Evaluación del desempeño de la tecnología WiFi para el control en tiempo real de un brazo robótico

Topón Albornoz, Luis Xavier

Vicerrectorado de Investigación, Innovación, y Transferencia de Tecnología

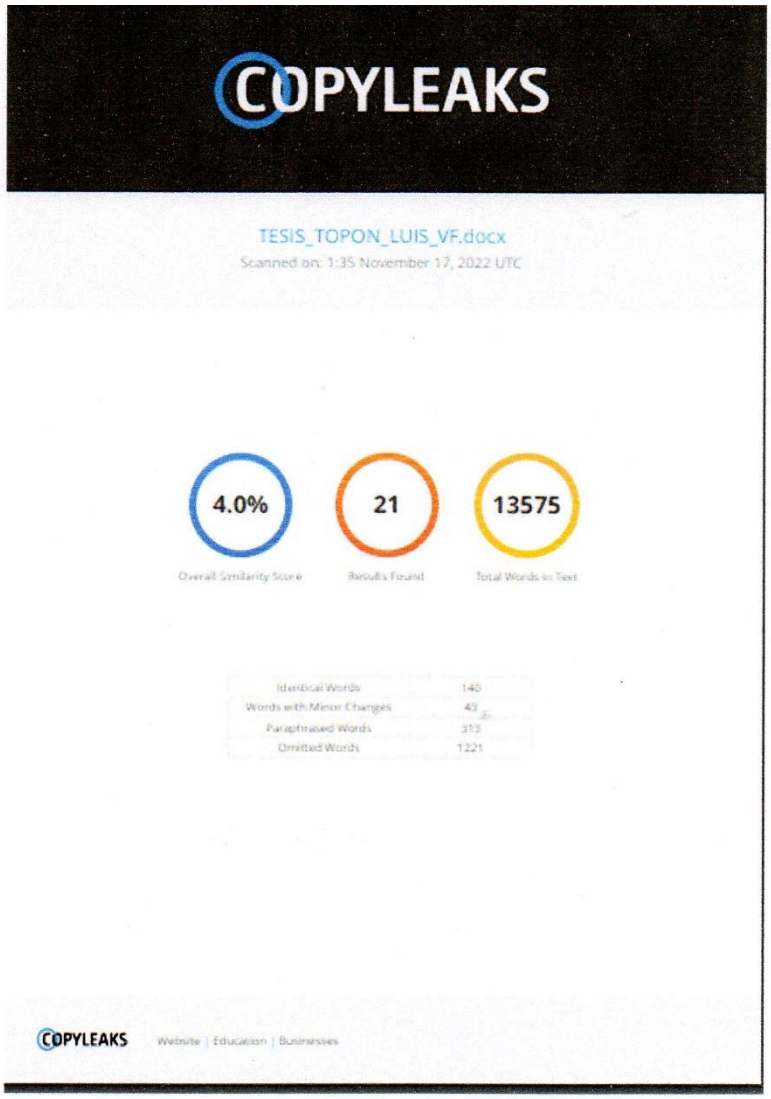
Centro de Posgrados

Maestría de Investigación en Electrónica

Trabajo de titulación, previo a la obtención del título de Magister en Electrónica mención Telecomunicaciones

Dr. Lara Cueva, Román Alcides

16 de mayo de 2023



COPYLEAKS

TESIS_TOPON_LUIS_VF.docx

Scanned on: 1:35 November 17, 2022 UTC

4.0% Overall Similarity Score

21 Results Found

13575 Total Words in Text

Identical Words	148
Words with Minor Changes	43
Paraphrased Words	313
Omitted Words	1221

COPYLEAKS Website | Education | Businesses

Firmado electrónicamente por:
ROMAN ALCIDES LARA CUEVA

Dr. Lara Cueva Román Alcides

CC: 1713988218

Director

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Certificación

Certifico que el trabajo de titulación, **"Evaluación del desempeño de la tecnología WiFi para el control en tiempo real de un brazo robótico"** realizado por el señor **Topón Albornoz Luis Xavier**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 16 de mayo de 2023

Firmado electrónicamente por:
ROMAN ALCIDES LARA CUEVA

Dr. Lara Cueva Román Alcides
CC: 1713988218
Director

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Responsabilidad de Autoría

Yo, **Topón Albornoz Luis Xavier**, con cédula de identidad No 1719690164, declaro que el contenido, ideas y criterios del trabajo de titulación **"Evaluación del desempeño de la tecnología WiFi para el control en tiempo real de un brazo robótico"** es de mí autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 16 de mayo de 2023

Ing. Topón Albornoz Luis Xavier
CC: 1719690164

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Autorización de Publicación

Yo, **Topón Albornoz Luis Xavier**, con cédula de ciudadanía N.º 1719690164 autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Evaluación del desempeño de la tecnología WiFi para el control en tiempo real de un brazo robótico"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 16 de mayo de 2023

Ing. Topón Albornoz Luis Xavier
CC: 1719690164

Dedicatoria

Esto va dedicado a mis Padres y hermano, Luis, Mariana y Roberto, quienes han sido el pilar para que esto fuera posible, ellos han sido mi inspiración para seguir adelante a pesar de mis caídas ya que con su paciencia y amor han sabido darme palabras de aliento.

Dedicado también a toda la “Comunidad Jesús es el Señor”, que, con su gran hermandad y sus enseñanzas, que cada mes comparto junto con ellos, han ido transformando mi vida, pudiendo así tener un mejor estilo de vida, viviendo en comunidad.

Dedicado para toda mi familia, Tíos, tías primos y primas, que con su gran amor y consejos han ido formando a un mejor ser humano.

Topón Albornoz Luis Xavier

Agradecimientos

Darle gracias a Dios por todo lo que ha podido hacer en estos 8 años por mí y por mi familia, Él ha sabido formarnos y unirnos más, gracias por permitir que mi Padre Luis este aquí el día de hoy, ya que permitiste que se cure de su enfermedad pues tu promesa la cumpliste de que él iba a seguir junto con nosotros y pues yo estaré siguiéndote a donde tú me lleves, gracias Papá Luis porque con tu gran ejemplo hiciste de mi un gran hombre a pesar de todos los problemas que te cause siempre creíste en mí en que yo era capaz de llegar lejos y ahora estoy aquí dándote una gran alegría de verme graduado al servicio de Dios y aun me faltan muchos logros que cumplir que espero que estés presente.

A ti Madre Mariana dulce y bella con tú forma de ser hiciste que pueda superar cada complejo de mi vida gracias por ser esa gran amiga por escucharme y aguantarme cada berrinche que tenía siempre me complaciste en todo lo que te pedía gracias por estar acá junto a mí en este día tan feliz de mi vida, solo te puedo decir perdóname por hacerte pasar casi la mitad de tu vida en un hospital cuidándome de cada operación que tenía no sé porque habré nacido así pero eso ya no me importa, ahora lo más importante es que usted sea feliz y vea en mi a una gran persona y se sienta orgullosa de mí.

A ti hermano mío solo decirte que te quiero mucho, aunque somos tan diferentes en temperamentos y en forma de ser, pues Dios hizo que tengamos algo en común y es que el día de hoy lo adoramos solo a Él y si hablamos el tema es solo Dios, gracias por cuidarme desde pequeño y perdón por cada lagrima que te hice derramar en algún momento de tu vida, solo cuídate y sigue adelante.

Doy gracias a mi mentor el Dr. Román Lara quien con su sabiduría me ha sabido guiar en este proceso de titulación con el cual se logró un paso más en la investigación para el avance tecnológico en redes inalámbricas.

Gracias a mis hermanos de la “Comunidad Jesús es el Señor” saben quisiera enlistarlos a todos pero me faltarían hojas, mis hermanos y hermanas gracias por todo lo que han hecho por mí por sus palabras por que pude ganarme su amistad, yo sé que he cometido varios errores con algunos de ustedes pero sepan que gracias a ello pues ahora comprendo cuán grande es Dios, el poder compartir un mismo ideal, una misma misión y visión es lo mejor que me ha pasado en estos últimos 10 años, si en un futuro me pidieran que pida un deseo pues ese sería que quisiera que ustedes junto conmigo siempre fuéramos jóvenes para así seguir viendo aventuras y seguir riéndonos y compartiendo este llamado a la santidad que aún nos falta por recorrer sin más que decirles pues espero que sigamos en este camino que escogimos desde nuestro primer compromiso donde reafirmamos el llamado que Dios nos hizo.

Topón Albornoz Luis Xavier

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

La tele operación en los últimos años ha tenido un gran avance tecnológico y el desarrollo de estos prototipos controlados remotamente permiten realizar trabajos donde la mano humana tenga inconvenientes, la limitante de estos sistemas es el retardo en la transmisión de datos y procesamiento de la señal, en este contexto el presente trabajo de titulación se basa en evaluar el desempeño de la tecnología WiFi en concordancia con el estándar IEEE 802.11n para el control en tiempo real de un brazo robótico, este prototipo es controlado por *flex sensor* que van ubicados en el brazo humano como adquisición de datos, estos datos son procesados por una tarjeta controladora ESP32 que envía a otra tarjeta ESP32 por medio de la señal WiFi a través de un servidor MQTT, la cual permite el control del brazo robótico con 4 grados de libertad, se implementa una cámara de video que genera la imagen de dicho proceso a un computador por medio de la plataforma de Node-red. La evaluación del desempeño de la tecnología WiFi se llevó a cabo el análisis de las métricas de desempeño asociadas a la calidad de servicio como el retardo y Jitter. Después de las pruebas realizadas con los proveedores Celerity y CNT se tiene que el retardo medio en un sistema en ausencia de dispositivos en su red es *83,38 ms* y *111.56 ms*, con un error estándar del retardo de *8,77 ms* y *21,05 ms,* y un Jitter de *10,37 ms* y *22,96 ms* respectivamente. En presencia de 10 dispositivos en su red se tiene un retardo medio de *142,2 ms* y *231,48 ms*, un error estándar del retardo de *40,52 ms* y *95,25 ms,* y un Jitter es de *45,59 ms* y *107,14 ms,* se corrobora que en presencia de mayor tráfico de datos el retardo y Jitter incrementan. Finalmente, en las pruebas de distancia en ausencia de dispositivos se demuestra que el retardo aumenta directamente proporcional a la distancia, mientras que el Jitter aumenta en una proporción mínima, así se tiene que en 20 y 100 m el retardo medio *85 ms* y *503 ms* con un Jitter de *9,76 ms* y *10,51 ms* respectivamente.

Palabras Clave: Retardo, Jitter, Red inalámbrica.

Abstract

Teleoperation in recent years has had a great technological advance and the development of these remotely controlled prototypes allow work where the human hand has problems, the limitation of these systems is the delay in data transmission and signal processing, In this context, the present degree work is based on evaluating the performance of WiFi technology in accordance with the IEEE 802.11n standard for real-time control of a robotic arm, this prototype is controlled by a flex sensor that is located in the arm. human as data acquisition, these data are processed by an ESP32 controller card that sends to another ESP32 card through the WiFi signal through an MQTT server, which allows the control of the robotic arm with 4 degrees of freedom, is implemented a video camera that generates the image of said process to a computer through the Node-red platform. The evaluation of the performance of WiFi technology was carried out by analyzing the performance metrics associated with the quality of service such as delay and Jitter. After the tests carried out with the providers Celerity and CNT, the average delay in a system in the absence of devices in its network is found to be *83.38 ms* and *111.56 ms*, with a standard error of the delay of *8.77 ms* and *21.05 ms*, and a Jitter of *10.37 ms* and *22.96 ms* respectively. In the presence of 10 devices in your network, you have an average delay of *142.2 ms* and *231.48 ms*, a standard error of the delay of *40.52 ms* and *95.25 ms*, and a Jitter is *45.59 ms* and *107.14 ms*, it is corroborated that in the presence of more data traffic, the delay and Jitter increase. Finally, in the distance tests in the absence of devices, it is shown that the delay increases directly proportional to the distance, while the Jitter increases in a minimal proportion, so that at 20 and 100 m the average delay is *85 ms* and *503 ms*, with a Jitter of 9.76 ms and *10.51 ms* respectively.

Keywords: Delay, Jitter, Wireless Network.

Capítulo I: Introducción

Antecedentes

En el mundo actual, la tele operación juega un papel muy importante en la prevención de riesgos. Es así que un robot puede realizar diversas operaciones complejas con mínima invasión, mayor precisión y flexibilidad, si a este se le agrega el control inalámbrico, se tiene robots de control a distancia el cual mantiene al usuario lejos del área de trabajo o sitio remoto, lo cual permite salvaguardar la seguridad del usuario y poder alcanzar sitios inalcanzables, para así aprovechar un robot el cual realiza tareas complejas con mayor precisión y flexibilidad.

Bajo este contexto se da paso a trabajos científicos que recopilan diferentes métodos para la comunicación inalámbrica por medio de protocolos de comunicación y métodos para el procesamiento de datos de entrada y salida, para las señales de entrada utilizan *flex sensor* o señales electromiografías y como salida un brazo robótico prefabricado o diseñado en 3D con servomotores. Estos trabajos científicos están clasificados cronológicamente con el fin de visualizar el avance científico en este tipo de prototipos.

Fernando Alberto Alvarado Clavijo (2011) en su tesis de “Mano robótica inalámbrica”, nos muestra una mano robótica diseñada con servomotores la cual emula el movimiento de los dedos de una mano para lo cual utiliza *flex sensor* en un guante como dispositivos de entrada de datos, en este proyecto para la comunicación inalámbrica utiliza el módulo XBEE que son pequeños radios que se comunican inalámbricamente ente sí y está basado en el estándar IEEE 802.15.4 teniendo la posibilidad de realizar conexiones Punto a Multipunto, una de las ventajas de estos módulos es que consumen muy poca energía. En este proyecto se menciona que la mano robótica presenta bastante retardo en su comunicación, mas no presentan un análisis de métricas

de desempeño respecto al retardo, pero nos sirve como punto de partida para la utilización de *flex sensor* como datos de entrada al sistema (Alvarado Clavijo, 2011).

Óscar Cortés Diart, Guillermo Sánchez, Javier Roldán Mckinley, Eugenio Yime Rodríguez (2014) en su artículo “RSM en la minimización del retardo en la red durante tele operación de robots” da a conocer por medio de su artículo que el retardo medio no depende del tiempo de la comunicación, pero se muestra altamente dependiente de la hora a la que se realiza la comunicación y de la densidad del paquete enviado, representado en él los experimentos por el tiempo entre envíos. El retardo varía de modo inversamente proporcional al tiempo entre envíos y es directamente proporcional a la hora. Dentro de un área de experimentación comprendido entre las 10:00 AM y las 5:00 PM del día, y desde 0.02 hasta 0.1 segundos de tiempo de envío, la condición de operación que produce los mínimos retardos en la comunicación es la combinación 10:00AM y 0.1 segundos, con un valor de retardo esperado de 26.4 ± 9 milisegundos, con un rango de confiabilidad del 95%, al visualizar esto se tiene en consideración que para las pruebas del proyecto se deben realizar en horas donde el tráfico de datos sea mínimo para obtener buenos resultados (Cortés Diart et al., 2014).

Ms. Puja Dhepekar (2017) en su artículo “Mano robótica inalámbrica para operaciones remotas usando *flex sensor*” se basa en la posición de la mano para controlar el sistema robótico, el cual utiliza un *flex sensor* que es colocado en el guante que al momento de ponerlo en el brazo humano cada movimiento que realiza la mano es emulada por el brazo robótico. Según diferentes posiciones del brazo, la resistencia del *flex sensor* cambia, y este cambio en la resistencia se utiliza para mover el eje del servomotor que mueve el robot según la posición del *flex sensor*, este es un elemento que se usa como adquisición de datos. Para la parte de comunicación inalámbrica utiliza el protocolo ZigBee que se basa en el estándar IEEE 802.15.4 la cual tiene como características no consumir demasiada energía y presenta un retardo mucho mayor porque

tiene una tasa de transmisión muy baja, por medio de estas características, ZigBee tiende a ser una solución viable en la comunicación inalámbrica que permita el control con sensores. Basado en esto ZigBee mantiene una conexión estable y dinámica. Gracias al proyecto que utiliza un guante antiestático para no producir ruido en los *flex sensor* proporciona información para poder realizar el presente proyecto con dichos guantes y no generar ruidos innecesarios y respecto a un análisis de retardos en el sistema de comunicación no presenta resultados (Puja & Yashwant G., 2017).

Jhosep Edgar Andrei Hower Ceballos, Carlos Andrés Correa Arroyave, Carlos Uriel Pareja Rodríguez (2019) en su tesis “Análisis y diseño de un prototipo de una mano robótica con catorce grados de libertad, capaz de ser dirigida a través de Internet en tiempo real”, nos proporciona información sobre la conexión por Ethernet para el envío y recepción de datos entre dos máquinas y conectados por el puerto USB la adquisición de datos y la mano robótica respectivamente, en su análisis nos indica que la interfaz de USB tiene una mayor velocidad de transmisión de datos, sin contar con el beneficio de que esta interfaz es automáticamente reconocida por el ordenador, esta mano robótica está controlada por una interfaz web por medio de una arquitectura Cliente/ Servidor el cual se conecta a la red remota y por medio de una dirección IP la información se traslade de un punto a otro, donde los Microcontroladores procesan la información y esta da pie a los movimientos de la mano robótica (Hower Ceballos et al., 2019).

Manuel Arias Montiel, Asis Martínez Miguel, Ester Lugo González, Rosebet Miranda Luna, Ricardo Tapia Herrera (2021) en su artículo “Prototipo de mano robótica controlado mediante señales electromiografías con un dispositivo comercial” en este proyecto se muestra el procesamiento de las señales electromiografías a través del software Matlab, permitiendo así en control de un brazo robótico el cual está diseñado en 3D con servomotores, para la comunicación de datos es realizada por un Circuito Inter-Integrado (I2C del inglés, *Inter integrated circuits*) entre

2 módulos de Arduino teniendo velocidades de comunicación que van desde 100 kbps a 400 kbps (Arias Montiel et al., 2021) .

En base a estos proyectos donde utilizan diferentes maneras de comunicación para el control de un robot, se observa que aún existe mucho por investigar y es un ante sala para el futuro, en el cual todo esto sea algo integral con el fin de que la tele operación de sistemas inteligentes sea ya parte del ser humano y se pueda seguir con más avances científicos, cometiendo el menor error posible respecto a la comunicación inalámbrica de datos. No obstante parte de esta investigación del presente proyecto considera al protocolo de comunicación MQTT (de las siglas, *Message Queuing Telemetry Transport*), el cual sirve como puente para él envío de datos por Internet, para lo cual se ha considerado el siguiente artículo científico en el cual utiliza módulos Arduino para el envío de datos a través de este protocolo y nos servirá como parte de la investigación del presente proyecto.

Mario Calleja Collado, Joan Guasch Llobera (2020) en su artículo “Monitorización de sensores con Arduino utilizando el protocolo MQTT”, nos muestra el procesamiento de sensores de temperatura y humedad como datos de entrada y un led como dato de salida, algo muy importante a resaltar en el proyecto es el grosor del tráfico, el cual es generado por los dispositivos M2M (de las siglas, *Machine to Machine*) lo que nos quiere decir un intercambio de información entre máquinas, donde la mayor parte de datos en el tráfico de información sea por dispositivos conectados en casa. La misma empresa, CISCO, prevé que sea un total del 50% del tráfico total generado por estos dispositivos. Estos datos tienen que ser procesados por los diferentes protocolos y estándares de comunicación. Y como se ha mostrado en este proyecto, uno de los más importantes es el MQTT. Que utiliza bloques de publicación y suscripción los cuales se utilizan con jerarquías haciendo que los clientes puedan enviar y recibir mensajes, este protocolo generalmente se usa en sistemas embebidos porque puede hacer que los sensores se

comuniquen hasta el sistema, de manera liviana a través del cable siendo fácil de implementar (Faidallah et al., 2015).

Mejores tecnologías se han desarrollado con el paso del tiempo en la actualidad el uso de WiFi, Bluetooth, entre otras, las cuales han ido en aumento y posicionándose para el uso en diferentes proyectos tanto a nivel académico como profesional, por medio de un tipo de microcontrolador llamado MCU ESP32 el cual viene integrado con la tecnología WiFi entre otras características, este dispositivo posee 2 núcleos que puede fácilmente intercambiar información desde la nube y de manera simultánea administrar datos de uno o varios sensores de manera precisa. La frecuencia de operación de estos microprocesadores es hasta 240 MHz con un alto nivel de integración otra de las ventajas que posee es un consumo de energía muy bajo a través de funciones de ahorro todo esto la convierte en una excelente herramienta para proyectos y aplicaciones directas a Internet de las Cosas (IoT del inglés, *Intnernet of Things*). (Misal et al., 2020).

Justificación e importancia.

La tele operación hoy en día juega un papel crucial en el campo de la investigación ya que permite el desarrollo de diversos prototipos que realizan trabajos donde la mano humana tenga inconvenientes o donde un operario corra el riesgo de sufrir un accidente, para esto se necesita de equipos que se puedan usar a distancia controlado de forma remota. Aunque la tecnología WiFi tiene muchas ventajas en gran variedad de campos, al momento de tele operar robots, uno de los principales retos que surgen en el diseño de sistemas de tele operación es el retardo de tiempo incierto, la pérdida de datos y la seguridad de la transmisión de datos. Solo la presencia de retardo de tiempo en el sistema puede inducir inestabilidad o mal desempeño en la comunicación hombre-máquina, aún más la pérdida de datos. Además, que el medio no guiado es el aire el cual afecta la potencia de la señal recibida, ya que se presentan fenómenos

como la lluvia viento u obstáculo, también es conveniente estimar algunas características del canal utilizado el cual no debe estar solapado con algún canal adyacente con el fin de tener un mejor desempeño en el control del prototipo.

En base a esto el presente proyecto está enfocado en el análisis de las métricas asociados a la calidad de servicio (QoS, del inglés *Quality of Service*) del sistema de comunicación en el manejo de un brazo robótico por medio de una red inalámbrica el cual emula los movimientos humanos con los que se puedan desarrollar de manera exacta tareas que solo los seres humanos las pueden hacer, pero sin riesgo y con elevada precisión, razón por lo cual, el tema del proyecto a elaborar tiene como fin el ampliar los conocimientos y beneficios en el área de la robótica en la industria o medicina entre otros proyectos. Además, el proyecto permite impulsar la elaboración de prototipos enfocados a la robótica y a sus innumerables avances tecnológicos en la tele operación, lo que permite la mejora tecnológica al momento de transmitir y receptar datos, pero sobre todo el desarrollo del mismo servirá de plataforma para elaboración de nuevos proyectos futuros de la misma clase tele operativa teniendo como principal objetivo la minimización de retardos en el sistema. El brazo robótico controlado inalámbricamente es un proyecto de gran interés puesto que permite inmiscuirse en el campo de la robótica por medio de un diseño e implementación del sistema de comunicación inalámbrica por medio de protocolos MQTT interconectados por la arquitectura de Node-Red donde se puede implementar la tecnología IoT, lo cual aporta nuevos conocimientos, criterios e interpretación de datos que ayuden a desenvolverse en un ambiente de trabajo acorde con las Telecomunicaciones.

Hipótesis

En este contexto y basándose en lo anterior se plantea la siguiente hipótesis:

La tecnología WiFi en concordancia con el estándar IEEE 802.11n, como base del IoT, permite controlar un brazo robótico con el menor retardo posible y ser considerado un sistema en tiempo real permitiendo el mejoramiento del desempeño hombre-robot. Con ello surgen interrogantes, ¿Podrá la Tecnología WiFi en concordancia con el estándar IEEE 802.11n transmitir datos con mínimo error y producir retardos no significativos?, ¿Qué configuraciones se deben realizar al router para maximizar su potencia de transmisión de datos?, ¿Cómo afecta el tráfico de Internet en el envío de datos en un enlace punto a punto?"

Variable independiente

- Protocolo MQTT
- Tráfico de la red
- Señal WiFi

Variable dependiente

- Retardo
- Tiempo de procesamiento de la señal
- Tiempo de respuesta del servomotor
- Tiempo de envío de datos por la señal WiFi

Alcance del proyecto.

El presente proyecto se basa en la implementación del sistema inalámbrico entre dos micro controladores por medio de la tecnología WiFi en concordancia con el estándar IEE 802.11n, para el presente proyecto se utiliza el módulo ESP32 el cual cuenta con tecnología WiFi integrado en su módulo la cual se conecta inalámbricamente con cualquier dispositivo, con el fin de controlar un brazo robótico en tiempo real, el brazo robótico viene prefabricado ya que la construcción del mismo está fuera del alcance del presente proyecto, pero se necesita saber su estructura y funcionamiento. La estructura del brazo robótico consta de 4 grados de libertad que emulan las articulaciones que permiten el movimiento entre sí, por lo que es necesario

comprender la mecánica que posee, en cuanto a los motores que se utilizan en el brazo son servos los cuales son dispositivos que funcionan con corriente continua estos mantienen un buen torque y generan el movimiento indicado e interpretan las señales enviadas por los sensores.

Para la adquisición de datos y movimientos que realiza el brazo robótico se utiliza el *flex sensor* ubicados en la mano humana por medio de un guante del cual son procesados por uno de los módulos ESP32 que hacen de transmisor. Este sistema debe ser controlado vía Internet en tiempo real desde cualquier parte del mundo utilizando el protocolo MQTT, que por medio de suscripción y publicación permite el envío de datos, y para visualizar los movimientos del brazo robótico se lo realiza por medio de una Cámara web, para el proyecto se utiliza el módulo ESP CAM en el cual viene integrado una cámara que se enlazada a un servidor de video, la imagen de video se muestra en la plataforma de Node-Red donde se visualiza el video y el tiempo de retardo del sistema. Se realizan pruebas con 3 proveedores de Internet (Celerity, Netlife y CNT) y 10 dispositivos conectados en la misma red inalámbrica, estos resultados serán procesados por medio de una regresión lineal con el fin de determinar el menor error estándar en la comunicación inalámbrica entre la mano humana y el brazo robótico, para que el brazo robótico emule los movimientos de la mano humana sin percepción a la vista del ojo humano de presencia de retardos y así tener un esquema más claro para futuros trabajos de investigación en la tele robótica.

Objetivos

Objetivo general

Evaluar el desempeño de la tecnología WiFi para el control en tiempo real de un Brazo Robótico

Objetivos específicos

- Implementar la red inalámbrica mediante protocolo MQTT para el control del Brazo Robótico.
- Programar el sistema para que cumpla con el funcionamiento adecuado y permita el movimiento del Brazo robótico.
- Realizar pruebas de inyección de tráfico invasivas y no invasivas para el análisis de funcionamiento del prototipo, mediante la comprobación de las variables que influyen en el sistema de comunicación inalámbrica.
- Analizar el tiempo de respuesta de cada servomotor una vez recibida la instrucción.
- Analizar el tiempo total de respuesta del sistema por medio de la inyección de tráfico en la red inalámbrica.

Estructura de tesis

Como se sabe todo documento presenta una estructura, en el presente proyecto se compone de varios capítulos los cuales están distribuidos de la siguiente manera:

El capítulo I describimos una introducción donde como primera parte se revisa los antecedentes donde se presentan trabajos previos en la comunicación para el control de un brazo robótico, como segunda parte se describe la justificación, alcance y objetivos del presente proyecto.

En el capítulo II se presenta el fundamento teórico, donde se describe los conceptos teóricos de cada parte que conforma el presente proyecto y su funcionalidad.

El capítulo III describe el desarrollo e implementación del sistema inalámbrico para el control en tiempo real de un brazo robótico En la primera parte la implementación del protocolo MQTT en el servidor. En la segunda parte se realiza el algoritmo para cada módulo ESP32 y

comunicación MQTT con el servidor. En la tercera parte se realiza un filtrado de la señal de los *flex sensor* que no generen datos basura. Como parte final del capítulo se realiza la comunicación con ESP32 CAM para la visualización en tiempo real del proceso.

En el capítulo IV se muestra el análisis de resultados del desempeño de las métricas asociadas al QoS en tiempo real, con diferentes servidores de internet y pruebas de distancia de comunicación.

Finalmente, en el capítulo V se presentan las conclusiones y trabajos futuros que se pretende seguir desarrollando en este campo de investigación.

Capítulo II: Marco teórico

Brazo robótico

Es un sistema mecánico que puede ser programado para que cumpla funciones parecidas a las de un brazo humano, estas pueden ser parte de un robot más complejo o ser independiente (Esneca, 2022).

Sus diferentes partes se unen y conectan a unos servomotores para hacer sus movimientos de rotación y traslación, estos pueden ser servomotores para aplicaciones en electrónica como servomotores para las industrias, con esta implementación en estos brazos el uso se amplía desde pequeños proyectos hasta grades empresas que puedan realizar tareas más delicadas y complejas **¡Error! No se encuentra el origen de la referencia.** (Esneca, 2022).

Figura 1.

Brazo Robótico

Nota, tomado de la página web de robotic,2022.

Servo motor

Es un servo mecanismo de bucle cerrado que utiliza la retroalimentación de posición para controlar su velocidad de rotación y posición (Abro et al., 2019).

La modulación por anchura de pulso PWM (del inglés, Pulse Width Modulation), es uno de los sistemas más empleados para el control de los servos. Este sistema consiste en generar

una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo periodo (normalmente), con el objetivo de modificar la posición del servo según se desee Figura 2 (Abro et al., 2019).

Figura 2.

Servo Motor

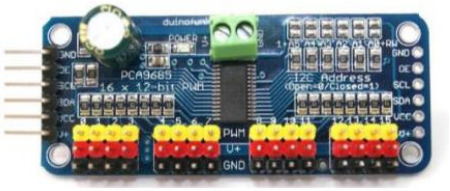
Nota, tomado de la página 533 del trabajo de *Role of modern fractional derivatives in an armature-controlled* DC servomotor, 2019.

Módulo PCA8695

El módulo PCA9685 es un sistema electrónico que permite generar 16 canales de salda PWM con una resolución de 12 bits y un canal de comunicación mediante I2C a un microcontrolador (Naylamp, 2021).

Figura 3.

Modulo PCA9685



Nota, tomado de la página web de mecatrónica, 2021.

Tabla 1.

Características del módulo PCA9586.

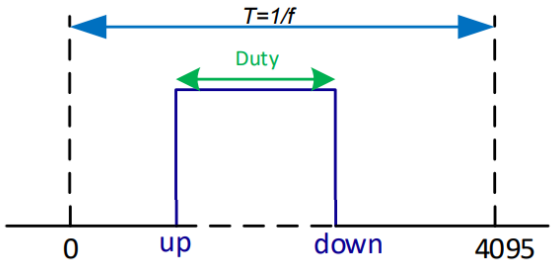
Características	
Frecuencia	1000 Hz
Número de canales	16 canales
Resolución	12 bits
Voltaje	DC 3-5 voltios
V Max +	6 voltios
Tamaño	60X25 mm

Nota, tomado de la página web de mecatrónica, 2021.

En la Figura 4 se puede identificar el *duty Cycle* generado acorde a la resolución de 12 bits (4096 niveles) y el periodo T. Si la alimentación del módulo PCA9685 es de 5 voltios, el periodo de 1 segundo y el pulso de trabajo es de una resolución de 2047 niveles, estaríamos hablando que el voltaje RMS es de 2.5 voltios, donde se puede variar el *duty cicle* y obtener variaciones de voltaje y corriente en la carga (Naylamp, 2021).

Figura 4.

Duty Cycle



Nota, tomado de la página web de mecatrónica, 2021.

Tarjeta controladora ESP32

Es un chip integrado (SoC, del inglés *System on Chip*), con conexión WiFi y compatible con el protocolo TCP/IP. Fue diseñado por la compañía *Espressif System* en su sede en

Shanghái, es un chip autómata que ofrece soluciones en proyectos de red WiFi (Ceja et al., 2017).

Tabla 2.

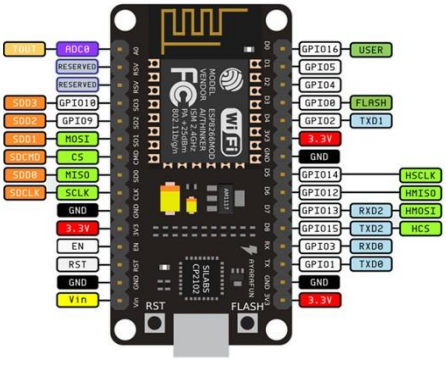
Características del módulo ESP32.

Características	
Voltaje de operación	3.3 V
Consumo de Corriente	10 µA – 170 ma
Memoria Flash	16 MB máx. (512 k normal)
Procesador	Tensilica L106 32 bits
Velocidad del procesador	80 -160 MHz
GPIOs	17
Analógico a Digital	1 entrada 10 bits
Soporte de 802.11	b/g/n/d/e/i/k/r
Máximas conexiones Simultaneas	5

Nota, son características tomadas de la página 24 de Módulo ESP8266 y sus aplicaciones en el internet de las cosas, 2017.

Figura 5.

Pines ESP8266



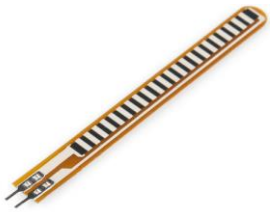
Nota, es una imagen con todos los pines tomadas de la página 24 de Módulo ESP8266 y sus aplicaciones en el internet de las cosas, 2017.

Flex sensor

Los *flex sensor* son elementos electrónicos que determinan el grado de flexión de un material, para determinar el grado de flexión de un dispositivo o un material es por medio de cambios en la resistividad asociada a dicho elemento. Existen sensores comerciales que realizan ese cometido con una precisión bastante aceptable. Estos sensores aumentan su resistividad conforme aumenta el ángulo de flexión, con valores nominales de 20 kΩ en reposo hasta 40 kΩ con flexiones de 180º (Arenas et al., 2021).

Figura 6.

Flex sensor



Nota, esta es una imagen que muestra la estructura de un flex sensor, tomado de Diseño y Construcción de un Guante de Datos mediante Sensores de Flexibilidad y Acelerómetro, 2021.

Estándar IEEE 802.11n

Es un estándar de la red inalámbrica que utiliza múltiples antenas para acelerar la transmisión de datos. El propósito de este estándar es aumentar la capacidad de la red con respecto a los 2 estándares anteriores (IEEE 802.11a y IEEE 802.11g). presenta una máxima transferencia de datos de 54 Mbps a 600 Mbps (Delta, 2022).

La velocidad del estándar IEEE 802.11n puede aumentar hasta 150 Mbps al realizar la configuración de canales en modo de 40 MHz, sino hay ninguna interferencia en la emisión, por ejemplo, Bluetooth, hornos microondas u otras redes WiFi en los alrededores, si se usan múltiples

antenas, la velocidad de IEEE 802.11n pueden alcanzar hasta 288 Mbps con la configuración de la frecuencia en 30 MHz por medio de 4 antenas y aumenta su tasa a 600 Mbps configurando la frecuencia a 40 MHz.(Delta, 2022).

La transmisión de datos con una velocidad de hasta 600 Mbps se logra mediante cuatro flujos especiales utilizando un canal con un rango de ancho de 40 MHz. Varios tipos de modulación y codificación se han determinado por los estándares y se han guardado en el esquema de modulación y codificación (MCS, del inglés *Modulation and Coding Scheme*). Con el fin de obtener la máxima eficiencia de IEEE 802.11n, es aconsejable una red limpia de 5 GHz. La banda 5GHz tiene un potencial considerable debido a los numerosos canales de radio que no se solapan y menores distorsiones con respecto a la banda 2.4 GHz (Delta, 2022).

La banda de 2.4 GHz es concurrida pero el estándar IEEE 802.11n ofrece la posibilidad de duplicar el ancho de banda por canal para 40 MHz, lo que da el doble de velocidad de transmisión de datos (Delta, 2022).

La certificación básica IEEE 802.11n incluye los canales 20 MHz y 40 MHz. Para un máximo de dos flujos espaciales, el ancho de banda es igual a 144,4 Mbps para el ancho del canal de 20 MHz, y 300 Mbps para el ancho del canal de 40 MHz (con una distancia de protección corta) (Delta, 2022).

Tabla 3.

Características del estándar IEEE 802.11n.

CARACTERISTICAS	**IEEE 802.11 n**
Throughput	600 Mbps
Frecuencia de operación	2.4 GHz / 5 GHz
Ancho de canal	20 MHz o 40 MHz
Modulación	64 QAM
MIMO	SU – SISO

Nota, Es una tabla característica del estándar IEEE802.11n sacado de la página web delta, 2022.

Protocolo MQTT

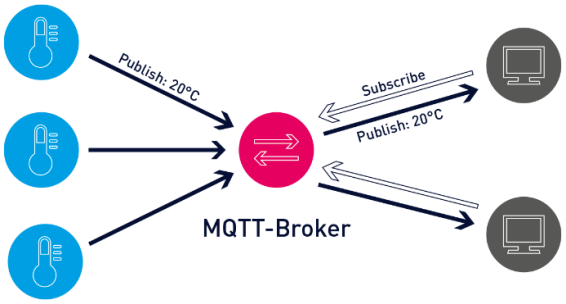
El protocolo MQTT permite una comunicación en la red, es el más utilizado ya que se puede implementar en computadores con pocos recursos a nivel de hardware y software ya que es muy ligero y sencillo de instalar, este protocolo se lo puede utilizar en tarjetas de desarrollo ya que también tiene un ancho de banda muy bajo y esto da hincapié a realizar prototipos u aplicaciones basados IoT en tiempo real (Paessler, 2022).

El protocolo MQTT es muy usado en el mundo por aplicaciones de renombre como Facebook o su Messenger ya que permite que los mensajes puedan ser entregados de manera eficiente y con un mínimo retardo en milisegundos (ms), sin tener consideración en los conexones inconsistentes del Internet en el mundo entero (Paessler, 2022).

Para poder entender la comunicación por medio del protocolo MQTT se tiene que utilizar un sistema de publicación y suscripción donde el cliente que envía información es el que publica y el cliente que recibe es el que suscribe teniendo así tópicos de comunicación con el fin de que el sistema no envié datos falsos, se tiene que tener en cuenta que se tiene 2 tipos de sistemas clientes y brókeres, donde el Broker hace de servidor donde se comunican los clientes, esto quiere decir que los clientes se pueden comunicar siempre y cuando se tenga un servidor que haga puente de comunicación (Paessler, 2022).

Figura 7.

Funcionalidad de MQTT

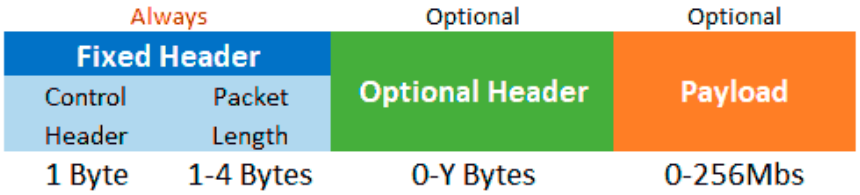


Nota, Es la parte estructural de un servidor y cliente MQTT extraído de una página IT Explained,2022.

El protocolo MQTT presenta componentes uno de estos es los mensajes para lo cual es importante definir el tipo de mensaje y su tamaño, estos mensajes constan de 3 partes:

Figura 8.

Estructura de mensaje MQTT



Nota, pertenece a la estructura de un mensaje MQTT extraído de una página web IT Explained,2022.

- **Cabecera fija**: Se tiene la cabecera que tiene un espacio de 2 a 5 bytes, los cuales son obligatorios, también cuenta con un código de control, que ayuda a identificar el tipo de mensaje a enviar y el tamaño del mansaje, Su longitud se puede codificar de 1 a 4 bytes del cual se usa 7 bits y uno de continuidad

- **Cabecera variable**. Esta parte del mensaje es opcional el cual solo se usa en ciertas ocasiones.
- **Contenido(payload)**. Es la parte donde va el mensaje real tiene un espacio de 256 Mb.

El protocolo MQTT tiene 3 niveles de QoS que ayudan a minimizar la tasa de retardo y fiabilidad entre las cuales se tiene: (Paessler, 2022).

- QoS 0: ocupa una mínima tasa de transmisión por lo cual si el mensaje es enviado este no recibe un mensaje de confirmación de la recepción del mensaje y a su vez si el cliente no está conectado cuando se envié mensajes estos no se guardan para cuando se conecte.
- QoS 1: el siguiente servicio nos ofrece que el mensaje debe ser enviado al menos una vez, esto se garantiza ya que el broker vuelve a enviar el mensaje sino recibe una confirmación dentro de un tiempo limitado.
- QoS 2: en el presente servicio es donde cliente y servidor de aseguran que llegue el mensaje por medio de un enlace de 4 pasos. (Paessler, 2022).

Métricas de desempeño

Las métricas de desempeño están enfocadas al QoS en la transmisión y recepción de datos (Zapata Rodríguez, 2018) y estos son:

Retardo de transferencia de paquetes: Es el tiempo de demora que tiene un paquete de datos al ser enviado de un punto a otro al cual se le denomina retardo, este parámetro depende del umero de nodos en la red, protocolos a utilizar entre otros como el tráfico de la red. Para sistemas en tiempo real se debe disminuir el retardo (Zapata Rodríguez, 2018).

$$\delta = t_p + t_{tx} + t_{cola} + t_{pr} \quad (1)$$

Jitter (Variabilidad del retardo): El retardo de un paquete varia a lo largo del camino entre el transmisor y receptor esta variación de tiempo se le llama Jitter. Este parámetro puede aumentar o disminuir dependiendo las turas que tome los paquetes para llegar a su destino ya que en una conexión existen varios nodos interconectados por el cual se presenta este tipo de retardos entre el retardo anterior respecto al posterior. (Zapata Rodríguez, 2018).

$$Jitter = \frac{\sum|retardo_i - retardo_{i-1}|}{n-1} \quad (2)$$

Paquetes perdidos: Porcentaje de paquetes descartados de un total que han sido transmitidos, estás pérdidas se pueden dar por diversos motivos como congestión en las colas de los nodos, tiempo de vida (TTL en IPv4) o (HL en IPv6) (Zapata Rodríguez, 2018).

$$P_p = P_e - P_r \quad (3)$$

Eficiencia: Es la velocidad real de transporte de datos a través de una red inalámbrica, representada por el número de bits que se transmiten en un período de tiempo como medida de eficiencia es el *Throughut*, (η_r) (Zapata Rodríguez, 2018).

De acuerdo a (Lara Cueva et al., 2016), la eficiencia de una red se consigue de acuerdo con la siguiente ecuación:

$$E_f = \frac{\eta_r}{RBR} \times 100 \quad (4)$$

Donde η_r corresponde al *throughput* y RBR es la tasa neta de transmisión.

Tabla 4.

Parámetros de calidad de servicio.

PARAMETRO	UNIDADES
Retardo	Ms
Jitter	Ms
Paquetes perdidos	Mb
Eficiencia	%

Nota, Esta tabla indica las unidades de los parámetros de calidad de servicio.

Módulo ESP32 CAM

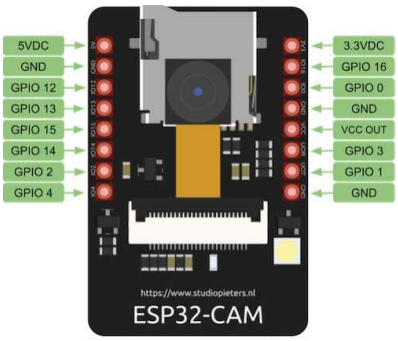
ESP32 CAM, es un dispositivo que tiene conectividad Wifi y Bluetooth, pines GPIO, aparte se le han añadido dos opciones más. Lleva integrado una pequeña cámara de video y una conexión para una tarjeta MicroSD, donde se puede almacenar fotos o videos, la bondad de este controlador es que puede realizar *streaming* tanto de video como de imágenes por medio de un servidor local (Naylamp, 2021).

Las aplicaciones que se puede realizar con el módulo ESP32 CAM un control de cámara para la transmisión de imágenes y procesadas por un robot o un circuito de video vigilancia, reconocimiento facial entre otros (Naylamp, 2021).

No es muy complicado la programación de este módulo solo se necesita un conversor USB externos e instalar las librerías necesarias en el IDE de Arduino (Naylamp, 2021).

Figura 9.

Módulo ESP32 CAM



Nota, es la estructura de un módulo ESP32-CAM como parte del marco teórico, consultado en la página de mecatrónica, 2021.

Capítulo III: Implementación del sistema inalámbrico y protocolo MQTT

Máquina virtual

Para la implementación de un servidor MQTT, como puente para la comunicación inalámbrica entre tarjetas ESP32 basándose en la Internet se tienen los siguientes proveedores que disponen de máquinas virtuales en modo gratuito por tiempo limitado:

- Google Cloud
- AWS
- Hosting Cloud
- Ardilu
- Nube Azufre

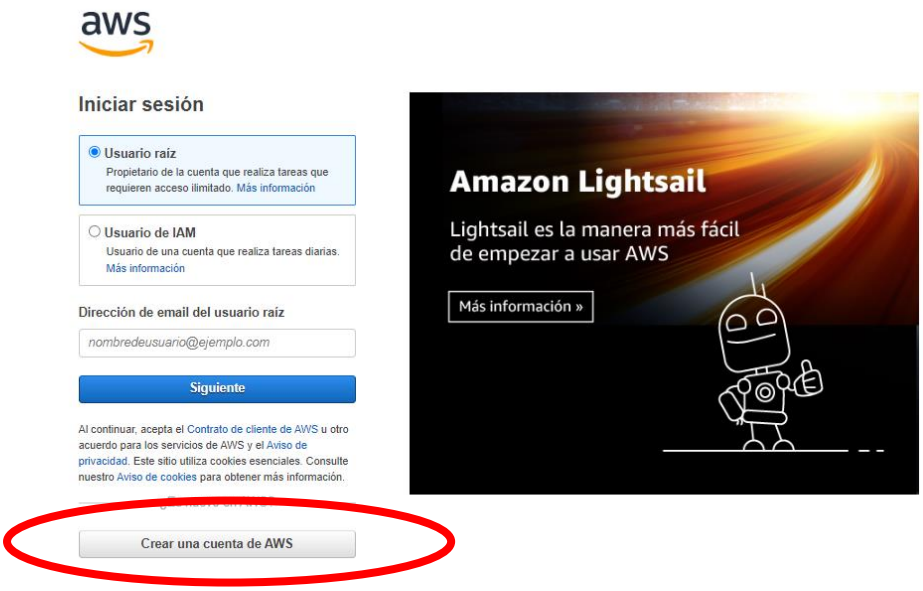
Para el presente proyecto y con fines educativos se elige AWS que ofrece un año de suscripción gratuita, 750 horas mensuales de uso y 5 GB de espacio mensual. Permite seleccionar el sistema operativo, el lenguaje de programación, la plataforma de aplicaciones web, la base de datos, así como el resto de los servicios que se necesite.

Para comenzar seguiremos los siguientes pasos para la implementación de una máquina virtual

PASO 1: Se debe crear una cuenta en AWS ingresando a la página oficial.

Figura 10.

Página Oficial de AWS

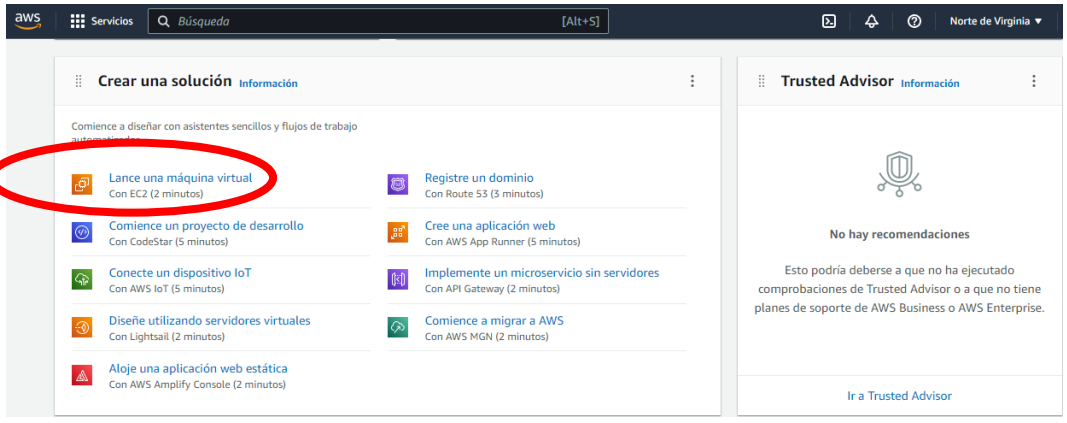


Nota, La imagen muestra la página oficial de AWS y la pestaña para crear una cuenta.

PASO 2: Una vez creada y activada la cuenta en AWS, ingrese a su cuenta con usuario raíz y se procede a lanzar una máquina virtual como se muestra en Figura 11.

Figura 11.

Crear una máquina virtual en AWS

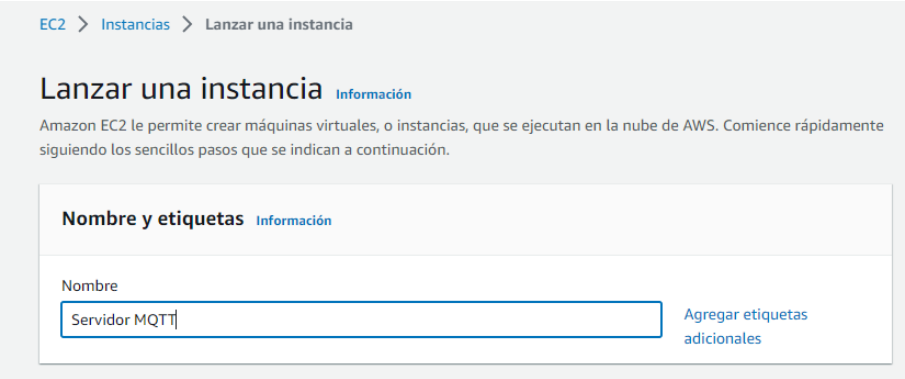


Nota, La imagen muestra cómo crear una instancia virtual.

PASO 3: Se configura la máquina virtual y el sistema operativo que se va a utilizar, para el presente proyecto el sistema operativo que se va a utilizar es UBUNTU en su versión más reviente, la configuración de la máquina virtual se presenta en las siguientes ilustraciones.

Figura 12.

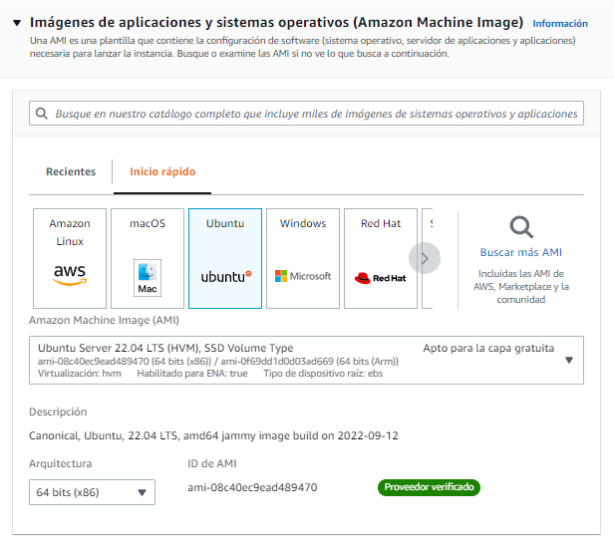
Nombre de la instancia de la máquina virtual



Nota, En la imagen muestra los campos a llenar para crear la instancia virtual.

Figura 13.

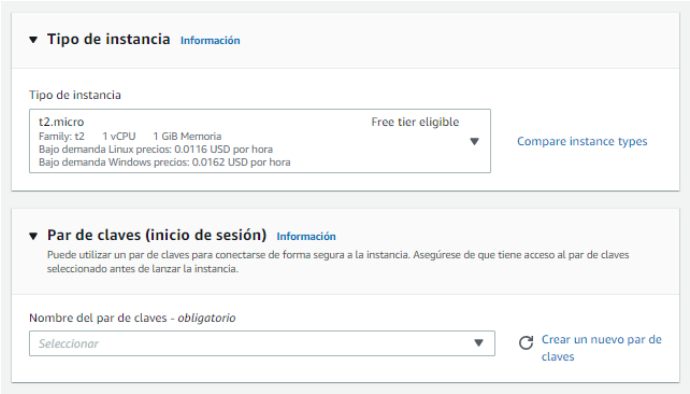
Sistema operativo.



Nota, En la imagen se muestra varios sistemas operativos a seleccionar para el proyecto se selecciona la distribución de UBUNTU.

Figura 14.

Parámetros de instancias.

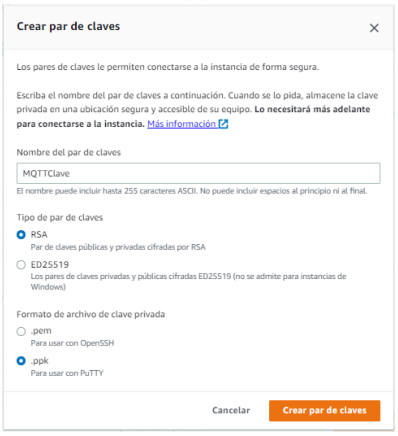


Nota, La imagen muestra los parámetros que tendrá la instancia creada.

Para el ingreso por protocolo SSH, en este punto se crea un nuevo Par de claves lo cual permitirá el acceso por el software PuTTY, en el proceso se crea y se descarga automáticamente un archivo con extensión **.ppk** como se muestra en Figura 15, lo cual se guarda para el siguiente proceso.

Figura 15.

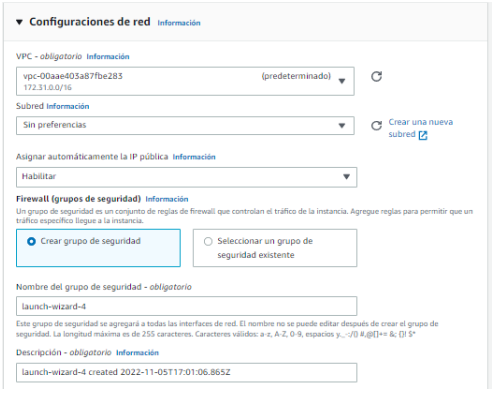
Creación de par de claves.



Nota, En la imagen se crea claves para poder ingresar a la instancia por otro software de comunicación.

Figura 16.

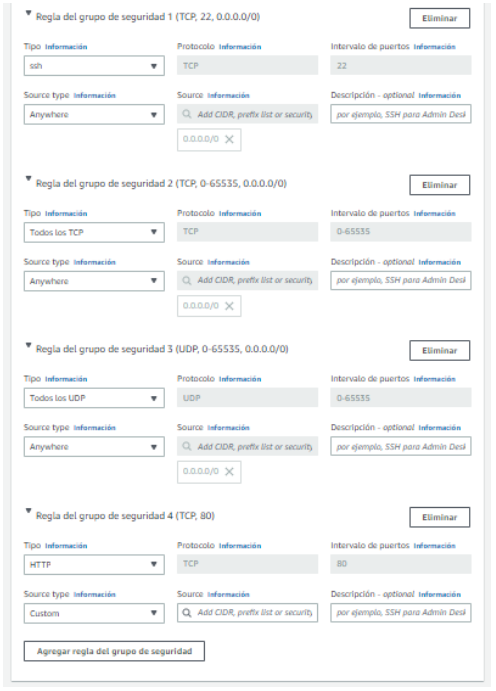
Protocolos de red.



Nota, En la imagen muestra los protocolos de comunicación que presenta la instancia.

Figura 17.

Activación de puertos para protocolo MQTT.



Nota, En esta imagen muestra como activar los protocolos de comunicación MQTT.

Figura 18.

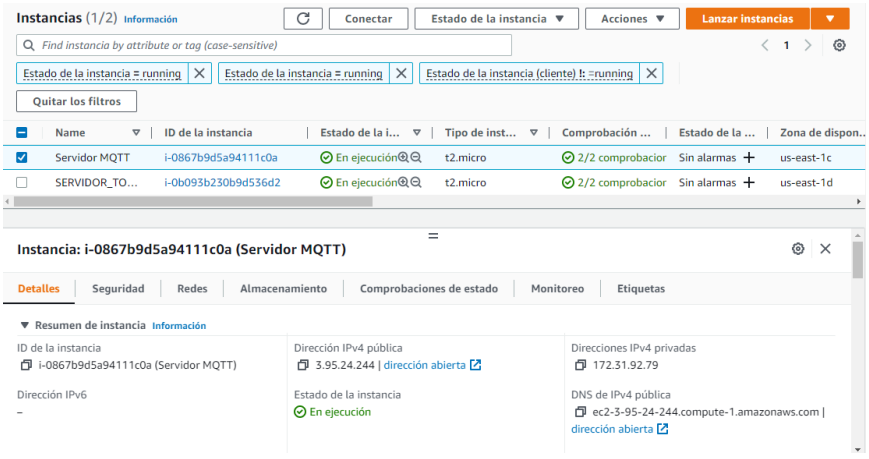
Almacenamiento del sistema.

Nota, En esta imagen muestra el tamaño de almacenamiento que tiene el disco duro de la instancia creada.

Una vez terminado las configuraciones principales de la máquina virtual al final de la página de configuración permite lanzar la instancia y estaría creada la máquina virtual. Para la verificación y visualización de las instancias creadas, se va a la pestaña INTANCIAS y verificamos las máquinas virtuales creadas como se muestra en Figura 19.

Figura 19.

Instancias en AWS.



Nota, Esta imagen muestra una lista de instancias creadas.

Algo muy importante a notar es que AWS asigna una dirección IP publica a la máquina virtual la cual sirve como puente más adelante para la comunicación entre módulos ESP32.

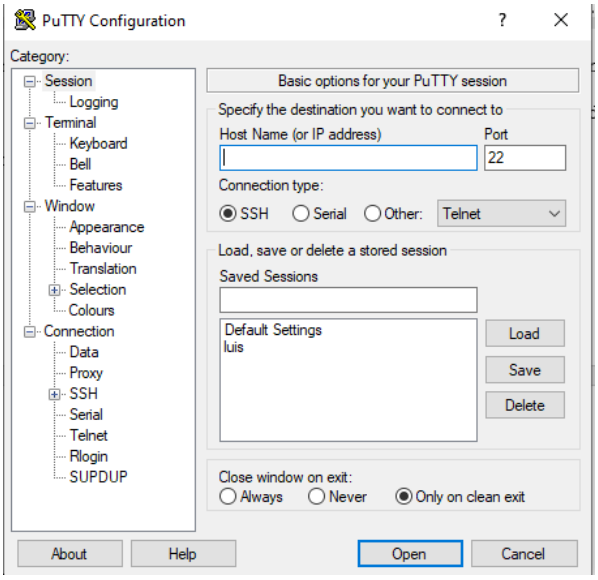
Implementación del protocolo MQTT

Para la implementación de nuestro servidor MQTT en nuestra máquina virtual se debe ingresar a la máquina virtual por un emulador que admita el protocolo SSH para fines educativos del proyecto se lo realiza en PuTTY, el cual es un emulador gratuito y se lo puede descargar e instalar fácilmente.

Se ingresa al emulador PuTTY.

Figura 20.

Emulador PuTTY.

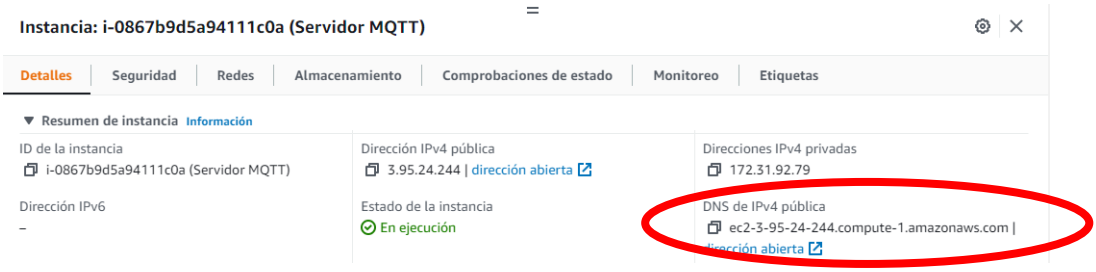


Nota, La imagen muestra el software de configuración de PuTTY.

Copiamos y pegamos la DNS IPv4 publica de la instancia creada.

Figura 21.

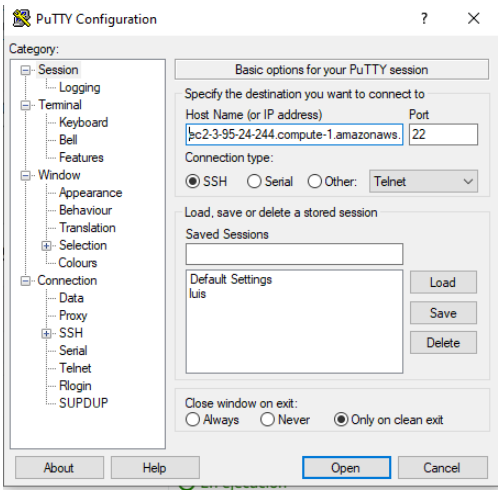
DNS IPv4 publica.



Nota, La imagen muestra los detalles de la instancia creada como direcciones IP y DNS.

Figura 22.

IPv4 en emulador PuTTY.

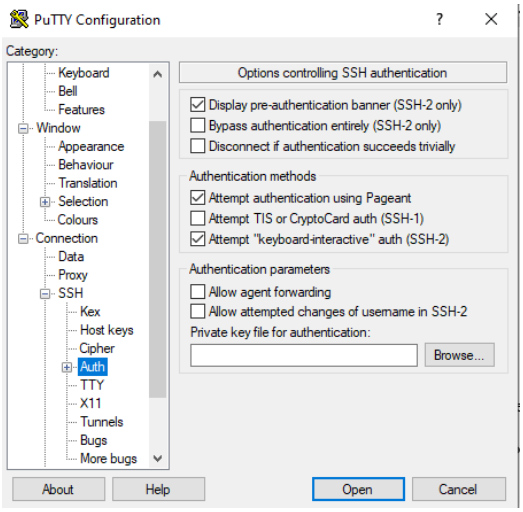


Nota, La imagen muestra como iniciar la instancia desde el software PuTTY.

Seguido de este proceso en *Category* se dirige a la pestaña SSH donde desplegamos el submenú y seguido dar click en **“*Auth*”** donde en ***Browser*** se busca el archivo “.ppk” generado al crear las 2 claves.

Figura 23.

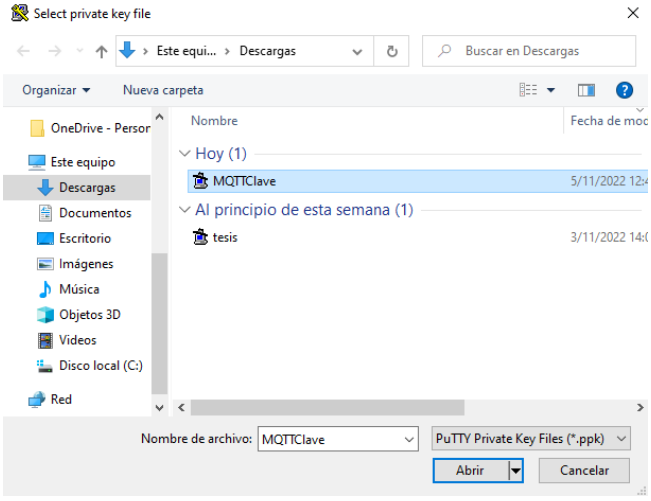
Pestaña Auth.



Nota, La imagen muestra las configuraciones realizadas para iniciar la instancia creada.

Figura 24.

Archivo clave de acceso.

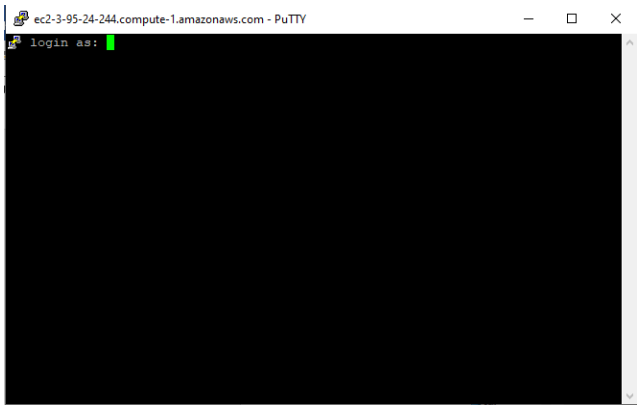


Nota, La imagen muestra la clave de seguridad creada para iniciar la instancia.

Una vez realizado este proceso se da click en *open* y se abrirá la instancia.

Figura 25.

Plataforma de instancia iniciando.

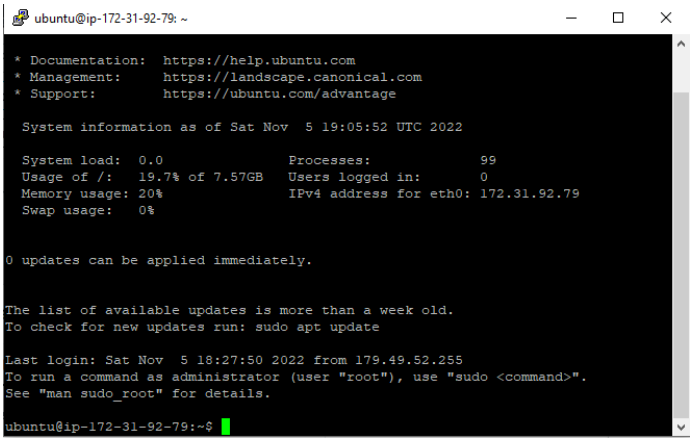


Nota, La imagen muestra la instancia iniciada.

Para ingresar al sistema operativo escribimos la palabra "*ubuntu*" y se presiona *enter*, lo cual automáticamente abre la máquina virtual donde se empieza la instalación del protocolo MQTT.

Figura 26.

Máquina Virtual Abierta.



Nota, La imagen muestra la instancia ya iniciada con la distribución de Ubuntu instalada.

Se descarga la *signin key* o clave de firma utilizando el comando *wget*, este comando descarga el fichero indicado.

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
```

Figura 27.

Descarga de clave de firma.

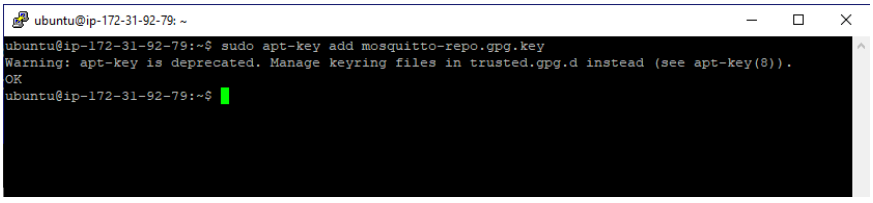
Nota, La imagen muestra la instalación del protocolo MQTT.

Se añade la clave para autenticar el paquete a descargar

```
sudo apt-key add mosquitto-repo.gpg.key
```

Figura 28.

Clave de autenticidad.



Nota, La imagen muestra la instalación del protocolo MQTT.

Se debe dirigir a la siguiente carpeta con el mandato cd.

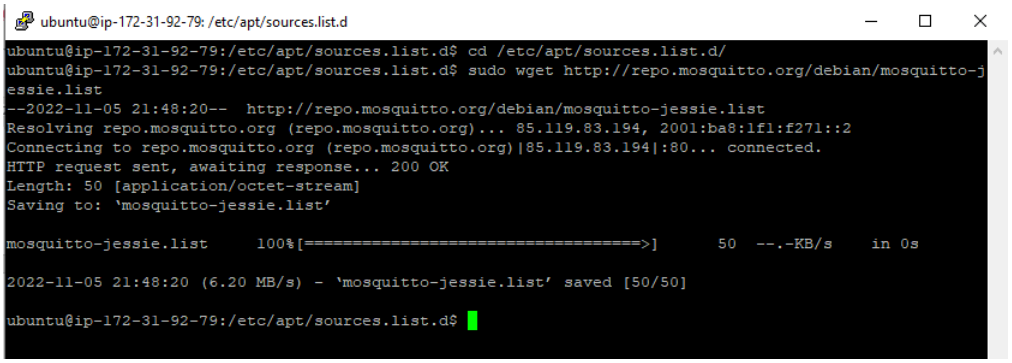
```
cd /etc/apt/sources.list.d/
```

Descargamos en el directorio establecido la lista de repositorios de Mosquito con *wget*

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

Figura 29.

Descarga de directorios.



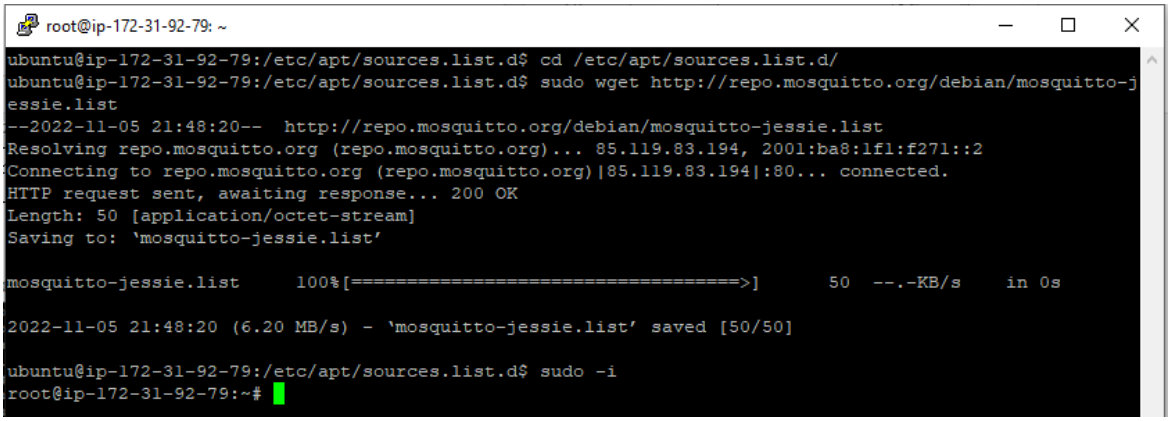
Nota, La imagen muestra la instalación del protocolo MQTT.

Ingresamos a super usuario

```
sudo -i
```

Figura 30.

Super Usuario.



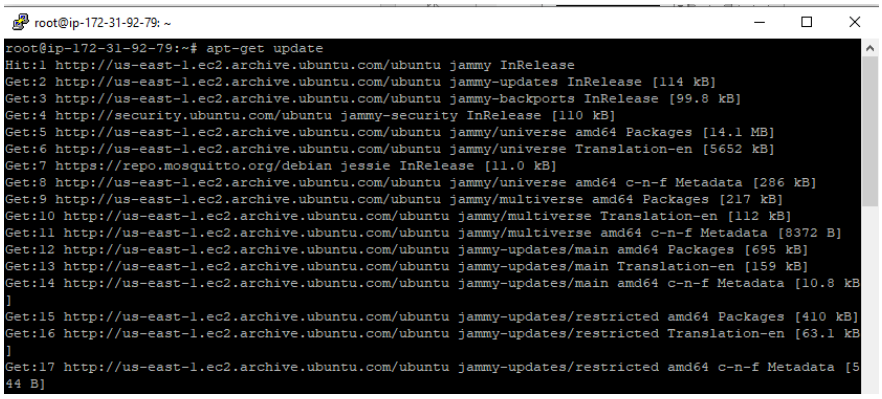
Nota, La imagen muestra la instalación del protocolo MQTT.

Actualizamos paquetes y versiones de Ubuntu

```
apt-get update
```

Figura 31.

Actualización del sistema.



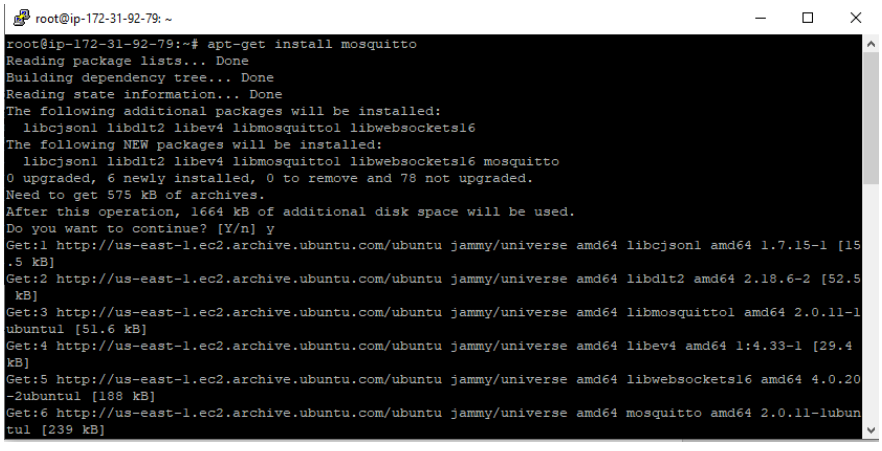
Nota, La imagen muestra como se actualiza el sistema operativo.

Instalamos Broker de Mosquito

```
apt-get install mosquitto
```

Figura 32.

Instalación de MQTT.



Nota, La imagen muestra la instalación del broker para el protocolo MQTT.

Para poder acceder al Broker MQTT se debe editar el archivo mosquitto.conf

```
sudo nano etc/mosquitto/mosquitto.conf
```

```
listener 1883

# Indica si se permite la conexión de cliente sin usuario y contraseña (true) o debe indicar usuario y contraseña (false)
allow_anonymous true
# Indica si almacenar los mensajes en la base de datos
persistence true

# Indica el directorio donde se almacenan los datos
persistence_location /mosquitto/data/

# Indica donde almacenar los logs de Mosquitto
log_dest file /mosquitto/log/mosquitto.log
```

Ahora instalamos el cliente MQTT

```
apt-get install mosquitto-clients
```

Realizada esta última operación podremos ya probar la conexión MQTT abriendo 2 terminales en PuTTY y en cada una respectivamente enviar los siguientes comandos.

En el servidor:

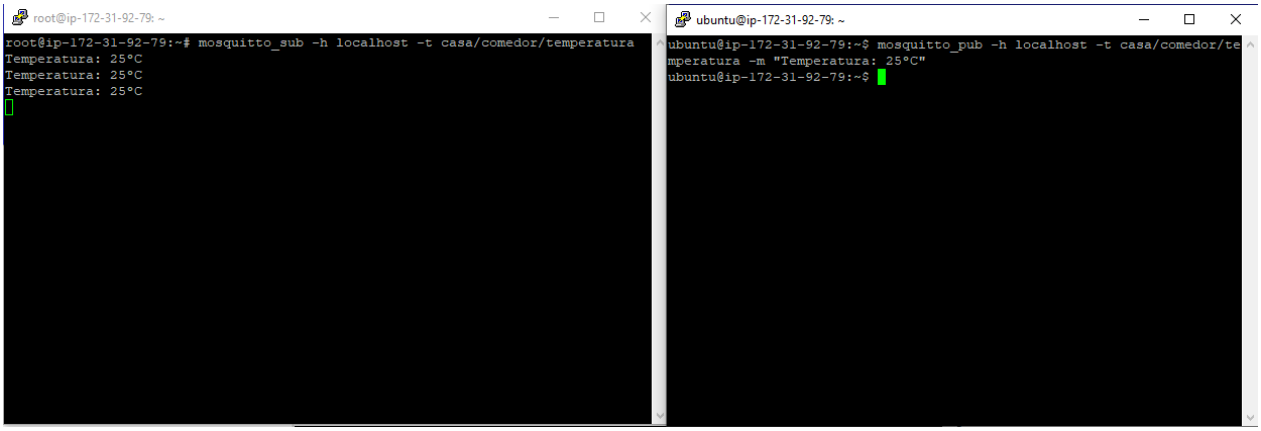
```
mosquitto_sub -h localhost -t casa/comedor/temperatura
```

En el Cliente

```
mosquitto_pub -h localhost -t casa/comedor/temperatura -m "Temperatura: 25ºC"
```

Figura 33.

Envió y recepción de mensajes MQTT.



Nota, La imagen muestra la realización de pruebas de envío y recepción de mensajes por medio del protocolo MQTT.

Comando mosquitto_sub: Este comando permite suscribirnos a un topic escuchando y mostrarlo por pantalla, los parámetros son los siguientes:

- -h: indica lo que viene después del host, el host se refiere a la dirección IP
- BROKER: dirección IP del servidor
- -t: indica que lo que viene después es el topic a suscribirse
- TOPIC: nombre del topic al que vamos a suscribir.

```
mosquitto_sub -h BROKER -t TOPIC
```

Comando mosquitto_pub: Con este comando podemos publicar mensajes muy simples, los parámetros son los siguientes:

- -h: indica lo que viene después del host, el host se refiere a la dirección IP
- BROKER: dirección IP del servidor
- -t: indica que lo que viene después es el topic a suscribirse
- TOPIC: nombre del topic al que vamos a suscribir.
- -m: indica que lo que viene es el mensaje.
- MENSAJE: el mensaje que deseamos enviar. Tiene que ir en comillas dobles, ejemplo “Hola”

```
mosquitto_pub -h BROKER -t TOPIC -m MENSAJE
```

Implementación del transmisor con el módulo ESP32

Una vez implementado el servidor MQTT en la máquina virtual el siguiente paso a proseguir es el algoritmo para el módulo ESP32 el cual pueda conectarse con el protocolo MQTT en el envío de datos como transmisor de señales.

Para poder obtener las señales que irán conectadas al módulo ESP32 se utilizaron potenciómetros y 2 *flex sensor* que ingresan a las entradas analógicas del ESP32.

La distribución de estos *flex sensor* es ubicada en un guante antiestático lo cual no genera señal electrostática que pueda dañar al sistema, los 2 *flex sensor* son colocados: el primero en el dedo índice simulando el movimiento de la mano para agarrar cosas y el segundo sensor en la parte superior de la muñeca el cual simula el movimiento hacia arriba y abajo, en la Figura 34 se ilustra la posición de estos sensores.

Figura 34.

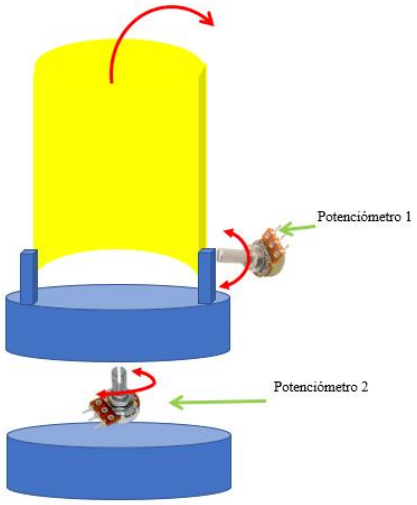
Ubicación de los flex sensor.

Nota, La imagen muestra la ubicación de cada uno de los sensores que van a simular movimiento.

Para simular el movimiento del antebrazo y el brazo se utilizó un sistema mecánico impreso en 3D que se ajuste a las necesidades del proyecto, en el cual esta implementado 2 potenciómetros para el giro del brazo y elevación del antebrazo como se muestra en la Figura *35*.

Figura 35.

Sistema Mecánico para brazo y antebrazo.



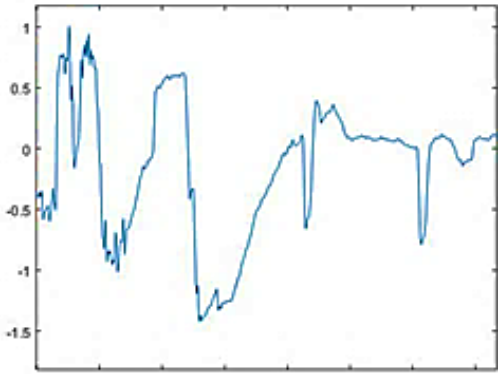
Nota, La imagen muestra el diseño en 3D, mecánico y posición de los potenciómetros que actuaran para la movilidad de articulaciones.

Todas estas señales ingresan al módulo ESP32 las cuales serán procesadas y enviadas por la señal WiFi al protocolo MQTT el cual envía hacia el otro modulo ESP32 por la señal WiFi.

Los *flex sensor* son muy sensibles, lo cual genera mucho ruido y envía falsas ordenes como se ilustra en la Figura *35*

Figura 36.

Flex sensor sin filtro.

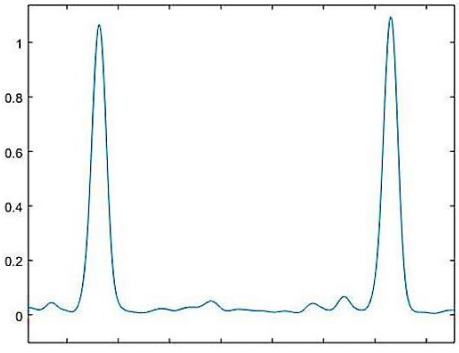


Nota, La imagen muestra la señal muestreada de los sensores flex que presentan mucho ruido.

Para contrarrestar este ruido y falsas órdenes en la salida de la señal del sensor se aplica un filtro con capacitores, el cual suaviza la señal y ya no presenta picos falsos como se ilustra en Figura 37.

Figura 37.

flex sensor con filtro



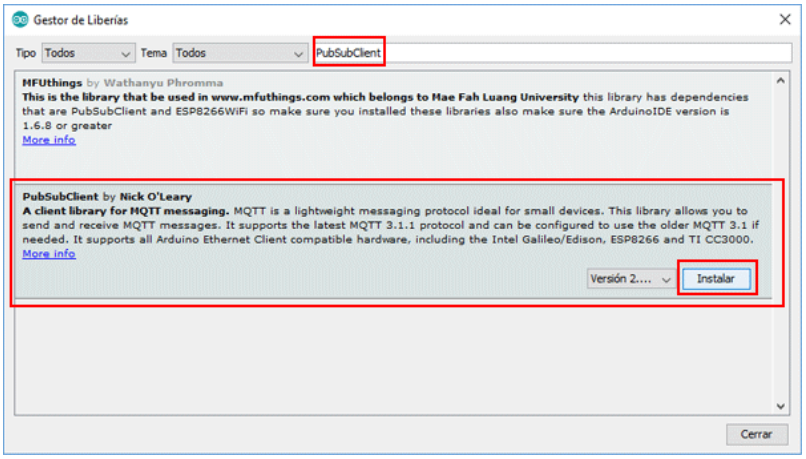
Nota, La imagen muestra la señal muestreada de los sensores flex con un filtro ante el ruido.

Una vez filtrada las señales de cada uno de los *flex sensor* y potenciómetros se procede al algoritmo.

Se deben tener instaladas las librerías en el IDE de Arduino, para el protocolo MQTT se debe instalar la librería *"PubSubClient"*, en el IDE de Arduino debemos ir al menú Programa>Incluir Librería> Gestor de librerías y buscar la librería mencionada como se muestra en la Figura 38.

Figura 38.

Instalación de librería PubSubClient



Nota, La imagen muestra como instalar las librerías del protocolo MQTT para Arduino.

En el siguiente fragmento de código empezamos declarando las librerías a utilizar en este caso *WiFi.h* que viene por defecto en la tarjeta instalada y la librería *PubSubClient.h* el cual sirve para Publicar o Subscribir datos en un tópico, se debe asignar variables de tipo chart para designar el nombre de la red WiFi y su contraseña, también es importante tener asignada a una variable la dirección IP publica de la máquina virtual antes creada la cual encontramos en las instancias creadas como se muestra en Figura 39.

Figura 39.

IPv4 Publica

Nota, La imagen muestra la dirección IP publica para la programación en Arduino.

```
#include <WiFi.h>
#include <PubSubClient.h>
consta char* ssid = "TESIS";                    //Nombre de la red a conectarse
const char* password = "tesis1719";             //Contraseña de la red
const char* mqtt_server = "3.90.220.149";       //IP pública del servidor
unsigned long tiempo1 = 0;                      //Variables independientes
unsigned long tiempo2 = 0;
unsigned long total = 0;
```

En la siguiente parte del código se realiza la conexión a la red WiFi, donde nos asignan una dirección IP disponible.

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Para la conexión con el protocolo MQTT y la declaración de cada Tópico, tenemos la siguiente fracción de código, en el cual es muy importante asignar a cada cliente MQTT un nombre diferente ya que el sistema inalámbrico no funcionaría y se tiende a llenar de datos basura.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("USUARIO1")) {  // Tener en cuenta que cada cliente debe tener un nombre diferente
      Serial.println("connected");
      client.subscribe("robot/mecanico/retardo1");
      client.subscribe("robot/mecanico/brazo");
      client.subscribe("robot/mecanico/antebrazo");
      client.subscribe("robot/mecanico/muneca");
      client.subscribe("robot/mecanico/mano");
      client.subscribe("robot/mecanico/retardo2");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Para medir y hacer un análisis futuro de los retardos en la transmisión y recepción de datos se crea los tópicos "robot/mecánico/retardo", los cuales miden el Jitter del sistema inalámbrico.

Para finalizar el análisis del algoritmo del Transmisor, la señal analógica de los sensores que ingresa a un pin ADC conversor analógico-digital se escalona para el ángulo de giro de un servomotor o ancho de pulso PWM que hace que gire a una cierta posición el servomotor, este dato una vez procesado se transforma en el Mensaje a enviar por medio del tópico del protocolo MQTT.

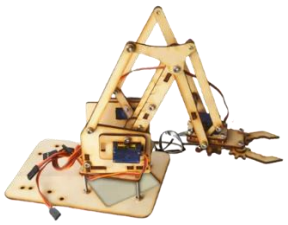
```
    int s1 = analogRead(34);                //leemos el valor del potenciómetro (de 0 a 1023)
 pin[0]= map(s1, 0, 1750, 330, 750);    // escalamos la lectura a un valor de ángulo (entre 0 y 180)
  if(v1>pin[0]+19 || v1<pin[0]-19)
  {
    brazo=pin[0];
    snprintf(msg, 75, "%ld", brazo);
    client.publish("robot/mecanico/brazo", msg); //Publica el mensaje a la red.
    v1=pin[0];

    delay(60);
          }
```

Implementación del receptor con el módulo ESP32

En el receptor tenemos un brazo robótico manejado mecánicamente con servo motores el cual tiene 4 grados de libertad el cual está conectado al módulo PCA9685 el cual maneja 16 puertos ara 16 servomotores, es alimentado independientemente por una batería de 5 V y 10 A para el manejo de potencia de los servomotores, esto se lo puede apreciar en la Figura *40* .

Figura 40.

Brazo robótico



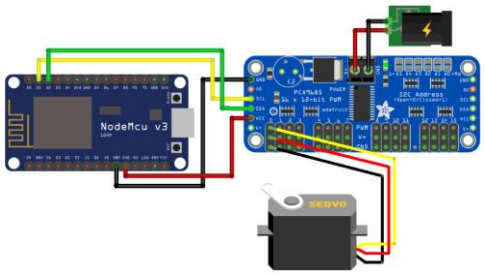
Nota, tomado de la página web de robotic,2022.

El módulo PCA9685 se conecta por I2C al módulo ESP32 el cual envía el ancho de pulso para el movimiento del servo motor, lo cual es ilustrado en la

Figura ***41***.

Figura 41.

Conexión de ESP32 y PCA9685.

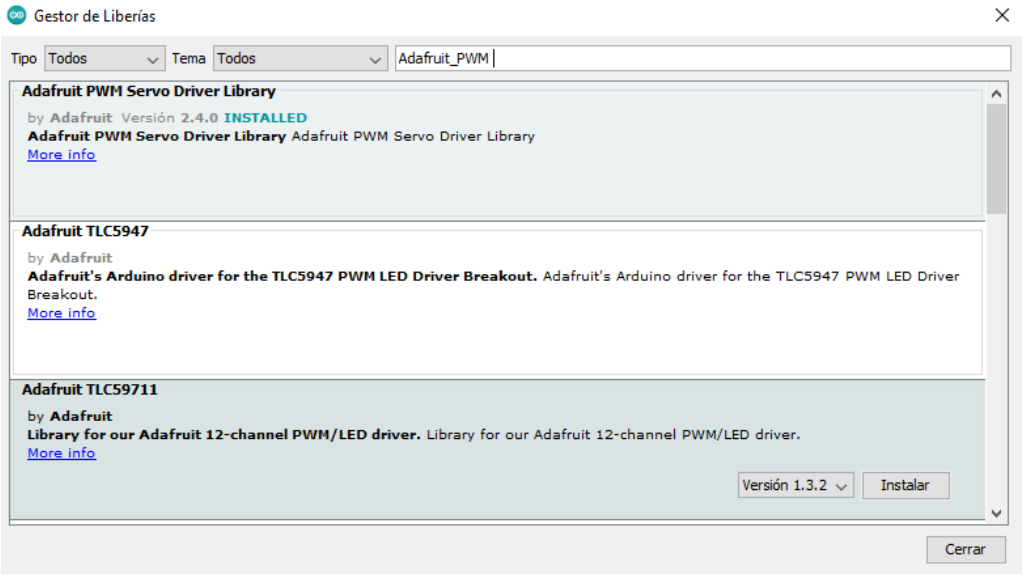


Nota, tomado de la página web de robotic,2022.

Para el control de los servomotores con el módulo PCA9685 se debe instalar la librería *Adafruit PWM Servo Driver Library,* en el IDE de Arduino debemos ir al menú Programa>Incluir Librería> Gestor de librerías y buscar la librería ya mencionada como se muestra en la Figura 42.

Figura 42.

Instalación de librerías para servo motores



Nota, La imagen muestra como instalar las librerías de PWM para Arduino.

En el siguiente fragmento de código empezamos declarando las librerías a utilizar en este caso *WiFi.h* que viene por defecto en la tarjeta instalada y la librería *Adafruit_PWMServoDriver.h* con el cual se envía datos para el manejo de los servomotores.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver srituhobby = Adafruit_PWMServoDriver();

#define servo1 0
#define servo2 1
#define servo3 2
#define servo4 3
int pin[]={0,0,0,0};
char msg[50];
const char* ssid = "CELERITY_TOPON";
const char* password = "luis1719690164";
const char* mqtt_server = "3.90.220.149";
```

También se declaran las variables estáticas con el nombre de la red Inalámbrica, contraseña y la dirección IPv4 pública del servidor para recibir los datos del Módulo Transmisor ESP32 por medio del servidor virtual.

La siguiente parte del código es inicializar los servomotores en una posición donde la mano robótica tenga una referencia.

```
void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  srituhobby.begin();
  srituhobby.setPWMFreq(60);
  srituhobby.setPWM(servo1, 0, 330);
  srituhobby.setPWM(servo2, 0, 500);
  srituhobby.setPWM(servo3, 0, 500);
  srituhobby.setPWM(servo4, 0, 210);
  delay(1000);
  client.setCallback(callback);
}
```

En la última parte explicativa del código, el mensaje recibido es una cadena de caracteres *String* lo cual debemos transformarlo a entero ya que la librería del servomotor solo acepta valores enteros para poder girar el servo motor al ángulo o posición designada.

```
void callback(char* topic, byte* payload, unsigned int length) {
 String val;

  for (int i = 0; i < length; i++) {
    val += (char)payload[i];              //Transformando en una cadena de caracteres
  }

  int entero=String(val).toInt();        //Transformar a un numero entero

 if (String(topic) == "robot/mecanico/brazo") {
  pin[0]=entero;
  srituhobby.setPWM(servo1, 0, pin[0]); //Movilizar el servomotor
  }
```

Implementación para determinar el retardo en la comunicación inalámbrica

Para la determinar el retardo del sistema se utiliza la función *millis()* en el algoritmo en el transmisor y receptor:

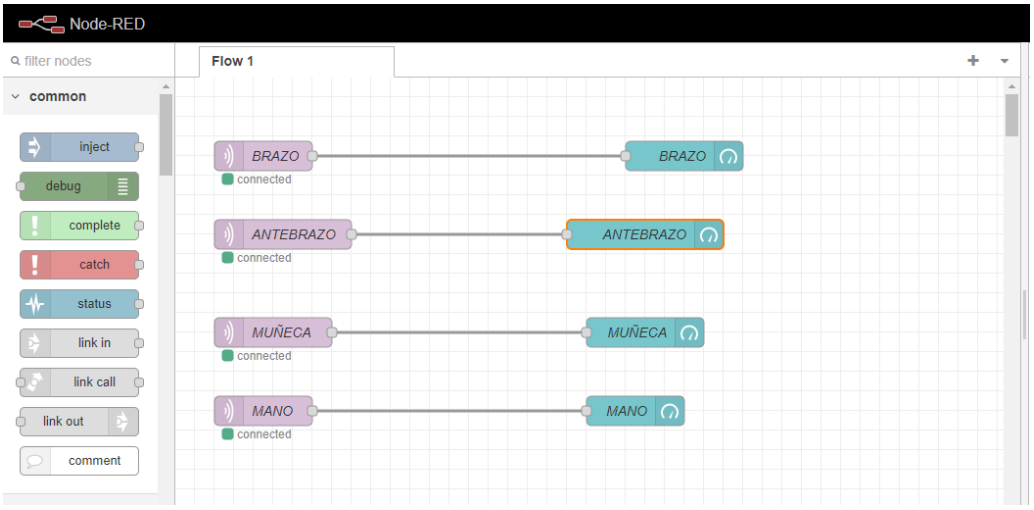
```
//CODIGO DEL TRANSMISOR
  tiempo1=millis();

  int s1 = analogRead(34);//leemos el valor del potenciómetro (de 0 a 1023)
 pin[0]= map(s1, 0, 1750, 330, 750);  // escalamos la lectura a un valor de ángulo (entre 0 y
180)
  if(v1>pin[0]+19 || v1<pin[0]-19)
  {
    brazo=pin[0];
    snprintf(msg, 75, "%ld", brazo);
    client.publish("robot/mecanico/brazo", msg);
    v1=pin[0];

    delay(60); //Tiempo de encerado de datos
  }
```

La función *millis()* es un contador en ms el cual determina el tiempo que se demora una instrucción o serie de instrucciones, por medio de esta función se determina el retardo en la comunicación Punto a Punto dejando a un lado el retardo en el servomotor o retardos en la señal de los sensores, se debe restar el tiempo que toma el Módulo ESP32 en procesar los datos para obtener un valor aproximado del retardo en la comunicación inalámbrica, esta función se aplica a cada servomotor.

Para una mejor visualización de los retardos de cada parte del servomotor se utiliza la plataforma NODE RED la cual es una interfaz gráfica para la muestra de datos. En las siguientes ilustraciones de la Figura 43 y Figura 44 se muestra su diagrama de bloques e interfaz gráfica.

Figura 43.

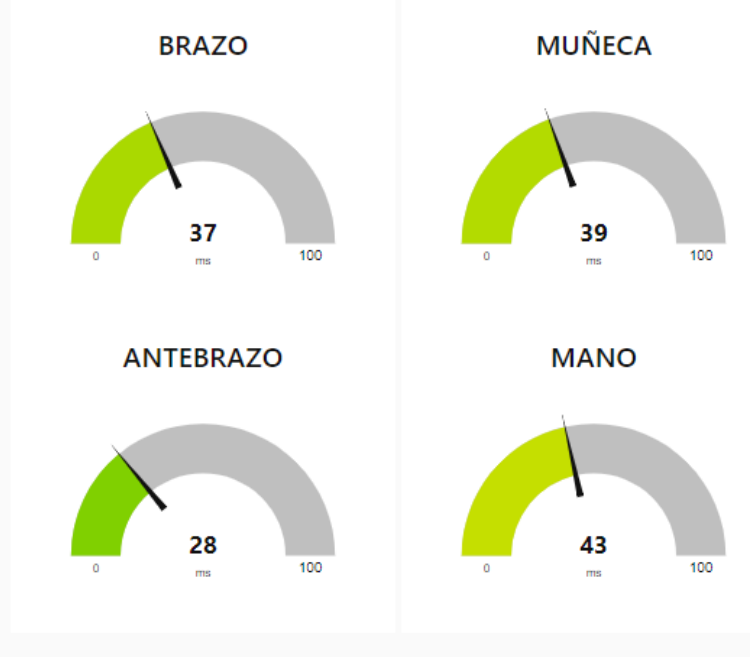
Diagrama de bloques NODE RED



Nota, La imagen muestra la programación en Node Red para la interfaz gráfica.

Figura 44.

Interfaz gráfica NODE RED



Nota, La imagen muestra la interfaz gráfica y mediciones del retardo de cada articulación de un brazo.

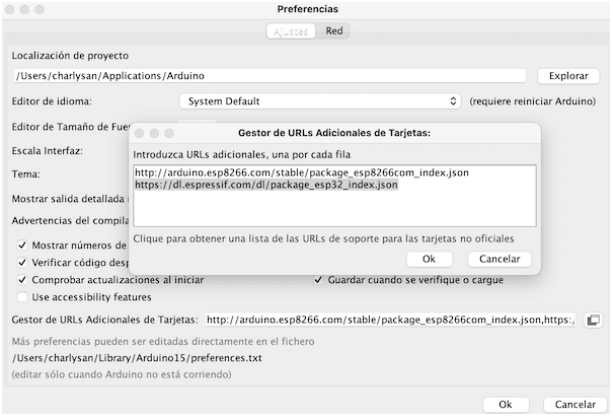
Implementación del servidor de video

Para la implementación de la cámara para la visualización del brazo robótico se debe instalar en el IDE de Arduino las librerías necesarias, si la opción no aparece, ESP32 Arduino puede suceder que no esté instalada esta tarjeta en el IDE de Arduino. Debes entrar a las preferencias del IDE Arduino. Luego donde indica Gestor de URLs adicionales de tarjetas debes ingresar el siguiente enlace, como se ilustra en la Figura 45.

https://dl.espressif.com/dl/package_esp32_index.json

Figura 45.

Instalación de ESP32 CAM en IDE de Arduino

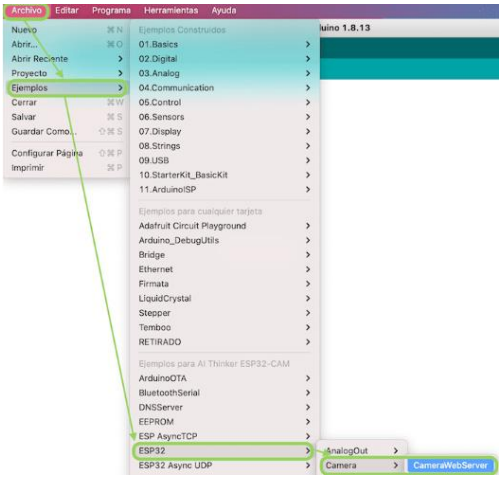


Nota, La imagen muestra como instalar las librerías para Arduino para ESP32 CAM.

Seguidamente de esto ya podremos cargar el ejemplo de servidor de cámara web, este se encuentra en la dirección Archivo/Ejemplos/ESP32 como se muestra en la Figura 46.

Figura 46.

Abrir el ejemplo para cama web



Nota, La imagen muestra la manera de como programar la Cámara de un ESP32 y sus ejemplos.

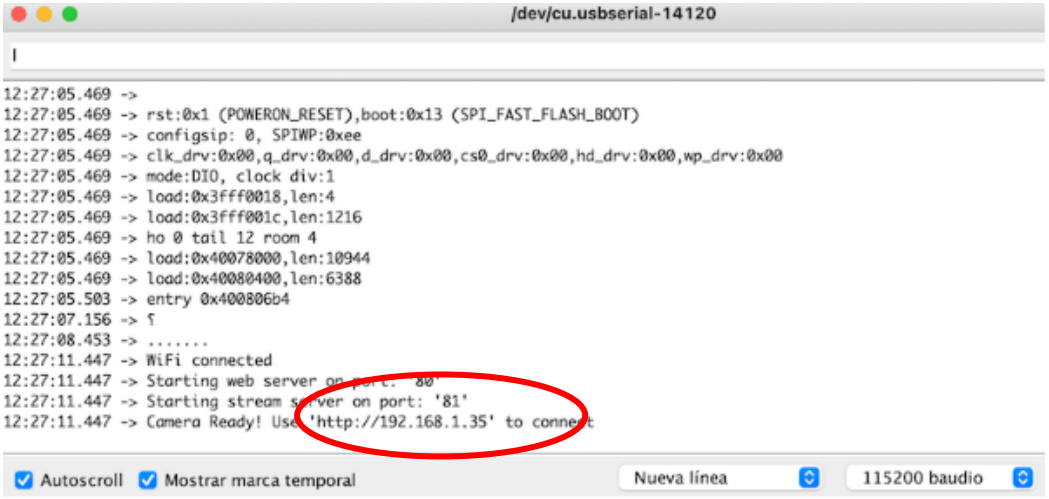
Una vez abierto el algoritmo solo se debe hacer 2 cambios, en el modelo de cámara y usuario con contraseña del servidor web.

```
//Replace with your network credentials
const char* ssid = "CELERITY_TOPON";
const char* password = "luis1719690164";
#define CAMERA_MODEL_AI_THINKER
```

Con esos cambios en el código, se lo vuelve a cargar a la tarjeta y se activa el serial de Arduino para identificar la dirección IP asignada hacia la cámara para poderle enlazar a un servidor de video como se muestra en la Figura 47.

Figura 47.

IP Asignada al módulo ESP32 CAM



Nota, La imagen muestra la dirección web para la visualización de la cámara.

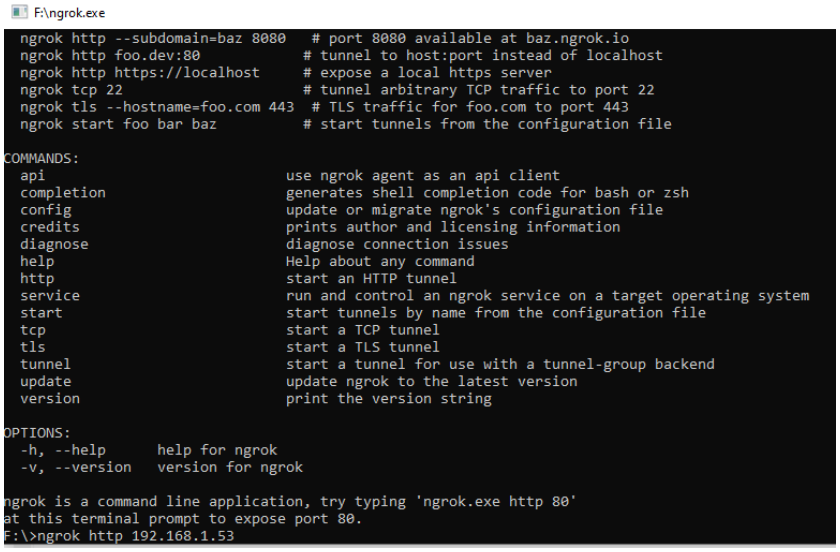
Una vez identificado la dirección IP asignada descargamos el software NGROK el cual sirve para enlazar a una dirección electrónica para que se pueda visualizar la cámara desde cualquier dispositivo que esté conectado a Internet u obtenga el *link*.

Abrimos el programa NGROK y escribimos la siguiente instrucción y damos *enter* como se muestra en la Figura 48.

```
ngrok http 192.168.1.35
```

Figura 48.

Software NGROK



Nota, La imagen muestra el software NGROK y su manera de redireccionar una dirección IP.

Una vez dado *enter* se asigna una dirección, la cual copiamos en el navegador y se enlaza desde cualquier parte por medio de Internet como se ilustra en la Figura 49.

Figura 49.

Cámara visualización

Nota, La imagen muestra cómo se visualiza la video cámara en tiempo real.

Capítulo IV: Análisis de resultados

Para el análisis del sistema inalámbrico en tiempo real basándonos en las métricas de desempeño se consideran para el presente proyecto el retardo y el Jitter del sistema para cálculos la ecuación (1) y (2), con lo cual se identifica la desviación estándar, varianza y por medio de una regresión lineal se calcula el error cuadrático medio del tiempo de retardo, esto se lo hace por el tipo de sistema diseñado el cual da datos en función del tiempo y esto sirve para futuras aplicaciones del brazo robótico u robot el cual debe emular los movimientos humanos sin retardos en la red lo cual es una característica de la tele robótica.

Una vez definidas la métricas, en el sistema de brazo robótico se realizaron las siguientes pruebas, con 3 proveedores diferentes de Internet, A cada proveedor se enlazaron hasta 10 dispositivos a la red (celulares, computadores, Smart TV) y nuestro sistema de brazo robótico, después se realizaron pruebas sin conectar ningún dispositivo a la red , solo nuestro sistema de brazo robótico con el fin de emular tráfico en la red y como última prueba con un solo proveedor se realizaron pruebas de distancia de 0 a 100 m en línea recta sin obstáculos esto se debe a que se desea determinar el retardo producido por el enlace inalámbrico con condiciones climáticas normales, en futuros proyectos se puede realizar pruebas de distancia para sistemas en tiempo real en diferentes ambientes, las pruebas se resumen en la siguiente Tabla *5*.

Tabla 5.

Pruebas de comunicación de un brazo robótico.

PROVEEDOR	# DE DISPOSITIVOS	# DE DISPOSITIVOS	DISTANCIA
CELERITY	10	0	0-100 m
NETLIFE	10	0	2 m
CNT	10	0	2 m

Nota, Esta tabla muestra los parámetros para la toma de muestras del retardo en la comunicación inalámbrica.

Métricas de análisis

Para el cálculo del retardo en el presente proyecto se lo determina por medio de la función *millis()* en el algoritmo el cual nos da el tiempo de retardo en el envío de datos, esta función se la utiliza en el receptor y trasmisor para estimar el retardo del canal.

La varianza y desviación estándar de una muestra se definen bajo las siguientes ecuaciones:

$$VARIANZA: \qquad s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{5}$$

$$DESVIACIÓN\ ESTÁNDAR: \qquad s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{6}$$

Donde $\bar{x}$ representa el retardo medio, x_i es la muestra de cada retardo y *N* en número de muestras.

Para el análisis se calcula el error estimado, para lo cual se utiliza una progresión lineal, la cual nos proporciona una ecuación de la recta y por medio de sus distancias de cada muestra se calcula el error estimado aplicando la siguiente ecuación:

$$ERROR\ ESTIMADO: \qquad e = \sqrt{\frac{1}{N-2}\sum_{i=1}^{N}(Y_i - \bar{Y}_i)^2} \tag{7}$$

Donde $\bar{Y}$ representa el retardo medio respecto a la ecuación de la regresión lineal, Y_i es la muestra de cada retardo esperado y *N* en número de muestras

Esta métrica de análisis nos permite analizar qué tan disperso puede estar el retardo respecto a la media de los datos y de acuerdo a los resultados se pueden analizar el tiempo de respuesta del sistema y mejorarlo teniendo una menor dispersión de datos.

Para el cálculo del Jitter se lo hace con el valor absoluto de la diferencia entre el retardo anterior y el retardo presente consecutivamente ya que la variabilidad del Jitter nos va a permitir tener una mejor apreciación del sistema inalámbrico, para lo cual se realiza la siguiente ecuación:

$$JITTER: \qquad J_n = |r_n - r_{n-1}| \tag{8}$$

Resultados

Para este capítulo se presentan tres tablas de resultados, las pruebas son realizadas con tres proveedores de Internet para el caso: Celerity, Netlife y CNT con y sin dispositivos adicionales conectados a su red, a cada una se le toma 50 muestras para reducir el error estándar medio, ya que al realizar pruebas con 1000 muestras o superiores el retardo del sistema no varía a gran escala y para una mejor representación de datos y análisis se estableció en 50 muestras con respecto a datos con invasión y sin invasión de tráfico, y para finalizar se estableció en 50 muestras para pruebas de distancia porque al realizar pruebas de 0 a 20 m el retardo no variaba considerablemente y a partir de los 30 m se incrementaba el retardo por lo que también se estableció hacer pruebas en escala de 20 m hasta su máxima distancia de 100 m porque a partir de los 110 m se pierde el enlace de conexión inalámbrica con el sistema.

Tabla 6.

Resultados de los 3 proveedores con 0 dispositivos en su red.

PRUEBA	CELERITY		NETLIFE		CNT	
	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)
1	79		106		99	
2	98	19	85	21	140	41
3	89	9	97	12	141	1
4	94	5	112	15	147	6
5	98	4	111	1	145	2
6	88	10	100	11	104	41
7	81	7	100	0	102	2
8	95	14	104	4	93	9
9	95	0	114	10	98	5
10	78	17	93	21	87	11
11	87	9	91	2	117	30
12	84	3	106	15	80	37
13	75	9	109	3	115	35
14	71	4	87	22	135	20
15	95	24	99	12	92	43
16	96	1	114	15	95	3
17	72	24	93	21	104	9
18	76	4	110	17	112	8
19	99	23	101	9	127	15
20	89	10	81	20	138	11
21	78	11	104	23	100	38
22	79	1	115	11	113	13
23	95	16	103	12	105	8
24	78	17	81	22	146	41
25	86	8	98	17	146	0
26	70	16	83	15	93	53
27	93	23	92	9	110	17
28	73	20	119	27	92	18
29	71	2	90	29	136	44
30	76	5	100	10	132	4
31	75	1	83	17	90	42
32	85	10	89	6	132	42
33	72	13	80	9	90	42
34	98	26	119	39	147	57
35	86	12	93	26	97	50
36	78	8	101	8	90	7
37	72	6	93	8	149	59
38	80	8	81	12	90	59
39	77	3	120	39	106	16
40	70	7	84	36	112	6
41	74	4	119	35	101	11
42	89	15	96	23	117	16
43	79	10	89	7	128	11
44	80	1	106	17	93	35
45	87	7	83	23	89	4
46	94	7	84	1	87	2
47	78	16	87	3	90	3
48	94	16	88	1	134	44
49	78	16	81	7	112	22
50	85	7	84	3	80	32
PROMEDIO	83,38	10,37	97,16	14,82	111,56	22,96

Nota, Esta tabla muestra el resultado del retardo en una comunicación inalámbrica sin dispositivos en la red.

De acuerdo a los resultados obtenidos se calcula las siguientes graficas de progresión lineal:

Figura 50.

Progresión lineal de los 3 proveedores con 0 usuarios en su red

Nota, La imagen muestra los resultados gráficamente sin dispositivos en la red con su progresión lineal.

Como podemos observar en la Figura 50, los datos no presentan mucha dispersión, por consiguiente, se obtuvieron los siguientes resultados.

Tabla 7.

Resultados de los 3 proveedores con 0 dispositivos en su red.

	SIN DISPOSITIVOS EN SU RED			
PROVEEDOR	MEDIA DE RETARDO (ms)	DESVIACIÓN ESTÁNDAR DE RETARDO (ms)	ERROR ESTÁNDAR DE RETARDO (ms)	JITTER (ms)
CELERITY	83,38	9,01	8,77	10,37
NETLIFE	97,16	12,12	11,58	14,82
CNT	111,56	21,20	21,05	22,96

Nota, Esta tabla muestra un promedio general de los datos obtenidos sin dispositivos en la red.

Tabla 8.

Resultados de los 3 proveedores con 10 dispositivos en su red.

PRUEBA	CELERITY		NETLIFE		CNT	
	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)
1	160		115		103	
2	177	17	176	61	201	98
3	176	1	138	38	190	11
4	196	20	280	142	164	26
5	100	96	220	60	157	7
6	172	72	223	3	347	190
7	82	90	282	59	345	2
8	74	8	255	27	295	50
9	185	111	122	133	305	10
10	91	94	242	120	190	115
11	195	104	97	145	105	85
12	152	43	212	115	85	20
13	176	24	121	91	360	275
14	147	29	265	144	215	145
15	188	41	242	23	288	73
16	120	68	139	103	268	20
17	107	13	135	4	319	51
18	94	13	114	21	140	179
19	154	60	238	124	166	26
20	97	57	176	62	209	43
21	110	13	172	4	326	117
22	170	60	139	33	282	44
23	127	43	175	36	253	29
24	119	8	84	91	163	90
25	194	75	219	135	382	219
26	107	87	151	68	83	299
27	190	83	186	35	273	190
28	127	63	81	105	147	126
29	95	32	266	185	125	22
30	97	2	272	6	365	240
31	149	52	125	147	102	263
32	191	42	85	40	365	263
33	192	1	117	32	300	65
34	178	14	214	97	99	201
35	169	9	252	38	353	254
36	103	66	137	115	356	3
37	147	44	221	84	206	150
38	139	8	92	129	221	15
39	176	37	213	121	349	128
40	90	86	212	1	106	243
41	100	10	147	65	107	1
42	102	2	214	67	112	5
43	82	20	251	37	277	165
44	183	101	273	22	262	15
45	200	17	193	80	333	71
46	173	27	166	27	216	117
47	98	75	272	106	358	142
48	198	100	154	118	150	208
49	159	39	235	81	162	12
50	102	57	220	15	289	127
PROMEDIO	142,20	45,59	186,60	73,37	231,48	107,14

Nota, Esta tabla muestra el resultado del retardo en una comunicación inalámbrica con 10 dispositivos en la red.

Una vez realizadas las pruebas con diferente proveedor y 10 dispositivos en su red se tiene los siguientes resultados:

Figura 51.

Progresión lineal de los 3 proveedores con 10 usuarios en su red.

Nota, La imagen muestra los resultados gráficamente con 10 dispositivos en la red con su progresión lineal.

Como se muestra en la Figura 51, los datos tienen una mayor dispersión y mayor retardo, esto se debe a que el protocolo MQTT busca recibir y enviar los datos por otro camino donde no este cargado el tráfico de datos.

Tabla 9.

Resultados de los 3 proveedores con 10 dispositivos en su red.

CON DISPOSITIVOS EN SU RED				
PROVEEDOR	MEDIA DE RETARDO (ms)	DESVIACIÓN ESTÁNDAR DE RETARDO (ms)	ERROR ESTÁNDAR DE RETARDO (ms)	JITTER (ms)
CELERITY	142,20	40,13	40,52	45,59
NETLIFE	186,60	60,49	60,94	73,37
CNT	231,48	94,45	95,25	107,14

Nota, Esta tabla muestra un promedio general de los datos obtenidos con dispositivos en la red.

Terminadas las pruebas con los 3 proveedores designados se observa en la Tabla 7 y Tabla 9, que cuando la red no presenta tráfico da prioridad al sistema de brazo robótico, así se tiene que el proveedor CELERITY tiene mayor estabilidad en su red y presenta un error estándar de 8,77 ms en comparación con el error con 10 usuarios en la red de 45,59 ms, respecto a los demás proveedores, por esa razón para las pruebas de distancia se lo realiza con este proveedor para el análisis del Jitter

Para el sistema inalámbrico se realizaron pruebas de distancia en pasos de 20 m, estos datos dependen del router utilizado para las pruebas de distancia, pueden tener menores retardos en relación a la potencia de transmisión de datos del router y su configuración, a continuación, se presenta las características del router.

Tabla 10.

Características del router TP-LINK WR940N

CARACTERISTICAS	
VOLTAJE	12 VDC
ANTENAS	3 omnidireccionales fijas de 5 dBi
ESTÁNDARES	IEEE 802.11 b/g/n/
FRECUNECIA	2.4-2.4835
POTENCIA DE TRANSMISIÓN	< 20 dBm

Nota, Esta tabla muestra las características del Router a ser utilizado para las pruebas de distancia.

En las siguientes tablas se presenta los resultados de los retardos respecto a la distancia:

Tabla 11.

Resultados respecto a la distancia.

PRUEBA	20 METROS		40 METROS		60 METROS		80 METROS		100 METROS	
	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)	SISTEMA (ms)	JITTER (ms)
1	78		143		241		379		518	
2	84	6	163	20	237	4	355	24	508	10
3	89	5	147	16	240	3	350	5	490	18
4	79	10	152	5	253	13	352	2	503	13
5	100	21	156	4	242	11	370	18	505	2
6	75	25	156	0	237	5	352	18	517	12
7	72	3	144	12	255	18	379	27	512	5
8	100	28	147	3	239	16	379	0	493	19
9	72	28	169	22	256	17	352	27	491	2
10	84	12	159	10	260	4	375	23	506	15
11	85	1	150	9	253	7	353	22	510	4
12	83	2	155	5	258	5	362	9	498	12
13	86	3	170	15	249	9	369	7	496	2
14	89	3	158	12	231	18	357	12	497	1
15	88	1	159	1	255	24	353	4	491	6
16	98	10	160	1	230	25	380	27	504	13
17	92	6	151	9	246	16	352	28	510	6
18	88	4	146	5	242	4	354	2	497	13
19	85	3	163	17	236	6	361	7	520	23
20	77	8	142	21	234	2	353	8	517	3
21	79	2	155	13	243	9	377	24	503	14
22	83	4	146	9	256	13	357	20	504	1
23	100	17	156	10	257	1	374	17	512	8
24	92	8	168	12	232	25	369	5	498	14
25	72	20	143	25	248	16	360	9	504	6
26	80	8	140	3	230	18	369	9	491	13
27	90	10	141	1	246	16	379	10	501	10
28	91	1	140	1	241	5	374	5	513	12
29	79	12	148	8	244	3	350	24	507	6
30	98	19	161	13	246	2	362	12	500	7
31	79	19	170	9	244	2	380	18	493	7
32	96	17	160	10	252	8	380	0	500	7
33	87	9	163	3	255	3	354	26	493	7
34	86	1	166	3	242	13	358	4	493	0
35	100	14	167	1	252	10	354	4	518	25
36	90	10	170	3	236	16	358	4	503	15
37	98	8	141	29	255	19	374	16	518	15
38	81	17	164	23	260	5	380	6	508	10
39	76	5	142	22	254	6	367	13	520	12
40	73	3	143	1	259	5	358	9	505	15
41	86	13	163	20	254	5	370	12	492	13
42	85	1	165	2	260	6	366	4	519	27
43	76	9	162	3	232	28	367	1	490	29
44	73	3	167	5	253	21	355	12	501	11
45	98	25	159	8	257	4	354	1	490	11
46	91	7	149	10	235	22	368	14	506	16
47	82	9	142	7	248	13	363	5	508	2
48	74	8	153	11	239	9	352	11	509	1
49	81	7	141	12	241	2	361	9	492	17
50	94	13	152	11	234	7	351	10	497	5
PROMEDIO	85	9,76	155	9,69	246	10,59	364	11,92	503	10,51

Nota, Esta tabla muestra el resultado del retardo respecto a la distancia con pasos de 20 m.

Como se observa en Tabla 12 a mayor distancia presenta mayores retardos en la señal, pero presenta un Jitter con una variación mínima, el cual varia respecto a la invasión tráfico en la red de datos, para estas pruebas de distancia se utilizó el proveedor Celerity que dio el menor retardo promedio sin invasión de tráfico. Que el Jitter se eleve en valores mínimos se debe a que en la red del proveedor Celerity presenta mayor ancho de banda y menor nodos en la red de Internet. Por consiguiente, se obtuvieron los siguientes resultados:

Tabla 12.

Resultados respecto a la distancia.

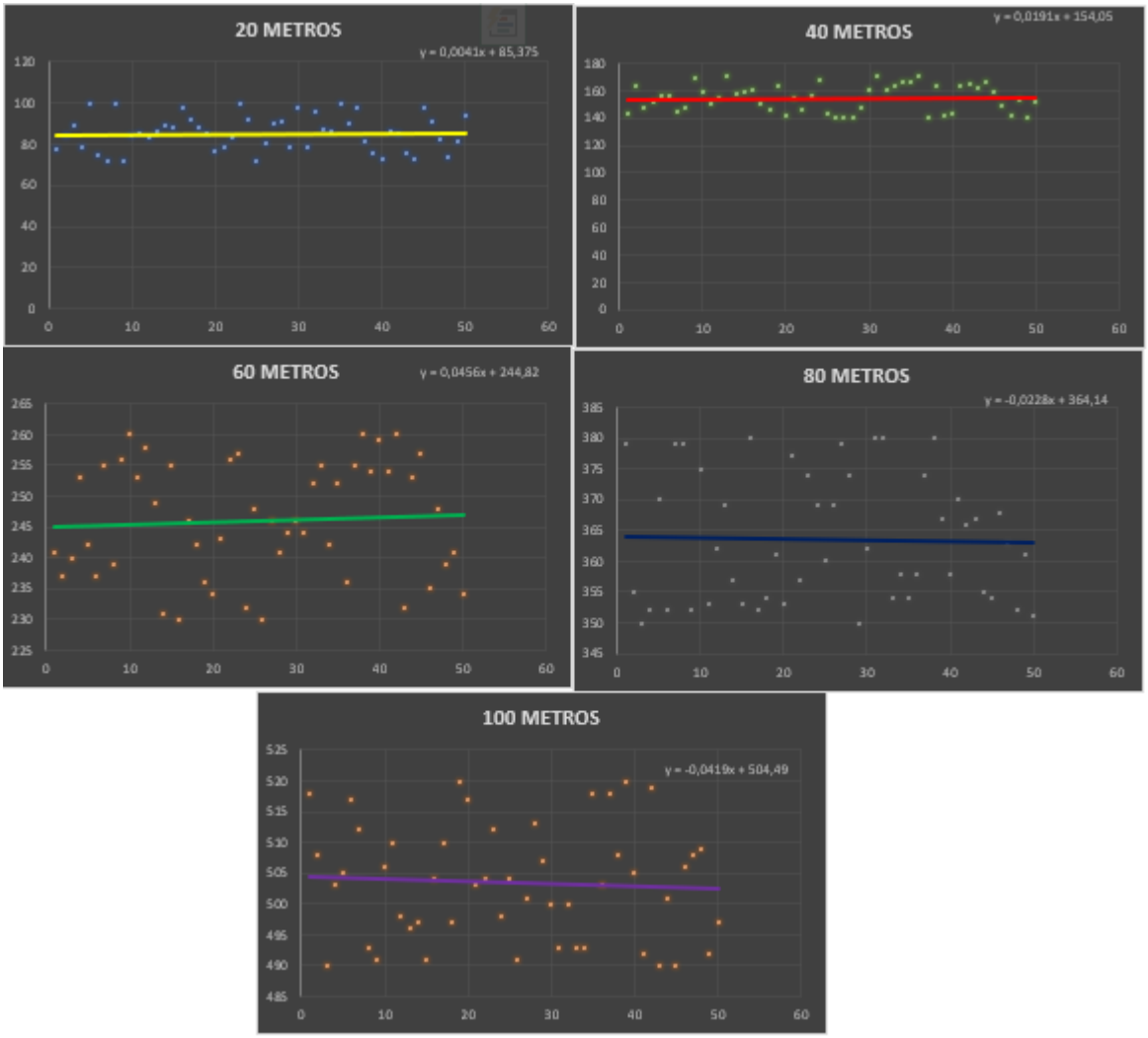
	PRUEBAS DE DISTANCIA			
DISTANCIA	MEDIA DE RETARDO (ms)	DESVIACIÓN ESTÁNDAR DE RETARDO (ms)	ERROR ESTÁNDAR DE RETARDO (ms)	JITTER (ms)
20 metros	85	8,67	8,53	9,76
40 metros	155	28,21	27,56	9,69
60 metros	246	36,03	35,55	10,59
80 metros	364	78,71	78,01	11,92
100 metros	503	139,67	138,12	10,51

Nota, Esta tabla muestra un promedio general de los datos obtenidos respecto a la distancia.

Como se puede observar en la Figura 52 el sistema presenta una menor dispersión en los datos en referencia a su distancia. Y mientras que el error estándar se eleva mientras mayor es la distancia. Así este proyecto demuestra que mientras tienes un sistema inalámbrico dedicado para sistemas en tiempo real el Jitter varia muy poco y con esta perspectiva se puede minimizar el Jitter en otros proyectos de investigación.

Figura 52.

Progresión lineal respecto a la distancia



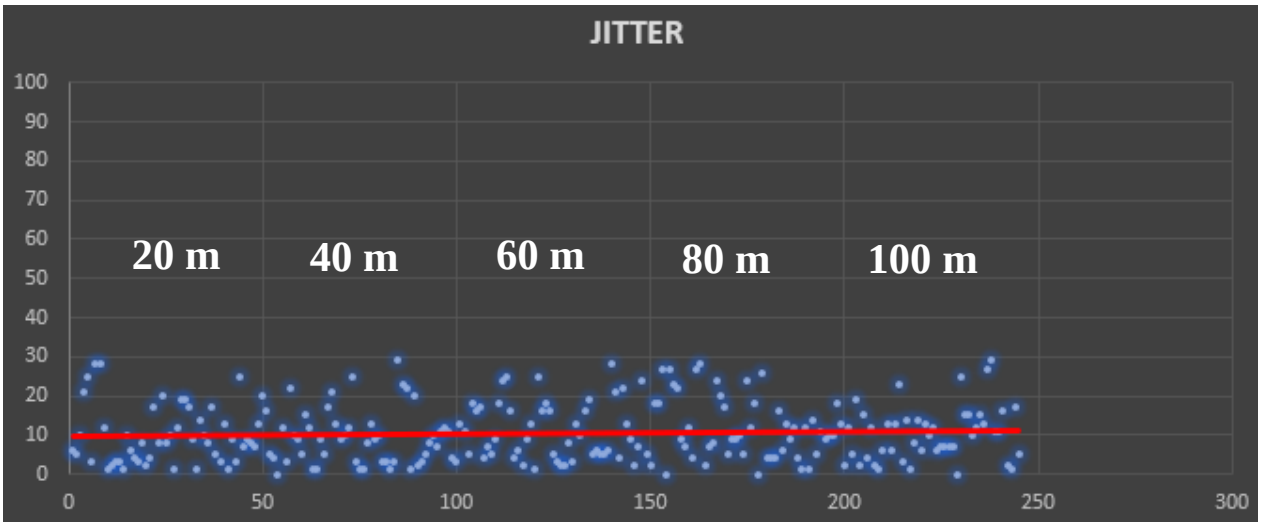
Nota, La imagen muestra los resultados gráficamente en las pruebas de distancia con su progresión lineal.

Para visualizar de mejor manera y reafirmar la teoría propuesta, que el Jitter no varía respecto a la distancia en sistema sin invasión de tráfico, se realiza la gráfica de dispersión de todos los datos del Jitter con todas las distancias en una sola ilustración como se muestra en la

Figura 53 y si se realiza la comparación respecto a distancias el Jitter no presenta mayor dispersión.

Figura 53.

Progresión lineal respecto al Jitter



Nota, La imagen muestra los resultados gráficamente del Jitter respecto a las pruebas de distancia.

Capítulo V: Conclusiones y trabajos futuros

Conclusiones

El desempeño de la tecnología WiFi sin tráfico invasivo para sistemas en tiempo real, se tiene que a mayor ancho de banda el retardo en la comunicación inalámbrica es menor, esto se verifica con los siguientes resultados, en el cual se realizaron pruebas con tres proveedores de Internet Celerity, Netlife y CNT, presentándose los siguientes valores 83.38 ms, 97.16 ms, y 111.56 ms respectivamente, siendo Celerity el de mayor ancho de banda ya que presenta el menor retardo, con estos resultados realizamos el cálculo del error estándar el cual nos da valores de 8.77 ms, 11.58 ms y 21.05 ms para cada proveedor respectivamente, y esto nos da a conocer cuanto retardo puede variar en la comunicación Inalámbrica y con estos valores se tiene que el Jitter promedio del sistema en los siguientes valores 10.37 ms, 14.82 ms y 22.96 ms, demostrando así que el ancho de banda influye mucho en los resultados y también el número de nodos que presente un su red de Internet cada uno de los proveedores, el cual permite que los datos viajen por nodos más cortos o nodos más largos que hacen que se produzca esta variación de retardo.

El proyecto se ha desarrollado con el fin evaluar el desempeño de la tecnología WiFi con inyección de tráfico invasivo y no invasivo que era unos de los objetivos planteados pero en el proceso de desarrollo del proyecto no se encontró un software que cumpliera con esa función real para inyectar tráfico a la red, para lo cual, para que se acerque a lo más real posible la inyección de tráfico se realizó pruebas con 10 dispositivos conectados a la red los cuales todos estaban sincronizados para enviar videos, video llamadas, datos etc. De esta manera se obtuvieron los siguientes resultados para los mismos 3 proveedores Celerity, Netlife y CNT, su retardo medio es 142.2 ms, 186.60 ms y 231.48 ms, aun así, se tiene retardos aceptables y esto sucede ya que los datos enviados no ocupan mucho espacio, son muy pequeños y una principal

característica del protocolo MQTT es dar la prioridad a este tipo de datos que no ocupan mucho ancho de banda, lo que si debemos ser conscientes es que el protocolo no minimiza el Jitter por la prioridad del envió de datos, basándonos en esto el Jitter presenta los siguientes resultados 45.59 ms 73.37ms, y 107.14, esto demuestra que los datos tienden a tomar diferentes caminos con el fin de llegar al destino.

Para quedar aún más claros con el análisis de desempeño de la tecnología WiFi se realizaron pruebas solo con el proveedor del cual se tiene menor retardo medio, para el caso Celerity, se realizaron pruebas con distancias cortas menores a 20 m y presentaba casi el mismo retardo de un sistema sin dispositivos, por lo que se decidió extender el panorama pero con línea de vista directa, para lo cual se extendió hasta 100 m ya que al ir unos metros más allá hasta los 110 m se perdía el enlace de comunicación inalámbrica, para lo cual se toma muestras en pasos de 20 m , teniendo así los intervalos en 20m, 40m, 60 m, 80m, y 100m, demostrando así que la distancia en un sistema inalámbrico de tiempo real, afecta al retardo en la comunicación mas no afecta al Jitter ya que en las pruebas de distancia no se conectó ningún dispositivo a la red que afecte la latencia del Jitter, en este contexto se tiene los siguientes resultados del retardo medio 85 ms, 155 ms, 246 ms, 364 ms y 503ms respectivamente ara cada una de las distancias propuestas y a si mismo el resultado del Jitter es el siguiente 9.76 ms, 9.69 ms, 10.59 ms, 11.92 ms y 10.51 ms, por consiguiente, por medio de estos resultados en la investigación y analizada la bibliografía general de estos sistemas donde no hablan sobre este tipo de análisis de desempeño en tiempo real, me atrevo a afirmar que el Jitter se mantiene constante en un sistema inalámbrico con tecnología WiFi basado en el estándar IEEE802.11n siempre y cuando en el sistema este dedicado solo para este propósito de prototipos teledirigidos con la tecnología WiFi.

El protocolo MQTT es muy seguro por los tópicos que maneja, el cual fue implementado con éxito pero cabe recalcar que una de las recomendaciones es que cada cliente debe tener su

ID propio con el fin de no confundir al protocolo MQTT y no se pierda paquetes de información por lo cual, no pueden existir 2 clientes con el mismo nombre, permitiendo así el control del brazo robótico en tiempo real por medio de un servidor AWS que se implementó con la plataforma Ubuntu y configuraciones básicas de un ordenador.

El sistema fue programado en el IDE de Arduino, el cual por medio de librerías se programó el módulo ESP32 el cual recoge la señal filtrada de los *flex sensor* por medio de las entradas ADC las procesa y convierte en datos los cuales son enviadas al servidor el cuál envía al receptor y envía la señal PWM al brazo robótico para ser movido en la posición correcta.

Se considera que las señales tienen un tiempo de retardo así mismo el servomotor el cual por la señal PWM presenta un retardo de 30 ms de respuesta ante las señales recibidas, no obstante, debe tener una buena corriente con el fin de que tenga potencia y fuerza para levantar un peso adecuado.

Los resultados arrojados por el sistema de comunicación inalámbrica han sido comparados con diferentes proveedores de internet lo cual es un factor muy importante al realizar este tipo de proyectos, ya que las empresas nos ofrecen altos anchos de banda y un equipo con poca potencia de transmisión y recepción de datos para lo cual se debe verificar que la red de internet sea estable, y así tener una menor latencia a los retardos.

Los diferentes proveedores de internet en el ecuador manejan diferentes tasas de transmisión o dependen de un plan contratado en la velocidad de envió de información el cual es un factor predominante en los Sistema IoT ya que las pruebas demuestran mayor retardo dependiendo del ancho de banda. También depende del número de dispositivos conectados a la misma red, se recomienda para sistemas IoT obtener un ancho de banda dedicado solo para

este propósito por el retardo que presenta la señal con otros dispositivos conectados a la misma red.

Una vez revisada las normas y estándares de la UIT conjuntamente con bibliografía científica se observa que no se tiene un tiempo de retardo estimado para este tipo de aplicaciones en tiempo real, por lo que el centro de investigación de la Universidad de las Fuerzas Armadas ESPE y desarrollador del presente proyecto, basándose en las pruebas realizadas, propone que el retardo mínimo para sistemas en tiempo real el retardo medio mínimo es 83 ms, en las condiciones presentadas en el presente proyecto, tendiendo así a la constante mejora y posible minimización de este tipo de retardos.

Trabajos futuros

Nuestro grupo de investigación está interesado en realizar pruebas dentro de las instalaciones de la Universidad de las fuerzas armadas ESPE donde se presenta un mayor número de usuarios conectados a la red con el objetivo de realizar el mismo análisis de desempeño en cuestión de retardo y Jitter

Así mismo diseñar e implementar un sistema inalámbrico con el objetivo de minimizar la varianza de retardo (Jitter), el cual como se muestra en el presente proyecto es un problema ya que los paquetes por Internet tienden a tomar diferentes caminos por lo que se produce el Jitter.

Trabajar en el diseño de nuevos robots que no ocupen servomotores mecánicos para la minimización de retardos por sistemas mecánicos.

Basándonos en los resultados obtenidos respecto a un Jitter contante se propone el diseño e implementación con otros protocolos con el fin de minimizar el Jitter ya que en la práctica

se consiguió mantener el Jitter constante, y se puede re alizar estudio por medio de fibra óptica y otros estándares de comunicación y protocolos aún más potentes.

Bibliografía

Abro, K. A., Gómez-Aguilar, J., Khan, I., & Nisar, K. (2019). Role of modern fractional derivatives in an armature-controlled DC servomotor. *The European Physical Journal Plus, 134*(11), 533. https://doi.org/Springer Berlin Heidelberg

Alvarado Clavijo, F. A. (2011). *Mano robótica inalámbrica .*

Arenas, M. A., Palomares, J. M., Girard, L., Olivares, J., & Castillo Secilla, J. M. (2021). Diseño y Construcción de un Guante de Datos mediante Sensores de Flexibilidad y Acelerómetro. *Jornadas Sarteco 2021*.

Arias Montiel, M., Martínez Miguel, A., Lugo González, E., Miranda Luna, R., & Tapia Herrera, R. (2021). Prototipo de mano robótica controlado mediante señales electromiografías con un dispositivo comercial. *Computacion y Sistemas, 25*(2), 307-315.

Ceja, J., Rentería, R., Ruelas, R., & Ochoa, G. (2017). Módulo ESP8266 y sus aplicaciones en el internet de las cosas. *Revista de Ingeniería eléctrica, 1*(2), 24-36.

Cortés Diart, Ó., Sánchez, G., Roldán Mckinley, J., & Yime Rodríguez, E. (2014). RSM en la minimización del retardo en la red durante tele operación de robots. *e-colabora Revista de ciencia, educación, innovación y cultura apoyadas por redes de tecnología avanzada", 2*(4), 118-140.

Delta. (2022). *Delta*. https://shopdelta.eu/802-11n-estandar-de-la-red-inalambrica_l6_aid756.html

Esneca. (2022). *Esneca Business School.* https://www.esneca.com/blog/brazo-robotico-industrias/

Faidallah, E. M., Hossameldin, Y. H., Rabbo, S. M., & El-Mashad, Y. A. (2015). Control and modeling a robot arm via EMG and flex signals,. *15th Int. Work. Res. Educ. Mechatronics, REM 2014*. https://doi.org/10.1109/REM.2014.6920226

Hower Ceballos, J. E., Correa Arroyave, C. A., & Pareja Rodríguez, C. U. (2019). *Análisis y diseño de un prototipo de una mano robótica con catorce grados de libertad, capaz de ser dirigida a través de Internet en tiempo real.*

Lara Cueva, R. A., Fernández Jimenez, C. B., & Maldonado, M. (2016). *Anáisis del desempeño en un enlace descendente de redes basadas en los estándares.*

Miqdad, A., Suhairi, R., Ali, A. M., Roslan, N. F., & Aziz, P. D. (2015). Development of artificial hand gripper by using flex force sensor. *2014 4th Int. Conf. Eng. Technol. Technopreneuship, ICE2T 2014, 2014-August*, 305–308. https://doi.org/10.1109/ICE2T.2014.7006267

Misal, S. R., Prajwal, S. R., Niveditha, H. M., Vinayaka, H. M., & Veena, S. (2020). Indoor Positioning System (IPS) Using ESP32, MQTT and Bluetooth. *Proc. 4th Int. Conf.*

Comput. Methodol. Commun. ICCMC 2020(Iccmc), 79–82. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00015

Naylamp. (2021). *Mechatronics SAC*. https://naylampmechatronics.com/blog/41_tutorial-modulo-controlador-de-servos-pca9685-con-arduino.html

Paessler. (2022). *IT Explained: MQTT*. https://www.paessler.com/es/it-explained/mqtt

Puja, D., & Yashwant G., A. (2017). Wireless Robotic Hand for Remote Operations using Flex Sensor. *Int. Conf. Autom. Control Dyn. Optim. Tech. ICACDOT 2016*, 114–118. https://doi.org/ 10.1109/ICACDOT.2016.7877562

Robotic. (2022). *Robotics Ecuador*. Retrieved 7 de 2019, from https://roboticsec.com/producto/kit-brazo-robotico-plastico-garra/

RZ redes zone. (2022). *RZ edes zone*. https://www.redeszone.net/tutoriales/redes-wifi/latencia-alta-usar-wi-fi-motivos-soluciones/

Yue, M., Ruiyang, Y., Jianwei, S., & Kaifeng, Y. (2017). A MQTT Protocol Message Push Server Based on RocketMQ. *Proc. - 10th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2017, 2017-October*, 295–298. https://doi.org/10.1109/ICICTA.2017.72

Zapata Rodríguez, M. A. (2018). *Evaluación de parámetros de calidad de servicio (QOS) para el diseño de una red VPN con MPLS.* PUCE.

Zare, A., & Iqbal, M. T. (2020). Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT protocol based SCADA system. *IEMTRONICS 2020 - Int. IOT, Electron. Mechatronics Conf. Proc.*, 0-4. https://doi.org/10.1109/IEMTRONICS51293.2020.9216412