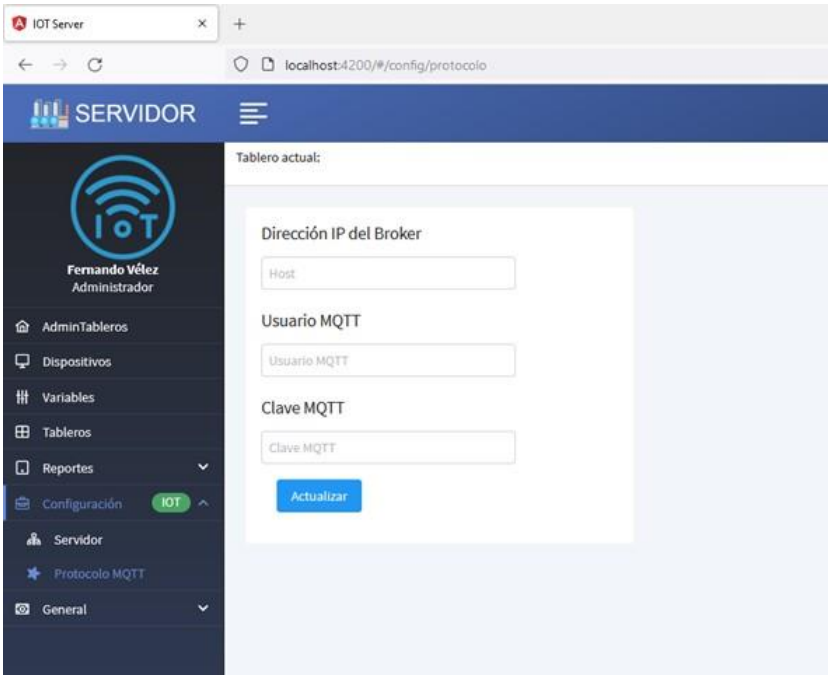


Figura 4.15: Configuración del Protocolo MQTT. Elaboración propia

Sección General

Esta sección muestra información general del desarrollo del presente proyecto.

4.2.4. Módulo MQTT

Para establecer la comunicación con todos los dispositivos y obtener lecturas de las variables, es necesario contar con un módulo independiente, el cual esté activo en forma continua y no únicamente cuando el usuario acceda al sistema. Este módulo establece la función de subscriptor de todos los mensajes MQTT, con la finalidad de obtener los valores y guardarlos en la base de datos.

Al ser un módulo autónomo, no depende de otros para funcionar, solamente realizará subscripciones MQTT por cada variable creada en el módulo Front-End; del mismo modo, realizará cancelaciones de subscripción para las variables que han sido eliminadas.

La figura 4.16 muestra un esquema general de la estructura de este

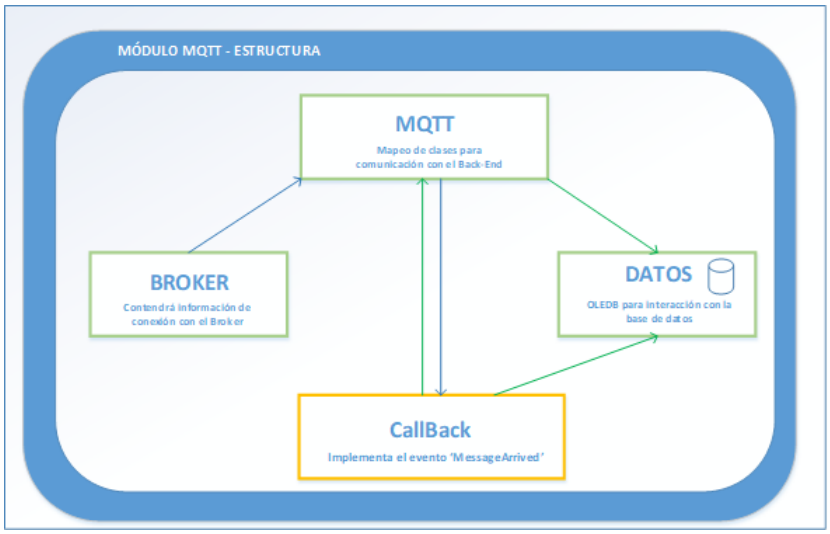


Figura 4.16: Estructura del módulo MQTT. Elaboración propia

módulo.

- La clase Bróker contendrá información de conexión con el bróker. Una instancia de esta clase será utilizada por el módulo MQTT para establecer la comunicación con el servidor donde está alojado el Bróker.
- La clase Datos, implementa un complemento OLEDB para establecer una conexión continua con la base de datos, de tal modo que cuando se reciba una lectura de cualquier variable sea capaz de almacenar la información en la tabla correspondiente. Además, este módulo permite la interrogación de información referente al bróker y variables definidas por medio de atributos específicos.
- La clase MQTT se encarga de implementar el protocolo como tal realizando la conexión con el bróker, además interroga la base de datos por medio del uso de la clase "Datos" para obtener las variables de supervisar.
- Clase CallBack.- Implementa el evento "MessageArrived" que se ejecutará cada vez que un mensaje proveniente de un publicador llega al bróker. Una vez recibido el mensaje lo almacenará en la base de datos colocando la identificación de la variable además de la fecha y hora de la ocurrencia.

4.2.5. Base de datos

Como último módulo del sistema está el motor de base de datos utilizada. Esta es una de tipo relacional con soporte de procedimientos almacenados. Se ha utilizado PostgreSQL versión 11.0, por ser libre de código abierto, además de su confiabilidad y capacidad de almacenamiento.

4.2.6. Implementación como servicio en la nube

El software que realiza las tareas de Servidor IoT se ha implementado en un servidor Droplet en la nube con las siguientes características.

- Tipo de servidor: Virtual en host Vultr.
- Procesador. 1 Core
- Memoria RAM: 1 Gb. NVMe.
- Sistema operativo: Debian 10, 64 bit.
- Dirección IP Pública: 155.138.242.72.
- URL de acceso: http://155.138.242.72/iot-frontend/

4.3. Evidencia de flujo de datos desde el origen hasta la presentación final

En esta sección se evidencia el proceso desde la generación inicial de datos, publicación mediante el protocolo MQTT, adquisición del lado del Servidor IoT y presentación de data en el Front-End del sistema.

4.3.1. Origen de datos

El origen de datos puede ser cualquier dispositivo capaz de publicar mensajes en brokers MQTT. Para este trabajo, durante la fase de pruebas, se obtuvieron datos desde dos fuentes. La primera mediante la programación de un PLC de Siemens S7-1200 modelo 1214C DC/DC/DC FW 4.1, el cual genera datos de control de sensores de nivel de líquidos; y una segunda fuente en la cual se generan datos numéricos ascendentes y descendentes mediante el uso de la herramienta Node-Red. La figura 4.17, muestra el flujo de generación de datos en Node-Red.

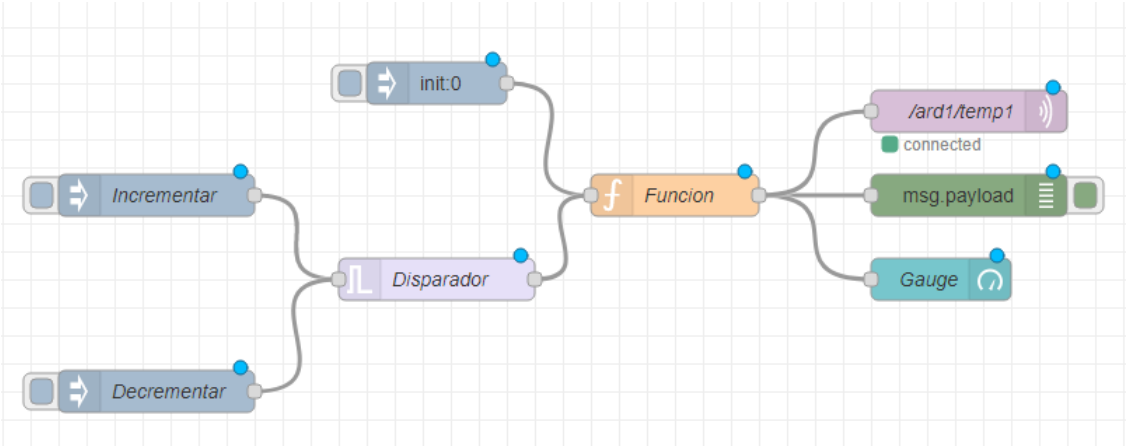


Figura 4.17: Primer origen de dato. Herramienta Node-Red

4.3.2. Adquisición con IoT 2040 y publicación MQTT

Luego que la información está disponible desde los orígenes de datos, se los debe adquirir mediante un dispositivo de adquisición de datos como IoT 2040, o por medio de la utilización de la herramienta Node-Red en forma autónoma. El proceso de adquisición realizará la publicación de los datos adquiridos en forma de mensajes MQTT utilizando el tópico configurado para cada variable. La figura 4.18 muestra el compilado jaMqtt.jar que realiza esta tarea, el cual es llamado desde un programa CRON del droplet en la nube. La figura 4.19 evidencia la información almacenada en la base de datos por parte del proceso CRON.

4.3.3. Subscripción MQTT - Servidor IoT

Del lado del Servidor IoT, se realiza la subscripción de los dispositivos identificados con un tópico, con la finalidad de recibir los datos publicados en la etapa anterior. Se manejan dos tipos de subscriptores que se encuentran detallados en los siguientes ítems.

4.3.4. Recepción y Almacenamiento en base de datos

Se utiliza una primera subscripción para realizar la exclusiva tarea de obtener datos y almacenarlos en la base de datos.

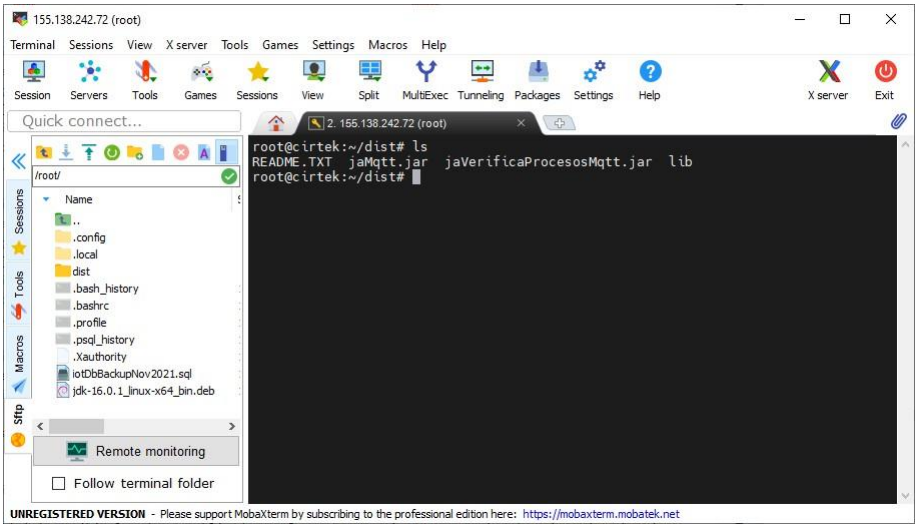


Figura 4.18: Archivo jaMqtt.jar - Adquisición y almacenamiento de datos

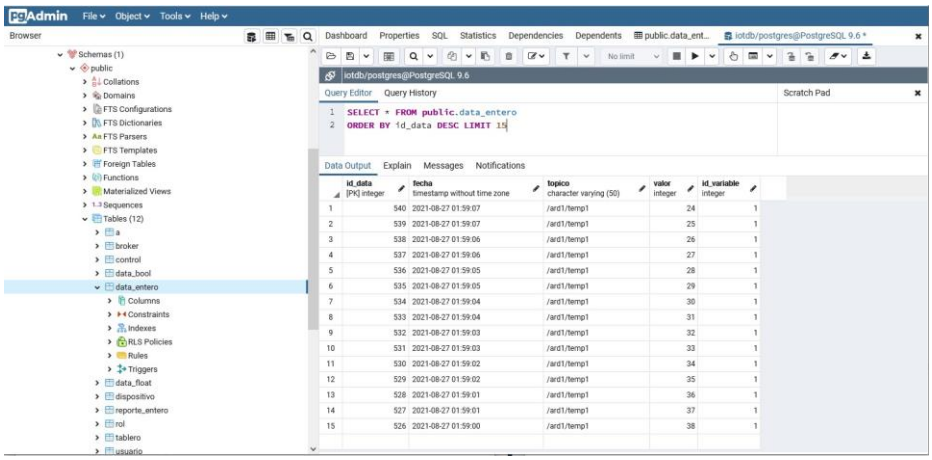


Figura 4.19: Muestra de información almacenados en la base de datos

4.3.5. Visualización gráfica en el panel de control

El segundo subscriptor se utiliza para actualizar en tiempo real los controles asociados a las variables. Estas dos tareas son realizadas con dos subscriptores MQTT diferentes con el objetivo de no afectar el refrescamiento en tiempo real de los controles del panel de control, lográndose así una

actualización gráfica de alta calidad. Ver figura 4.11 donde se evidencia la visualización gráfica del estado actual de los controles en la sección Panel de Control.

4.3.6. Visualización mediante reportes gráficos y texto

Finalmente, mediante el uso de reportes gráficos y de texto se interroga la información adquirida y almacenada en la base de datos. Ver figuras 4.12 y 4.13 donde se evidencia la generación de reportes gráficos y de texto respectivamente.

Capítulo 5

Comparativa entre Plataformas Comerciales y Solución propuesta. Conclusiones y recomendaciones

Si bien las herramientas comerciales presentadas en el capítulo anterior poseen una gran variedad de opciones, interfaces gráficas de usuario muy amigables, así como soporte a muchos protocolos IoT para establecer la comunicación, es también cierto que el costo que se debe pagar por contar con estos servicios es muy elevado.

Para lograr el objetivo principal de este trabajo de titulación, se ha creado una plataforma piloto con funcionalidades menores a las ofrecidas por productos comerciales, sin embargo, es una plataforma que cumple con las tareas básicas que debe poseer un servidor IoT, las mismas que podrán incrementarse según se requiera.

5.1. Análisis de costos de la plataforma propuesta

La solución desarrollada en este proyecto de titulación se ha implementado en un VPS (Virtual Physical Server), el cual es conocido como Droplet. El proveedor que se ha seleccionado para contar con este servicio presenta varias opciones, cuyo costo varía en función del tipo de servidor seleccionado, así como lo muestra la figura 5.1.

Para este proyecto se ha seleccionado la opción más económica, la cual

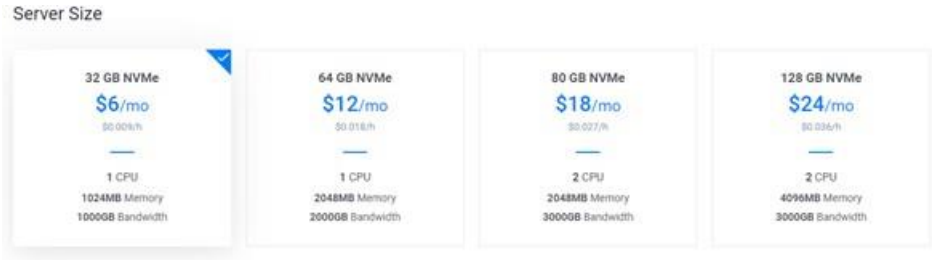


Figura 5.1: Precios de Droplets ofertados por Vultr. Fuente: [Vultr, 2021]

tiene un costo por mes de USD 6,00 y dispone de las siguientes características:

- Memoria RAM: 1GB.
- Espacio en disco duro.- 30 GB. NVMe.
- CPU.- 1 Core.
- Sistema Operativo.- Debian 10 X 64 Buster.

5.2. Hardware utilizado durante las pruebas

La finalidad del servidor IoT es aceptar comunicaciones de dispositivos sin atarse a una tecnología en específico, por tal motivo, se ha utilizado diferentes herramientas de hardware y software que publican lecturas de sensores y simulación de sensores para obtener y almacenar dichas lecturas.

La primera herramienta utilizada para la publicación de datos mediante el protocolo MQTT es el programa Node-Red, que por programación de bloques genera y publica valores de sensores simulados.

Le segunda herramienta se trata del dispositivo IoT 2040 de Siemens, el cual también mediante programación con Node-Red realiza las correspondientes lecturas y publicación de datos.

La figura 5.2, muestra la implementación de este equipo en el laboratorio, en el que se utilizó un PLC Siemens S7-1200 modelo 1214C DC/DC/DC FW 4.1 para ejecutar un programa de simulación de una escala de medición de un tanque que almacena líquido, en la cual se obtiene lecturas del nivel, y por medio de la red LAN dichos datos son adquiridos por el dispositivo IoT 2040, el que a su vez mediante el uso de Node-Red publica el dato adquirido en el servidor IoT objeto de este trabajo de titulación.

Figura 5.2: Implementación de IoT 2040 en laboratorio. Fuente: Elaboración propia

Tabla 5.1: Uso de recursos al declarar variables de dispositivos.
Elaboración propia.

Dispositivos				Memoria RAM				
	us	sy	idl	TOTAL	Libre	Usada	Buffer	Disponible
0	0,00 %	0,00 %	99,00 %	987,20 %	183,10 %	427,10 %	377,10 %	386,80 %
10	0,30 %	0,30 %	99,30 %	987,20 %	180,60 %	429,20 %	377,30 %	384,70 %
50	0,30 %	0,30 %	99,30 %	987,20 %	181,60 %	427,60 %	377,90 %	386,30 %
100	0,00 %	0,30 %	99,70 %	987,20 %	175,20 %	432,40 %	379,60 %	381,50 %
200	0,30 %	0,70 %	98,70 %	987,20 %	169,10 %	436,20 %	381,90 %	377,50 %
500	1,30 %	0,70 %	97,70 %	987,20 %	167,80 %	438,00 %	381,40 %	375,80 %
1000	0,70 %	0,30 %	99,00 %	987,20 %	162,90 %	441,40 %	381,40 %	375,80 %
5000	0,70 %	0,70 %	98,70 %	987,20 %	159,80 %	440,30 %	387,00 %	373,30 %

5.3. Recursos utilizados en función del número de dispositivos supervisados

El análisis de costos de las plataformas comerciales realizado en el capítulo 3, se desarrolló completando una tabla estándar de precios que consideraba un número específico de dispositivos supervisados, entre los cuales se consideró: 10, 50, 100, 200, 500, 1000 y 5000. En esta sección se analiza el uso de recursos al declarar un número similar de dispositivos en supervisión, y también se analiza la utilización de recursos al realizar lecturas y almacenamiento de datos con diferentes números de dispositivos.

La tabla 5.1 muestra los recursos utilizados al declarar dispositivos sin generar datos de prueba, es decir, sin recibir información, únicamente se considera la creación de los controles previos a la recepción de datos.

Como se puede apreciar en la tabla 5.1, el consumo de recursos del

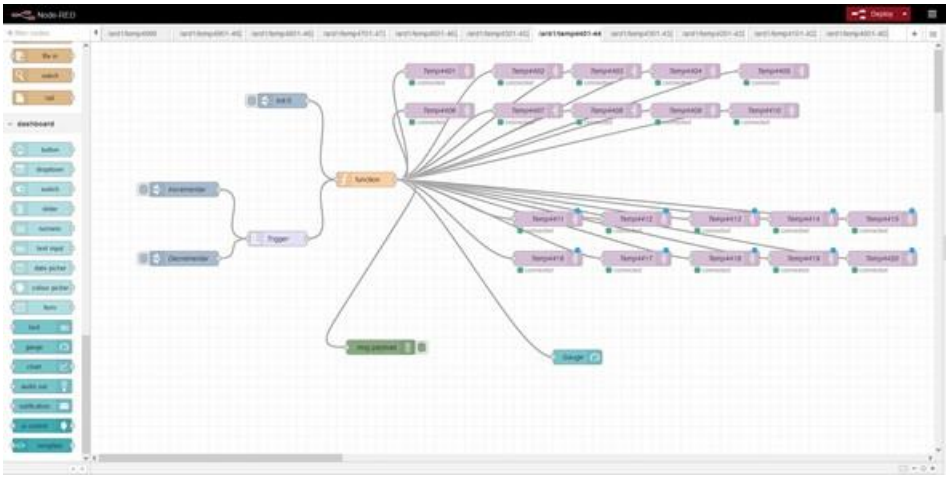


Figura 5.3: Generación de datos en Node-Red. Elaboración propia.

servidor para declarar los diferentes niveles de variables es poco significativa, concretamente hablamos de un uso de memoria RAM de aproximadamente 23.3 Mb, si consideramos la diferencia entre la memoria libre en 0 y 5000 elementos. En cuanto al uso de tiempo de CPU es también insignificante, ya que se consume alrededor de dos puntos porcentuales en mantener las 5000 variables activas.

El siguiente parámetro que se analiza es el consumo de recursos al generar datos desde una fuente externa, con la finalidad de que las variables supervisadas reciban valores, para los cuales el servidor debe desplegar al usuario y almacenar en la base de datos. De hecho estas tareas demandarán mayor procesamiento.

La forma de generar datos se la realizó utilizando la herramienta Node-Red, la cual desde un computador local envía datos al Droplet, quepara este caso se encuentra físicamente en Dallas-Texas, EE.UU.

La figura 5.2 muestra un flujo que emite valores correspondientes a 20 variables, y se ha generado 50 flujos independientes, por lo que se analiza hasta 1000 variables realizando lecturas con un intervalo de una por segundo.

La Tabla 5.2 muestra los recursos utilizados al supervisar variables, que para este ensayo se consideró un número máximo de 1000 dispositivos que simulan sensores en la vida real.

Al analizar los datos de la Tabla 5.2, se determina que los recursos utilizados para leer 1000 variables con un período de un dato por segundo

Tabla 5.2: Uso de recursos al supervisar variables con lectura de datos. Elaboración propia.

Dispositivos				Memoria RAM				
	us	sy	idl	TOTAL	Libre	Usada	Buffer	Disponible
0	0,00 %	0,30 %	99,70 %	987,20 %	95,70 %	397,10 %	494,50 %	301,00 %
20	1,70 %	0,30 %	97,70 %	987,20 %	88,30 %	406,80 %	492,10 %	291,30 %
40	1,70 %	0,70 %	97,00 %	987,20 %	82,60 %	417,20 %	487,40 %	280,90 %
60	2,00 %	1,00 %	96,30 %	987,20 %	77,50 %	418,00 %	491,70 %	280,00 %
80	2,40 %	1,00 %	96,00 %	987,20 %	68,80 %	417,40 %	501,00 %	280,40 %
100	3,30 %	1,30 %	94,00 %	987,20 %	62,90 %	418,40 %	505,90 %	279,40 %
200	6,00 %	2,70 %	89,00 %	987,20 %	65,40 %	417,80 %	504,00 %	279,70 %
400	12,60 %	5,30 %	77,10 %	987,20 %	88,20 %	414,00 %	485,10 %	283,10 %
600	18,00 %	7,10 %	68,70 %	987,20 %	73,50 %	413,80 %	499,90 %	282,90 %
800	20,60 %	11,30 %	59,80 %	987,20 %	77,70 %	415,00 %	494,50 %	281,10 %
1000	27,20 %	12,90 %	50,00 %	987,20 %	65,90 %	414,90 %	506,50 %	280,60 %

Tabla 5.3: Uso de disco duro al almacenar lecturas de variables. Elaboración propia.

Dispositivos	***Num.Lecturas***	***UsodeDisco***	***UtilizadoporelSistema***	***Tiempo***
1000	0,00	3,5 GB	$0 GB	0 horas
1000	1000000,00	3,8 GB	0,3 GB	0,27 horas
1000	26000000,00	5,0 GB	1,5 GB	7,22 horas
1000	35000000,00	5,8 GB	2,3 GB	9,72 horas

por cada variable, consume alrededor de 21Mb de memoria RAM, si se toma en cuenta la columna de memoria disponible. En cuanto al tiempo de uso del procesador, se muestra que se utiliza un 50 por ciento al realizar la supervisión de las 1000 variables.

La Tabla 5.3 muestra el consumo de espacio en disco duro al realizar el almacenamiento de las lecturas de datos correspondiente. Se concluye de este estudio que se requiere 2.3 GB de espacio en disco duro para almacenar valores provenientes de 1000 variables en un período de captura de un segundo por un lapso de 9,72 horas, es decir, se requerirán alrededor de 170 GB para almacenar información en forma continua por un mes.

Con estos resultados se puede simplificar el uso de recursos por cada variable a razón de un dato por cada segundo de la siguiente manera:

- Uso aproximado de Memoria RAM por cada variable: 21Mb/1000 = 0,021 Mb.
- Tiempo de uso del microprocesador por cada variable: 50.0/1000 = 0.05

Tabla 5.4: Costos de la plataforma propuesta en el trabajo de titulación. Elaboración propia.

Dispositivos	*CostoporServidor*	*TipodeDroptlet*	*Seleccin*
10	$6,00	1 CPU, 1GB Ram, 30 GB disco	$6,00
50	$6,00	1 CPU, 1GB Ram, 30 GB disco	$6,00
100	$6,00	1 CPU, 1GB Ram, 30 GB disco	$6,00
200	$6,00	1 CPU, 1GB Ram, 30 GB disco	$6,00
500	$6,00	1 CPU, 1GB Ram, 80 GB disco	$20,00
1000	$6,00	8 CPU, 32GB Ram, 640 GB disco	$160,00
5000	$6,00	8 CPU, 32GB Ram, 1280 GB disco	$280,00

Tabla 5.5: Costo de la plataforma propuesta por número de lecturas. Elaboración propia.

Dispositivos	*Num.Lecturas*	*UsodeDisco*	*UtilizadoporelSistema*	*Tiempo*
1000	0,00	3,5 GB	$0 GB	0 horas
1000	1000000,00	3,8 GB	0,3 GB	0,27 horas
1000	26000000,00	5,0 GB	1,5 GB	7,22 horas
1000	35000000,00	5,8 GB	2,3 GB	9,72 horas

- Espacio en disco: 0,000067 Mb. por cada variable por segundo.

Con estos datos se puede proyectar los recursos utilizados como muestra la Tabla 5.4.

Una vez analizados los recursos necesarios para soportar 5000 variables, se determina los costos asociados, los cuales se muestran en la Tabla 5.5.

5.4. Comparativa de Precios

La Tabla 5.6 muestra la comparativa de precios de plataformas de productos comerciales expuesta en el capítulo 3, frente a los precios obtenidos de la plataforma objeto de este trabajo de titulación.

Al analizar la figura se nota que la solución propuesta es la más económica entre las estudiadas, es de tomar muy en cuenta que en la plataforma propuesta se realiza un análisis de costos considerando una frecuencia de adquisición de datos de una muestra por segundo por variable, mientras que en las comerciales se ha realizado el ejercicio con una muestra cada 30 minutos, ya que resultaría en un costo demasiado elevado realizarlo con la frecuencia propuesta en el presente trabajo.

Tabla 5.6: Comparativa de costos de la plataforma propuesta frente a opciones comerciales.
Elaboración propia.

Dispositivos	*GoogleCloud*	*AWS*	*IBMWatson*	*MicrosoftAzure*	*Kaa*	*Ubidots*	*PROPIA*
10 disp	$225,56	$64,85	$310,00	$2.505,49	$14,99	$49,00	$6,00
50 disp	$1.136,81	$324,26	$350,00	$2.532,94	$44,99	$199,00	$6,00
100 disp	$2.275,88	$648,52	$400,00	$2.567,25	$79,99	$199,00	$6,00
200 disp	$2.024,00	$1.297,04	$500,00	$2.635,87	$175,00	$199,00	$6,00
500 disp	$5.061,50	$3.242,59	$800,00	$2.841,73	$325,00	$499,00	$20,00
1000 disp	$10.124,00	$6.485,18	$1.300,00	$3.184,83	$625,00	$499,00	$160,00
5000 disp	$11.390,40	$32.425,92	$5.300,00	$5.929,63	ND	$2.199,00	$280,00

Si bien existen diferencias en las funcionalidades con respecto a las herramientas comerciales, ya que proveen más opciones de selección de protocolos a utilizar, así como de interfaces más elaboradas que se traducen en la facilidad de uso de las plataformas; este trabajo representa un prototipo que realiza las funciones básicas, y representa una solución que está al alcance de la realidad ecuatoriana, ofreciendo una solución viable especialmente para el sector PYMES que normalmente en nuestro medio tiene restricciones económicas.

5.5. Conclusiones

Un análisis inicial de los protocolos de comunicación utilizados en el ambiente IoT indicó que existen muchos que realizan tareas que se enfocan en problemáticas diferentes, hubiese sido ideal implementar los más utilizados, pero debido a la gran cantidad de variantes y por el enfoque que se dio a este trabajo de titulación, se seleccionó el que mejor se ajustaba para cumplir con los objetivos propuestos. La implementación del protocolo MQTT funcionó de manera adecuada, ya que permitió alcanzar lo planteado, la velocidad de procesamiento de información y la confiabilidad en la transmisión resultó en una muy buena opción seleccionada. Cabe indicar que se debería implementar otros protocolos si las necesidades de viabilizar nuevos tareas son agregadas al producto desarrollado, como por ejemplo, si se requiere implementar tareas asíncronas de distinta índole, seria deseable implementar el protocolo AMQP.

En cuanto a la interfaz de usuario, se seleccionó la funcionalidad SPA, que como se mencionó en el capítulo 4, es una tecnología que utiliza una sola página y sobre ella se trabaja con componentes. Esta visión ayudó en forma muy significativa al refrescamiento de lectura de datos sobre los controles.

Se realizaron muchas pruebas con un número muy elevado de elementos en supervisión, y la visualización de los mismos no se vio afectada a pesar de utilizar cientos de variables supervisadas.

La generación de reportes es muy sencilla, y permite visualizar resultados en forma eficiente. Se presenta dos tipos reportes; un gráfico y uno de texto. El reporte gráfico permite sobre montar lecturas correspondientes a varios elementos en rangos de fechas seleccionadas, mientras que el de texto presenta una funcionalidad similar pero los datos son desplegados en listas. Estos dos tipos de reportes facilitan la toma de decisiones sobre el funcionamiento general del sistema supervisado.

Se ha desarrollado un producto que debería resultar muy interesante para el sector industrial del medio local, ya que representa una opción de muy bajo costo para la publicación de información relevante al estado de sensores y variables de interés, que hasta ahora, en cierta manera ha sido restringida por el alto costo que implica contratar productos comerciales.

Esta plataforma contribuye a la implementación de la Industria 4.0 localmente, lo que significa disminuir la brecha digital con respecto a otros países que están muy adelantados en este ámbito por el poder económico que disponen.

5.6. Recomendaciones

El prototipo desarrollado en este trabajo de titulación puede ser complementado con nuevas funcionalidades para soportar diferentes enfoques de comunicación IoT, por lo que se recomienda considerar nuevos trabajos de titulación sobre esta plataforma.

Al tratarse de un tema actual y que a nivel local no se ha implementado en forma significativa, se recomienda realizar convenios con empresas del sector público o privado con la intención de introducir técnicas de publicación de datos mediante la utilización de plataformas IoT.

Finalmente, en la universidad se podría proponer asignaturas que se orienten a implementar y/o mejorar los protocolos de comunicación IoT y plataformas existentes para impulsar al medio local a esta nueva tecnología.

Bibliografía

E. Al-Masri, K. R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo, and C. Yan. Investigating messaging protocols for the internet of things (iot). *IEEE Access*, 8:94880–94911, 2020.

I. Amazon Web Services. Aws iot platform pricing. https://aws.amazon.com/es/iot-core/pricing/, 2021. Accessed August 15, 2021.

G. Cloud. Google cloud iot platform pricing. https://cloud.google.com/pricing, 2021. Accessed August 15, 2021.

A. Corsaro. The data distribution service tutorial. *Technical Report 4.0*, 2014, 2014.

M. H. Elgazzar. Perspectives on m2m protocols. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 501–505. IEEE, 2015.

H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel. Survey of platforms for massive iot. 2018 ieee international conference on future iot technologies. *Future IoT*, 2018, 2018.

C. Helen. Watson iot platform new price. https://www.ibm.com/cloud/blog/better-pricing-better-standard-plan-watson-iot-platform, 2021. Accessed August 10, 2021.

H. Jin, S. Ibrahim, T. Bell, W. Gao, D. Huang, and S. Wu. Cloud types and services. In *Handbook of Cloud Computing*, pages 335–355. Springer, 2010.

L. KaaIoT Technologies. kaa iot platform pricing. https://www.kaaiot.com/pricing?currentPlan=5, 2021. Accessed August 15, 2021.

F. F. Moghaddam, M. B. Rohani, M. Ahmadi, T. Khodadadi, and K. Madadipouya. Cloud computing: Vision, architecture and

characteristics. In *2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, pages 1–6. IEEE, 2015.

S. Munirathinam. Industry 4.0: Industrial internet of things (iiot). In *Advances in computers*, volume 117, pages 129–164. Elsevier, 2020.

D. Priyadarshi and A. Behura. Analysis of different iot protocols for heterogeneous devices and cloud platform. In *2018 International Conference on Communication and Signal Processing (ICCSP)*, pages 0868–0872. IEEE, 2018.

D. Rani and R. K. Ranjan. A comparative study of saas, paas and iaas in cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(6), 2014.

H. RiLi. Research and application of tcp/ip protocol in embedded system. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 584–587. IEEE, 2011.

A. Rojko. Industry 4.0 concept: Background and overview. *International Journal of Interactive Mobile Technologies*, 11(5), 2017.

M. Sala-Zárate and L. Colombo-Mendoza. Cloud computing: a review of paas, iaas, saas services and providers. *Lámpsakos*, (7):47–57, 2012.

Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). 2014.

P. Thota and Y. Kim. Implementation and comparison of m2m protocols for internet of things. In *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD)*, pages 43–48. IEEE, 2016.

Ubidots. Ubidots iot platform pricing. https://ubidots.com/pricing/, 2021. Accessed August 15, 2021.

S. Vinoski. Advanced message queuing protocol. *IEEE Internet Computing*, 10(6):87–89, 2006.

Vultr. Vultr vps pricing. https://my.vultr.com/deploy/, 2021. Accessed August 25, 2021.

J. Yang, K. Sandström, T. Nolte, and M. Behnam. Data distribution service for industrial automation. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pages 1–8. IEEE, 2012.

M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi. Internet of things: Survey and open issues of mqtt protocol. In *2017 international conference on engineering & MIS (ICEMIS)*, pages 1–6. Ieee, 2017.