UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:
DESARROLLO DE UNA ARQUITECTURA CLOUD COMPUTING PARA COMUNICAR DOS ESTACIONES DEL SISTEMA DE PRODUCCIÓN MODULAR UTILIZANDO EL PROTOCOLO MQTT

AUTORES:
EDISON FABRICIO FERNÁNDEZ MARTÍNEZ
JHORDAN ANDRÉS ORDÓÑEZ NARVÁEZ

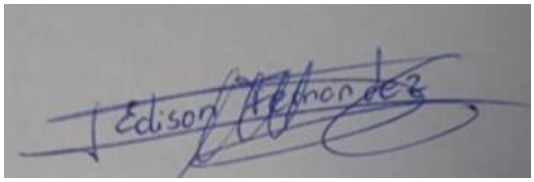
TUTOR:
GUSTAVO JAVIER CAIZA GUANOCHANGA

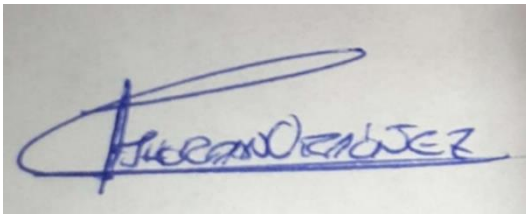
Quito, marzo del 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Edison Fabricio Fernández Martínez y Jhordan Andrés Ordóñez Narváez con documentos de identificación N° 0503458374, N° 0401555990 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de grado intitulado: DESARROLLO DE UNA ARQUITECTURA CLOUD COMPUTING PARA COMUNICAR DOS ESTACIONES DEL SISTEMA DE PRODUCCIÓN MODULAR UTILIZANDO EL PROTOCOLO MQTT, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento de la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.





Edison Fabricio Fernández Martínez

C.I. 0503458374

Jhordan Andrés Ordoñez Narváez

C.I. 0401555990

Quito, marzo del 2021

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, DESARROLLO DE UNA ARQUITECTURA CLOUD COMPUTING PARA COMUNICAR DOS ESTACIONES DEL SISTEMA DE PRODUCCIÓN MODULAR UTILIZANDO EL PROTOCOLO MQTT realizado por Edison Fabricio Fernández Martínez y Jhordan Andrés Ordóñez Narváez, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, marzo de 2021


.

Gustavo Javier Caiza Guanochanga

Cédula de identidad: 1721192191

DEDICATORIA

El presente artículo va dedicado a mi madre Melida Martínez por nunca haberme abandonado en este proceso de formación académica.

Edison Fabricio

El presente trabajo investigativo lo dedico principalmente a Dios, por ser el inspirador y darme la fuerza para continuar en este proceso de obtener uno de los anhelos más deseados.
A mis padres Ramiro Ordóñez y Mery Narváez por su amor, trabajo y sacrificio en todos estos años, gracias a ustedes he logrado llegar hasta aquí y convertirme en lo que soy.
A mi hermana por estar siempre presente, acompañándome y por el apoyo moral, que me ha brindado a lo largo de esta etapa.
Finalmente quiero dedicar este artículo a mi esposa e hija por apoyarme cuando más las necesito, por extender su mano en momentos difíciles y por el amor brindado cada día.

Jhordan Andrés

AGRADECIMIENTO

Nuestros más sinceros agradecimientos a todas esas personas que nos ayudaron moral y económicamente para culminar nuestros estudios.

A la Universidad Politécnica Salesiana, nuestro tutor Ing. Gustavo Caiza y todos aquellos profesores que nos compartieron sus conocimientos.

Desarrollo de una arquitectura Cloud computing para comunicar dos estaciones del Sistema de Producción Modular utilizando el protocolo MQTT

Edison Fernández
Universidad Politécnica Salesiana
Ingeniería Electrónica
Quito, Ecuador
efernandezm1@est.ups.edu.ec

Jhordan Ordóñez
Universidad Politécnica Salesiana
Ingeniería Electrónica
Quito, Ecuador
jordanzn@est.ups.edu.ec

Gustavo Caiza
Docente Universidad Politécnica Salesiana
Ingeniería Electrónica
Quito, Ecuador
gcaiza@ups.edu.ec

***Resumen*— En este artículo se aborda la implementación de una red inalámbrica en el laboratorio de MPS 500 (Sistema de Producción Modular), otorgando la capacidad de recibir y registrar datos a través del protocolo MQTT (Message Queue Telemetry Transport), para minimizar el uso de cableado y un enfoque a IoT (Internet de las cosas). El propósito es sustituir los PLC´s (autómatas programables) de las estaciones de distribución, clasificación y conveyor incluido su medio de comunicación guiado por un medio inalámbrico que incluya al protocolo MQTT para la transmisión de mensajes y que por medio de publicación y suscripción las piezas (fichas) involucradas sean procesadas desde la estación de distribución y almacenadas en la estación de clasificación, además de incluir una base de datos que permite la visualización de los registros por cualquier usuario conectado a la red, interconectando dispositivos como tarjetas controladores ESP32 y Raspberry´s empleadas como dispositivo maestro y bróker respectivamente, consiguiendo una arquitectura que permita la comunicación entre los módulos mencionados del MPS 500 con el propósito de demostrar que el protocolo MQTT puede transmitir datos a velocidades comparables a los estándares de comunicación, como el de PROFIBUS, donde además la eficacia sea del 100% y no existan perdidas en los temas publicados y sean recibidos y registrados en la base de datos.**

Palabras clave—MQTT, Raspberry-pi, IoT, ESP32

***Abstract*— This article addresses the implementation of a wireless network in the MPS 500 (Modular Production System) laboratory, granting the ability to receive and record data through the MQTT (Message Queue Telemetry Transport) protocol, to minimize the use of cabling. and a focus on IoT (Internet of Things). The purpose is to replace the PLC's (programmable automatons) of the distribution, classification and conveyor stations, including its means of communication guided by a wireless medium that includes the MQTT protocol for the transmission of messages and that by means of publication and subscription pieces (tokens) involved are processed from the distribution station and stored in the sorting station, in addition to including a database that allows the viewing of the records by any user connected to the network, interconnecting devices such as ESP32 and Raspberry controller cards It is used as master device and broker respectively, achieving an architecture that allows communication between the mentioned modules of the MPS 500 in order to demonstrate that the MQTT protocol can transmit data at speeds comparable to communication standards, such as PROFIBUS, where in addition the efficiency is 100% and there are no losses gone in the published topics and are received and registered in the database.**

Keywords—MQTT, Raspberry-pi, IoT, ESP32

I. INTRODUCCIÓN

Actualmente, IoT (Internet de las Cosas) se ha convertido en una plataforma que se utiliza en diversas áreas, como automatización industrial, domótica o con fines médicos, conectando dispositivos y monitoreando esos datos desde cualquier lugar.[1]

La cantidad de dispositivos conectados a Internet está creciendo y se espera que continúe creciendo exponencialmente en todo el mundo a medida que artefactos se compren alrededor de este. Con este crecimiento están surgiendo millones de nuevos tipos de dispositivos a su vez protocolos que permiten que las máquinas se conecten entre sí. Estos aparatos se comunican y ofrecen servicios a través de Internet.[2]

En la Unión Europea y Corea del Sur emitieron políticas para el desarrollo de IoT, y el gobierno chino incluyo en su plan nacional para el desarrollo científico y tecnológico, a medio y largo plazo, con el propósito de acelerar las industrias con internet de las cosas.[3]

Con el desarrollo de MQTT se generan aplicaciones para el intercambio de información entre los dispositivos como el caso de mensajería instantánea por parte de Facebook[4][5], donde los clientes reciben solo la información solicitada en cada conversación en cuestión de milisegundos o notificaciones relacionadas a su interés, como el servicio de AWS IoT en el caso de Amazon[6]

La implementación de MQTT es ideal para diversos campos como prototipos de SmartHome donde los datos de los sensores pasan a través del gateway para alojarse en la nube y publicarse en una plataforma IoT.[7], incluso puede abarcar procesos industriales más complejos como variación de velocidad de motores o control de sistemas de presión, donde los sensores recopilan información para publicarlos en las herramientas IoT con la finalidad de realizar procesos de control más sofisticados y sean administrados por cualquier dispositivo conectado a internet.[8]

En el laboratorio de MPS existen las estaciones de distribución y clasificación, separadas físicamente entre sí, que fueron seleccionadas porque simulan procesos de automatización reales de ordenamiento de piezas, además de una tercera estación de conveyor o banda de transporte que es el enlace entre las dos anteriores y tomada como estación maestra, estas están regidas por PLC´s y cuya comunicación es por medio de un bus de comunicación de campo (Profibus). Es así que este trabajo pretende implementar una red inalámbrica que reemplace la estructura cableada a través del protocolo MQTT, porque permite una arquitectura publicador-suscriptor bidireccional de forma sencilla y ligera para publicar datos [3][9] tanto así, que este tipo de protocolo se ha convertido en un pilar fundamental de IoT. Debido a que estos sistemas carecen de módulos inalámbricos y con la finalidad de sustituir los PLC´s de las

estaciones de clasificación, distribución y conveyor, pues no cuentan con la capacidad de soportar este nuevo protocolo para cumplir con el cometido planteado en este estudio se emplean dos tarjetas MCU (micro-controladores) ESP32 con módulos WI-FI y dos dispositivos MPU (microprocesadores) Raspberry Pi 3 Y Pi 4 como maestro y bróker respectivamente, además que permiten al usuario (operador) visualizar y controlar las variables de proceso del MPS 500 culminando con un registro en una base de datos incluida en la Raspberry Pi 3.

Las secciones de este trabajo están organizadas de la siguiente manera: en la sección II se describen los conceptos básicos del protocolo MQTT y la metodología utilizada; en la sección III se muestra el diseño del protocolo para las estaciones de MQTT, en la sección IV se expresan los resultados y, finalmente, en la sección V se presentan las conclusiones.

II. MATERIALES Y METODOS

Las estaciones de clasificación, distribución y conveyor al estar constituidas con sensores y actuadores que permiten cumplir con un proceso de ordenamiento, distribución y transporte de fichas acorde a un color, sea este rojo, negro o plateado como se especifica en el diagrama de flujo de la figura 1.

El proceso normal que realiza el MPS 500 se describe así, el color de la ficha es ingresado por el usuario y puede ser seleccionado aleatoriamente, la estación de distribución evalúa mediante sensores si el color es correcto, en caso de no serlo será desechada, si es verdadero entonces lo envía hacia el conveyor, al final del recorrido la estación de clasificación los direcciona y ordena en tres diferentes filas.

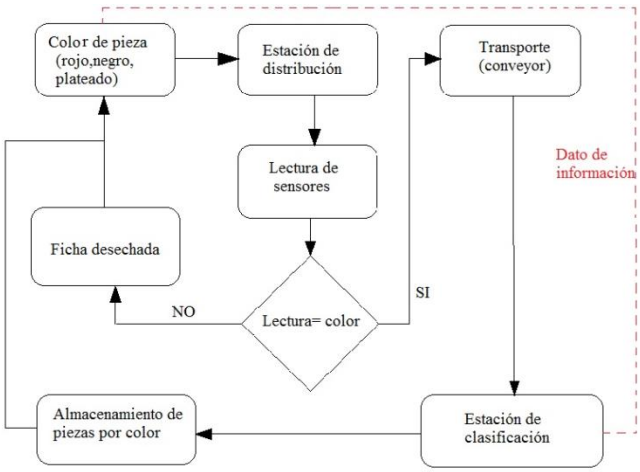


Figura 1. Proceso de MPS 500.

El funcionamiento del protocolo MQTT en este mismo proceso requiere de los siguientes parámetros.

SOFTWARE

A. Librería MQTT

La librería de MQTT incluye los algoritmos que facilitan la programación para la publicación de mensajes desde los códigos escritos tanto para Raspberry y las tarjetas electrónicas ESP32 pues los temas publicados contienen y necesitan de comunicación bidireccional de punto a multiples puntos, además que la Liberia MQTT está disponibles para los lenguajes de programación de IDE Arduino y python, [10] con los cuales se desarrolla este trabajo.

B. Servidor de mensajes (Mosquitto)

MQTT como protocolo puede ejecutarse dentro de varios sistemas operativos, y por eso existen una variedad de brókers disponibles, Mosquitto es un servidor de mensajes liviano y adecuado para uso en dispositivos de baja potencia [11] como la Raspberry Pi 3 y Pi 4 y es idóneo para ejecutar la aplicación de IoT dentro del MPS 500 pues la publicación de mensajes que se requiere no necesitan de un servidor más robusto, además existen guías dentro de los repositorios digitales para su uso dentro de Python.

El servidor Mosquitto cumple ampliamente con los estándares de MQTT y las implementaciones para usarlos con las tarjetas controladoras ESP32 y Raspberry´s cuenta con tres partes esenciales necesarias a incluirse:[12]

- El servidor principal para el bróker Raspberry Pi 4.
- Las utilidades de cliente mosquitto_pub y mosquitto_sub que son un método de comunicación entre Raspberry´s y tarjeras controladoras ESP32.
- Una biblioteca de cliente MQTT escrita en C, con un contenedor C ++ y compatible con Python.

Desde la terminal de Raspberry, Mosquitto debe ser instalado y se inicia automáticamente dejándolo listo para enviar datos, los temas que pueden publicarse no solamente son del tipo numérico, provenientes de sensores del MPS 500, también son alfanuméricos es decir cadenas de texto.

C. Lenguaje de programacion (Python)

Como ya se mencionó la versatilidad de Python para desarrollar aplicaciones de IoT es innegable porque posee compatibilidad con las librerías de MQTT y el servidor de mensajería, y para lograrlo lo único que hace falta es que las librerías estén dispuestas dentro de su IDE en Rapberry, el paquete para que puedan escribirse los algoritmos llevan el nombre de *paho-mqtt* y está disponible en el repositorio digital del desarrollador. [13]

D. IDE Arduino

Arduino es una plataforma electrónica, sin mencionar que es la más popular, de código abierto y fácil de usar[14] presta sus bases de software abierto para que otros fabricantes puedan desarrollar sus placas, como es el caso de las tarjetas controladoras ESP32 pues estas no cuentan con un entorno de desarrollo propio.

Los repositorios de Arduino tienen tanto software aprobado que se encuentra dentro del entorno del desarrollador, como no aprobado que se puede hallar en la web, en el caso de este artículo las tarjetas ESP32 no poseen librerías aprobadas, a la fecha, en el IDE Arduino pero se las puede encontrar dentro del repositorio digital *github* que es un dominio conocido donde programadores y desarrolladores prestan las librerías compatibles con las ESP32, la librería está bajo el nombre de *arduino-esp32*.

Para el paquete de MQTT en el repositorio de *github* se lo localiza con el nombre de *pubsubclient*, aunque también está disponible en el gestor de librerías de IDE Arduino.

E. Gestor de base de datos (PHPMyAdmin)

PhpMyAdmin es una herramienta sencilla para administrar y gestionar bases de datos su instalación es relativamente sencilla. [15] Se pueden registrar datos de tipo entero, chars, strings, etc mismo que son necesarios para guardar los identificadores de las piezas almacenadas en el proceso del MPS 500.

Antes también se debe contar con un servidor web como *Apache* con PHP y una base de datos como *MySQL* disponibles en varios repositorios similares a los puntos anteriores y la librería para gestionar *Python* y *MySQL* lleva el nombre de *PyMySQL*.

PhpMyAdmin brinda la versatilidad que se necesita para acceder a la base de datos y crear una nueva compatible con esta aplicación de IoT en este trabajo, pues con cualquier navegador se debe escribir la dirección *http://DIRECCION_IP/phpmyadmin/* reemplazando *DIRECCION_IP* con la correspondiente a la placa que funciona como servidor, en este caso la tarjeta maestro Raspberry Pi 3 lleva también esa función de servidor, pueden separarse sí, pero se toma en cuenta el concepto de optimización de recursos.

PhpMyAdmin cuenta con el registro de usuario y contraseña, definidos por el programador, para proteger la gestión de los datos. Para crear una nueva base de datos se debe incluir un nuevo nombre y una cadena de caracteres *utf8_spanish_ci* esto nada más para que admita caracteres especiales como la <<ñ>> para que albergue una nueva tabla con la cantidad de campos que merezca la aplicación como el nombre del campo, el tipo de dato, y la longitud del dato máximo es de 9999 caracteres, la arquitectura solo demanda estos antecedentes: color de pieza, posición, fecha.

HARDWARE

A. Dispositivo maestro (Raspberry Pi 3)

La tarjeta Raspberry Pi 3 tiene pines GPIO (*General Purpose Input/Output*) de entrada y salida de propósito general y mediante estos se tiene una interfaz para controlar el conveyor, pues el uso de estos pines habilitan su avance y detención para encaminar la pieza de la estación de distribución hasta la estación de clasificación. [16]

Las tarjetas Rapberry´s fueron seleccionadas pues existe un amplio repositorio de librerías y guías sobre el uso del protocolo MQTT.

B. MPS 500

El MPS-500 (*Modular Production System*) es una estación de producción flexible, compatible, modular y versátil de marca FESTO. Forma la base para la formación técnica en general, utilizando problemas prácticos de aplicaciones reales. Permite la interacción de la mecánica, neumática, ingeniería eléctrica, tecnología de control y las interfaces de comunicación. [16]

Las estaciones presentan similares características en sensores, pues según su naturaleza son inductivos y capacitivos para leer color, magnéticos como finales de carrera y sensores de presencia para detectar objetos, de la misma forma en actuadores neumáticos se hallan vástagos de simple y doble efecto, elementos de succión y expulsión de aire así como también motores de corriente directa para las bandas. Los sensores y actuadores serán conectados a una placa PCB desarrollada por los autores de este documento con elementos que soporten *24V* para aislar elementos de potencia y control.

C. Cliente ESP32

ESP32 de 38 Pines es una placa de desarrollo que integra el micro-controlador. Esta placa permite controlar todo tipo de sensores, módulos y actuadores mediante WIFI, para proyectos de Internet de las cosas de forma eficiente y económica. El módulo posee un regulador de voltaje que permite ingresar 5V por el puerto USB, de igual manera puede ser alimentado con 3.3v en los pines de 3.3v y GND.[17]

Puede ser programado con gran variedad de software´s, lenguajes de programación[17] y como ya se señaló el entorno de desarrollo escogido fue IDE Arduino.

D. Broker (Raspberry Pi 4)

El medio por el cual todos los mensajes son publicados esta administrado por la Raspberry Pi 4 y fue escogida por sus capacidades físicas respecto a su antecesor, en MQTT la capacidad teórica de clientes conectados alcanza los 10000[18], y los paquetes de publicación y suscripción no demandan de un dispositivo más robusto haciéndolo idóneo para administrar el tráfico en esta red del MPS 500.

III. IMPLEMENTACIÓN Y DISEÑO

El diseño de la red para implementar el protocolo MQTT requiere de una arquitectura donde el bróker (Raspberry Pi 4) pueda soportar una topología de estrella y que no demanda excesivos recursos de hardware para la transmisión de temas en dicho protocolo, además que sea capaz de gestionar la red manteniendo activo el canal para que los clientes (suscriptores y publicadores) tengan activa siempre la comunicación y mantenerlos siempre enlazados a él, pues este mismo dispositivo es el nodo central de la red, remítase a la figura 2.

Las tarjetas controladoras (ESP32) y el dispositivo maestro (Raspberry Pi 3) nunca quedan inactivas dentro de esta red aunque no se transmitan mensajes. El bajo consumo energético y su disponibilidad en el mercado facilitan la instalación dentro de las estaciones de clasificación, distribución y conveyor.

El proceso empieza cuando se publica un mensaje desde el maestro, eso quiere decir, el usuario al seleccionar un color de pieza, estará publicando un tema (topic) el cual será enviado hacia el bróker y este a su vez distribuirá de forma paralela a los suscriptores en la estación de distribución y clasificación tal como se muestra en la figura 2.

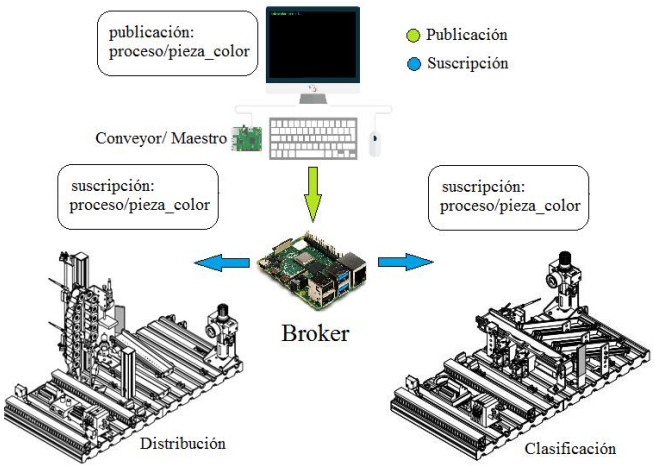


Figura 2. Publicación de mensajes MPS 500.

A. *Estación de distribución*

La tarjeta controladora ESP32 correspondiente a la estación de distribución, al estar suscripta, tiene la capacidad de recibir el tema de color antes publicado, dando paso al proceso de funcionamiento de los actuadores y sensores involucrados en el proceso de dicha estación, posteriormente la tarjeta de control ESP32 publica un nuevo tema hacia el bróker, habilitando de esta manera el transporte de la pieza por medio de las salidas físicas del maestro moviendo así el conveyor, tal como se observa en el diagrama de la figura 3. Caso contrario si la lectura de los sensores es diferente al tema publicado se procede a desechar la pieza y nuevamente el usuario debe publicar un nuevo color.

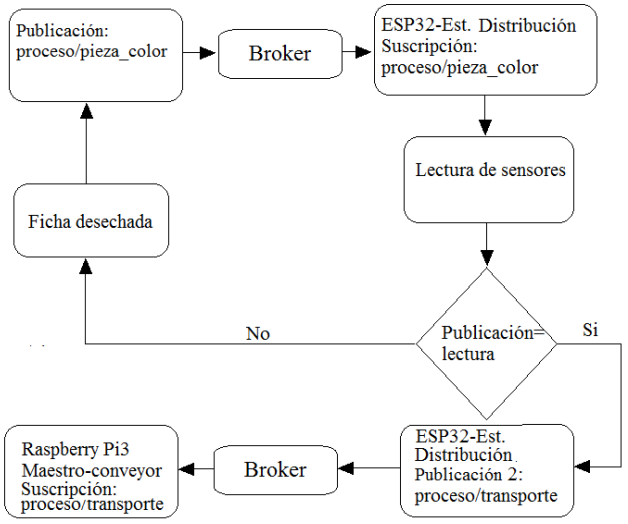


Figura 3. Diagrama estación de distribución.

B. *Estación de conveyor*

La unión física entre las estaciones de clasificación y distribución está regido por el conveyor del MPS 500 y controlado por la Raspberry Pi 3, esta estación no realiza ningún proceso que involucren sensores o actuadores a más de la movilización de la ficha, está catalogado como el dispositivo maestro porque la terminal incluida dentro de la Raspberry funciona como una interfaz visual entre máquina y usuario, dentro de su algoritmo se ejecuta el código de avance del conveyor para el transporte de la pieza por medio de la publicación de mensajes provenientes de la estación de clasificación y como MQTT es bidireccional el dispositivo maestro-conveyor se convierte también en publicador y suscriptor de sus mismos temas para la interrupción del transporte como se describe en el diagrama de la figura 4, no obstante al finalizar el recorrido de la ficha entre en ejecución el funcionamiento de la estación de clasificación cuyo objetivo es almacenar las piezas.

Así mismo la Raspberry Pi 3 alberga el gestor de la base de datos de phpMyAdmin, en donde las características de las fichas son guardadas cuando finalmente la estación de clasificación haya cumplido su cometido.

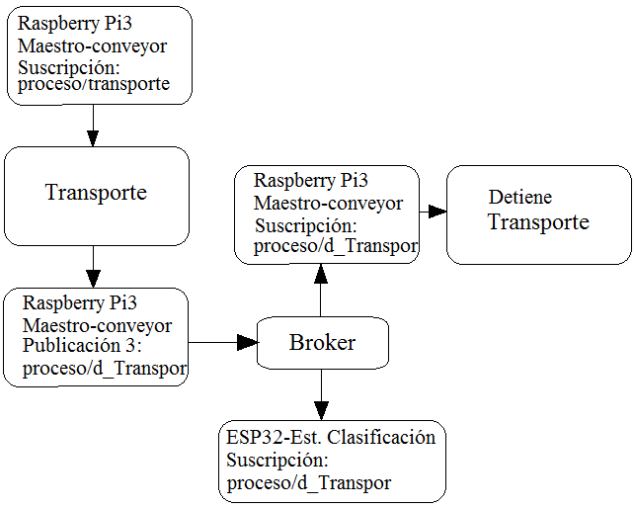


Figura 4. Diagrama estación de conveyor.

C. *Estacion de clasificacion*

En la última etapa se reitera en la implementación de la tarjeta controladora ESP32, su proceso inicia cuando el transporte del conveyor ha finalizado su recorrido, conjuntamente con el color anteriormente publicado, los sensores y actuadores de la estación de clasificación mueven y almacenan las piezas en tres diferentes posiciones, para finalmente publicar el último mensaje hacia el maestro con el objetivo de registrar los datos en phpMyAdmin. Los temas que maneja esta estación para su funcionamiento son descritos como se muestran en el diagrama de la figura 5.

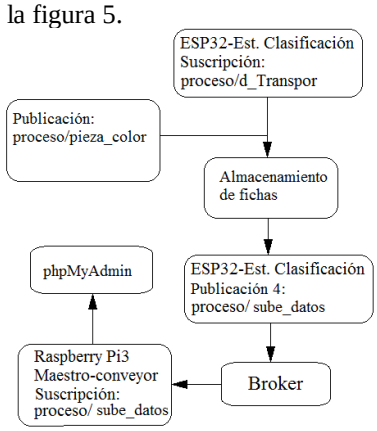


Figura 5. Diagrama estación de clasificación.

Las estaciones concluyen su funcionamiento una vez los datos sean registrados en phpMyAdmin y tanto los actuadores, variables como contadores, temporizadores y demás reinician sus valores y posiciones iniciales dando paso a espera de una nueva orden de color de ficha.

Como MQTT funciona bajo la pila TCP-IP las tarjetas ESP32 y Rasberry´s Pi deben poseer una dirección IP que las identifique dentro de la red.

El proceso fue desarrollado a partir de estos parámetros:

- Lenguajes de programación: Arduino IDE, Python.
- Protocolo: MQTT.
- Sistema embebido: Raspberry Pi 3B y 4 /Raspbian, Esp32.

IV. RESULTADOS

A través de la arquitectura desarrollada en este artículo y la utilización del protocolo MQTT entre M2M (maquina a máquina), se logró obtener un promedio de respuesta, en la publicación suscripción de paquetes, de 3,59 milisegundos desde cualquier dispositivo involucrado en la red de comunicación inalámbrica con topología estrella como: ESP32-Distribución, ESP32-Clasificación y Raspberry Pi 3-Maestro, hacia la Raspberry Pi 4-Broker, tal como muestran los resultados obtenidos desde el terminal del maestro, ver figura 6.

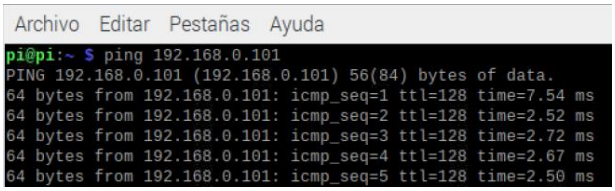


Figura 1. Tiempo de respuesta recopilado desde el terminal hacia el bróker.

Así, si lo comparamos con el estándar de PROFIBUS que permite una transferencia de entre 1,2 bytes a 1500 bytes por cada milisegundo[19], el protocolo MQTT realiza la transferencia de entre 17,82 bytes por cada milisegundo (64 bytes/3.59 milisegundos aproximadamente) situando al protocolo MQTT en un margen idóneo para comunicaciones entre dispositivos.

Para conocer el número total de paquetes publicados en la red antes mencionada, se utilizó la aplicación phpMyAdmin, la cual permite gestionar la base de datos desde el servidor local (Maestro) mostrando la información en un tiempo de consulta de 0,7 milisegundos. Es así que; mediante una verificación experimental, de un total de 63 temas publicados, todos fueron recibidos y almacenados con éxito tal como muestra la figura 7.

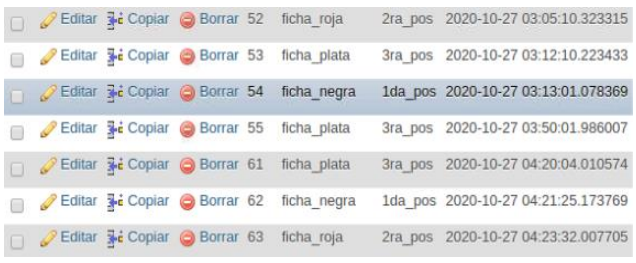


Figura 7. Verificación de paquetes recibidos y almacenados utilizando phpMyAdmin.

La figura 7, también muestra el número de cada fila, el número total de piezas, el color y la posición donde fue almacenada dentro de la estación de clasificación, indicando adicionalmente la fecha y hora de finalización del proceso.

Se realizaron pruebas experimentales, ocasionando desconexión de dispositivos y con eso fallas en la red, logrando observar en la figura 8, que la tasa de recuperación promedio es aproximadamente 1,3 segundos a la reconexión.

```
conectando a: CNT FIBRA JOSE ADRIAN
.
WiFi conectado
direccion Ip:
192.168.1.8
Conexion perdida volviendo a conectar...
Restalecido 1,3 segundos aprox
conectado
```

Figura 8. Reconexión de la red.

Mediante la aplicación para testeo del protocolo MQTT, los usuarios de la red pueden verificar los paquetes que se están enviando en ese momento y así conocer el estado actual de cada actividad del proceso, como; el color de la pieza y su posición tal como se observa en la figura 9.

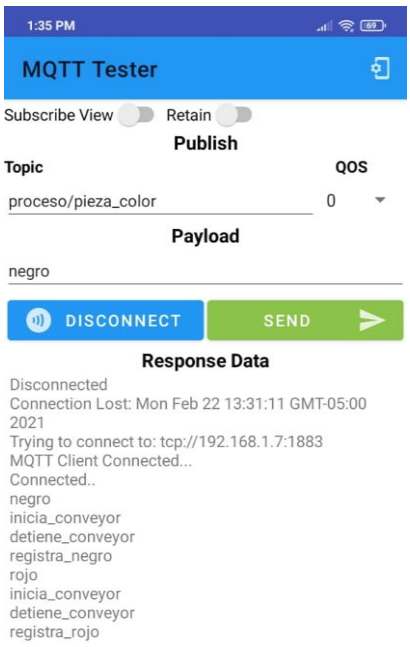


Figura 9. Monitoreo de datos desde una aplicación móvil.

V. CONCLUSIONES

La implementación del protocolo MQTT dentro del Sistema de Producción Modular parece no muy distintiva a una programación y comunicación convencional de PLC´s, sin embargo si se lo compara con el protocolo de comunicación industrial PROFIBUS, el protocolo MQTT realiza la transferencia de datos dentro del margen permitido para comunicaciones industriales, lo que lo convierte en un

protocolo de comunicación con las velocidades acceptables en procesos no críticos como el caso del tema de este artículo, además que el 100% de pruebas de mensajes publicados no presentaron perdidas y todos fueron recibidos y registrados en la base de datos.

La rápida capacidad de volver a conectarse a la red en caso de una caída permite que los procesos sean recuperados, los paquetes no sean perdidos y el MPS 500 continúe con su ejecución de clasificación de piezas, pues como se muestra en la figura 8 en caso de fallas de la red el protocolo MQTT se recupera en 1.3 aproximadamente, haciéndolo ideal para el desarrollo de este trabajo, pues clasificar y registrar fichas en una base de datos no requiere de un protocolo más robusto ni respuestas más rápidas por parte de las tarjetas ESP32 como velocidad de procesamiento o capacidad de memoria más amplia..

Como IoT permite interconectar diversos aparatos y no solamente a aquellos que se involucran directamente en el nivel de campo, sino también los dispositivos móviles pueden visualizar los temas publicados por medio de aplicaciones desarrolladas para la compatibilidad del protocolo MQTT, como se muestra la figura 9.

REFERENCIAS

[1] Y. Upadhyay, A. Borole, and D. Dileepan, "MQTT based secured home automation system," *2016 Symp. Colossal Data Anal. Networking, CDAN 2016*, pp. 1–4, 2016, doi: 10.1109/CDAN.2016.7570945.

[2] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, and D. Boyle, *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*. 2014.

[3] X. Liu, T. Zhang, N. Hu, P. Zhang, and Y. Zhang, "The method of Internet of Things access and network communication based on MQTT," *Comput. Commun.*, vol. 153, no. January, pp. 169–176, 2020, doi: 10.1016/j.comcom.2020.01.044.

[4] "MQTT, el nuevo modelo de comunicación para la industria 4.0." [Online]. Available: http://www.automaticaeinstrumentacion.com/es/noticias/2020/04/mqtt-el-nuevo-modelo-de-comunicacion-para-la-industria-4.0-46436.php#.YDFHzUS23IU.

[5] M. Antonio and F. Gonz, "Marcelo Antonio Figueroa González Escuela de Ingeniería Eléctrica Facultad de Ingeniería," 2018.

[6] A. W. Services, "IoT Núcleo de AWS Developer Guide."

[7] A. G. MORILLO RUANO, "Universidad Politécnica Salesiana Sede Quito," *Tesis*, pp. 1–100, 2017.

[8] V. A. Chalán Padilla, "DESARROLLO DE UN CONTROLADOR ÓPTIMO LQR UTILIZANDO HERRAMIENTAS IOT PARA UN SISTEMA DE PRESIÓN CONSTANTE CONTROLADO REMOTAMENTE," Universidad Politécnica Salesiana, 2020.

[9] OASIS, "MQTT Version 3.1.1," *OASIS Stand.*, no. October, p. 81, 2014.

[10] M. Learning and R. Cookbook, "APRENDIZAJE MQTT." 2017.

[11] Eclipse Fundation and Cedalo, "Eclipse Mosquitto." [Online]. Available: https://mosquitto.org/.

[12] R. A Light, "Mosquitto: server and client implementation of the MQTT protocol," *J. Open Source Softw.*, vol. 2, no. 13, p. 265, 2017, doi: 10.21105/joss.00265.

[13] Python Software Foundation, "Welcome to Python.org." [Online]. Available: https://www.python.org/.

[14] Arduino AG, "What is Arduino? | Arduino." [Online]. Available: https://www.arduino.cc/en/Guide/Introduction.

[15] The phpMyAdmin Project, "phpMyAdmin." [Online]. Available: https://www.phpmyadmin.net/.

[16] G. J. Caiza and M. V. García, "Implementación de sistemas distribuidos de bajo costo bajo norma IEC-61499, en la estación de clasificación y manipulación del MPS 500," *Ingenius*, no. 18, p. 40, 2017, doi: 10.17163/ings.n18.2017.05.

[17] esp32, "ESP32 38 Pines ESP WROOM 32 - CDMX Electrónica." [Online]. Available: https://cdmxelectronica.com/producto/esp32-38-pines-esp-wroom-32/.

[18] "Firtec - Que es MQTT." [Online]. Available: https://www.firtec.com.ar/cms/53-que-es-mqtt.

[19] Phoenix, "Perfiles de Profibus." 1993.