Implementación de una red inalámbrica de cámaras integrada a un sistema mecatrónico para el monitoreo de entornos residenciales y el manejo remoto de puertas usando MQTT.

Juan Sebastian Trujillo Loaiza

Universidad Antonio Nariño

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Bogotá D.C, Colombia

2022

Implementación de una red inalámbrica de cámaras integrada a un sistema mecatrónico para el monitoreo de entornos residenciales y el manejo remoto de puertas usando MQTT.

Juan Sebastian Trujillo Loaiza

Proyecto de grado presentado como requisito parcial para optar al título de:

Ingeniero Mecatrónico

Director:

Ph.D. Sergio Andrés Díaz Salas

Línea de Investigación:

Redes inalámbricas de sensores, Internet de las cosas

Grupo de Investigación:

GIBIO - Grupo de Investigación en Bioinstrumentación y Control

Universidad Antonio Nariño

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Bogotá D.C, Colombia

2022

Agradecimientos

En primer lugar, quiero agradecer a mis padres, que me han brindado todo su cariño y confianza dándome continuo soporte para lograr mis sueños y enseñándome cada día a ser una mejor persona. A mis hermanos, que me han apoyado en los buenos y malos momentos, ofreciéndome su ayuda incondicional y no dejándome caer en los momentos más difíciles.

También quiero expresar mi profundo agradecimiento al PhD. Sergio Andrés Diaz Salas, que con su gran compromiso, paciencia, amplio conocimiento y buen criterio me ha guiado y supervisado en el desarrollo de este proyecto de grado, permitiéndome así culminar con este proceso tan importante en mi vida. Le doy también las gracias a todos los profesores con los que tuve la oportunidad de aprender en el transcurso de mi carrera, y de los cuales me llevo todas las enseñanzas que con convicción impartieron en sus clases.

Resumen

En el presente proyecto se propone una red inalámbrica de cámaras (RIC) compuesta de cámaras como sensor principal, que, conectada a un sistema mecatrónico, permite la actuación de motores DC y así el manejo remoto de puertas. La comunicación entre la RIC y el sistema mecatrónico será gestionada por un servidor MQTT Bróker. La red de cámaras se limitará a cuatro cámaras ubicadas dentro de una residencia, mientras que el sistema mecatrónico contará sólo con 2 salidas de actuación.

El proceso a seguir para lograr este propósito consta de cinco fases: i) La implementación de la RIC para detección por cámaras; ii) La construcción del sistema mecatrónico para la actuación de las puertas, iii) La integración de ambos sistemas por medio de MQTT, iv) Reconocer el rostro de los transeúntes con el fin de generar una alarma para apoyar la labor del vigilante y v) La evaluación de la solución propuesta a partir de variables como la velocidad de traslación de la puerta, el torque del motor, la latencia, la cantidad de paquetes perdidos, la intensidad de señal recibida (RSSI) y el consumo de energía. Así mismo, toda esa información será expresada en gráficas que mostrarán a detalle el comportamiento de la RIC bajo diferentes condiciones de prueba.

Palabras clave: Internet de las Cosas, MQTT, Red Inalámbrica de Cámaras, Sistema Mecatrónico, Videovigilancia Residencial

Abstract

In this project, a wireless camera network (RIC) is proposed, composed of cameras as the main sensor, which, connected to a mechatronic system, allows the actuation of DC motors and thus the remote control of doors. The communication between the RIC and the mechatronic system will be managed by an MQTT Broker server. The camera network will be limited to four cameras located within a residence, while the mechatronic system will only have 2 actuation outputs.

The process to follow to achieve this purpose consists of five phases: i) The implementation of the RIC for detection by cameras; ii) The construction of the mechatronic system for the actuation of the doors, iii) The integration of both systems through MQTT, iv) Recognize the face of passers-by in order to generate an alarm to support the work of the security guard and v) The evaluation of the proposed solution based on variables such as the door translation speed, motor torque, latency, the number of lost packets, received signal strength (RSSI) and power consumption. Likewise, all this information will be expressed in graphs that will show in detail the behavior of the RIC under different test conditions.

Keywords: Wireless Camera Network, Residential Surveillance, MQTT, Mechatronic system, Internet of Things

Tabla de contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de Símbolos y abreviaturas

Símbolos con letras latinas

Símbolo	Término	Unidad SI	Definición
$\Delta R_{cremallera}$	Desplazamiento de cremallera	m	Ec. 2.1
$v_{cremallera}$	Velocidad lineal de cremallera	m/s	Ec. 2.2, 3.3
m	Pendiente	1	Ec. 3.2
r_p	Radio del piñón	m	Ec. 3.4
F_c	Fuerza tangencial de cremallera	N	Ec. 3.5
V	Volumen de puerta	m^3	Ec. 3.7
P	Potencia eléctrica	W	Ec. 4.1
t_{vida}	Tiempo de vida de batería	s	Ec. 4.2

Símbolos con letras griegas

Símbolo	Término	Unidad SI	Definición
τ_m	Torque del motor	$N \cdot m$	Ec. 3.1
ρ	Densidad de la puerta	kg/m^3	Ec. 3.6
μ	Coeficiente de fricción	1	Sección 3.2.3

Abreviaturas

Abreviatura	Término
RIC	Red Inalámbrica de Cámaras
RIS	Red Inalámbrica de Sensores
RSSI	Received Signal Strength Indicator
MQTT	MQ Telemetry Transport
MRU	Movimiento Rectilíneo Uniforme

Introducción

Hoy en día se puede evidenciar el uso y la aplicación de la tecnología en muchas de las actividades cotidianas de las personas y sus entornos. Tomando en cuenta el comportamiento humano se hace necesario el uso de tecnologías como la implementación de sistemas de seguridad y de monitoreo de diferentes espacios, en función de tener un control en las acciones de las personas. Dichos sistemas cumplen principalmente tareas de vigilancia y permiten la detección de comportamientos inseguros. Sin embargo, esto en ocasiones puede no ser suficiente, por ejemplo, en escenarios como hurtos y agresiones: Suelen cometerse rápidamente y la detección puede llegar a ser tardía, además no se garantiza el ejercicio de la seguridad.

En el artículo [1] se presenta un análisis del caso donde se describe el uso del circuito cerrado de televisión (CCTV) para la vigilancia contra robos, siendo esta la estrategia más común en la prevención de este tipo de delitos. Sin embargo, también se mencionan los inconvenientes de usar el CCTV, que, además del excesivo uso de recursos como energía, almacenamiento e incluso de recursos humanos, también se evidencia una falta de automatización y de obtención de datos, que pueden ser vitales a la hora de contrarrestar alguna irregularidad.

A fin de solventar dichos inconvenientes ya se ha propuesto como una alternativa el uso de cámaras conectadas a ordenadores de placa reducida Raspberry Pi, como es el caso de los autores en [2],que además incluyen un sensor infrarrojo PIR (Por sus siglas en inglés Passive Infrared Sensor) para detección de movimiento. Todo esto conectado a internet, permite una conexión al móvil mediante una aplicación web, que se encarga de gestionar varias de las funcionalidades de este sistema.

El presente documento plantea un sistema que, si bien tiene similitudes con aquel mencionado, va a diferenciarse en que se va a emplear como elemento principal el módulo ESP32-CAM, que también posee una cámara para registrar imagen y video y la capacidad

de conectarse a internet mediante Wifi. Así mismo, el proyecto difiere en que se considerará la aplicación de más de uno de estos dispositivos, que irán conectados entre sí mediante internet y formarán una red inalámbrica de cámaras, cuyo funcionamiento pueda controlarse por medio de la programación de cada sensor. La configuración por programación se puede observar en el trabajo realizado en [3], donde se usan algoritmos que, implantados en los sensores, les dan la capacidad de conectarse entre sí.

Sin embargo, se requiere de un análisis para lograr la construcción de una red inalámbrica de cámaras (RIC). Esto incluye aspectos como rendimiento en la transmisión de datos y las pérdidas dadas por condiciones externas (tráfico de la red, disposición de espacios, distancias entre sensores, entre otros). En resumen, se plantea estimar el rendimiento de la RIC con la obtención de variables tales como latencia, intensidad de señal recibida (RSSI), cantidad de paquetes perdidos y consumo de energía. El estudio realizado en [4] considera estas variables para verificar el rendimiento en diferentes topologías de la RIC, poniendo en evidencia lo conveniente de analizar este tipo de información para obtener mejores resultados.

Por otra parte, al poner en funcionamiento la RIC propuesta, se busca resolver todas aquellas deficiencias de un sistema común de seguridad, provenientes del uso único de cámaras, sin ningún apoyo diferente al almacenamiento de datos de grabación. De este modo, se contempla la implementación de un tipo de sensor diferente a las cámaras, que cumpla funciones diferentes a la recepción de imágenes y video. Esto es posible mediante la interconexión de diferentes tipos de sensores por medio del internet de las cosas. Por ejemplo, en el artículo [5] los autores lograron conectar entre sí sensores de temperatura, humo y humedad con el fin de obtener un sistema de automatización en un establecimiento residencial.

Para este proyecto, se propone el uso de las cámaras como sensores, donde ellas mismas tengan capacidades de adquisición de información y que por métodos de detección, puedan activar otros dispositivos a favor de la seguridad. Un ejemplo es el trabajo del autor en [6], que simuló la apertura y cierre de una puerta a través de un motor DC, y cuyo movimiento es gestionado a través de cualquier dispositivo con acceso a la IP del módulo del sistema (Raspberry Pi). Para este caso, además de permitir el movimiento manual, se incluye la actuación automática de uno o más motores de acuerdo a ciertos parámetros

dados por los sensores. Estos pueden transmitir movimiento rotacional y traslacional teniendo en cuenta su instalación y uso, como se ha referido en [7].

Otro caso de aplicación de sistemas anexados a un dispositivo inalámbrico corresponde al tratado en [8], donde se hace el manejo de forma autónoma de compuestas con motores DC como actuadores, a través del microcontrolador Arduino Uno. Este trabajo destaca en cuanto a la utilización de diversos tipos de elementos que, en resumen, cumplen tareas de recolección de datos, procesamiento y de actuación, convirtiéndolo en un sistema mecatrónico. El presente documento plantea la aplicación de un sistema de este tipo, aplicando el mismo concepto y creando así un sistema mecatrónico que pueda comunicarse con la red inalámbrica de cámaras.

Además de los sensores, este sistema requiere de un método de administración de las funciones y características de la RIC en conjunto. El uso de aplicaciones presenta una muy buena opción, como se presenta en el artículo [9]. En este artículo, al igual que en algunos otros antes mencionados, se emplean este tipo de recursos con el fin de tener mayor accesibilidad a la red inalámbrica de cámaras, y dada la facilidad del usuario con el dispositivo, las “apps” se convierten en una herramienta conveniente para la gestión de un sistema de seguridad.

Ya cubierta la interacción usuario – sistema, otro aspecto a considerar es el tamaño de la red. De acuerdo a [10], si el sistema es muy grande se pueden presentar diversos inconvenientes en lo que respecta a la transmisión de información. En el documento se menciona que, entre otras cosas, se puede presentar dificultad en la administración de cada nodo de información que manejan los diferentes sensores, lo que significa un aumento de tráfico de datos y en consecuencia posibles pérdidas o ralentizaciones en el envío de información. Por lo tanto, en el sistema propuesto también se tendrá en cuenta el rendimiento del sistema de acuerdo a la cantidad de elementos conectados en la red.

A continuación, se presenta una tabla comparativa, que permite visualizar los aspectos de análisis por parte de cada trabajo relacionado (incluyendo el presente trabajo):

Ref.	Protocolo de comunicación	Tipo de sensor	Sistema embebido	Alimentación	Latencia	RSSI	Cantidad de paquetes perdidos	Consumo de energía
Este trabajo	WiFi	Cámara OV2640	Módulo ESP32 CAM	Baterías	SI	SI	SI	SI
[1]	WiFi	Cámara Raspberry Pi Rev. 1.3, Sensor infrarrojo pasivo PIR.	Raspberry Pi 3	Paneles solares	SI	NO	NO	NO
[2]	Ethernet o WiFi	Cámara Raspberry Pi, Sensor infrarrojo pasivo PIR.	Raspberry Pi 3	Cualquier dispositivo con entrada USB, de 5V a 1A.	NO	NO	NO	NO
[3]	WiFi	No especifica (cualquier sensor o router)	No especifica	Baterías	NO	NO	SI	SI
[4]	NS2 (Simulación)	No aplica	No aplica	No aplica	SI	NO	SI	SI
[5]	WiFi	Sensor de humedad y temperatura DHT11, sensor de fuga de gas MQ2, sensor de flama LM393 junto a un fototransistor infrarrojo, sensor PIR HC-SR501, sensor de lluvia M009, sensores de voltaje y corriente.	Módulo ESP8266	No especifica	NO	NO	NO	NO
[6]	WiFi y Ethernet	Cámara Raspberry Pi	Raspberry Pi 3	Alimentado del computador (experimentación)	NO	NO	NO	NO
[7]	No aplica	No aplica	No aplica	No aplica	No aplica	NO	No aplica	No aplica
[8]	No aplica	Sensores de indicación del nivel del agua (diseñado con transistores)	Arduino Uno	No especifica	No aplica	No aplica	No aplica	No aplica
[9]	WiFi y Ethernet	Cámara Raspberry Pi, Sensor infrarrojo pasivo PIR.	Raspberry Pi 3	Alimentado del computador (experimentación)	NO	NO	NO	NO
[10]	WiFi	No aplica	No aplica	Baterías	NO	NO	NO	NO

1. Capítulo 1. Planteamiento del problema

1.1. Justificación

La relevancia de aplicar este proyecto es que se basa principalmente en el internet de las cosas (IoT), lo que implica entrar en un campo de la tecnología que se expande rápidamente con el paso del tiempo, en términos de importancia y aplicación. Esto se puede evidenciar a nivel global, donde la gran cantidad de elementos conectados a la red muestra la virtud de esta tecnología aplicada a diferentes áreas de la producción. De acuerdo al artículo “IOT EN ALC 2019: Tomando el pulso al Internet de las Cosas en América Latina y el Caribe” [12], del Banco Interamericano de Desarrollo (BID), en el 2018 se encontraban unos 10 mil millones de dispositivos conectados a internet alrededor del mundo, y se espera que para el 2025 esa cantidad sea superior a los 65 mil millones, dando en evidencia la rápida expansión del IoT. De hecho, se estima que Colombia será el segundo país con mayor crecimiento en inversión de IoT (24.9 CAGR) siendo sólo superado por México (28.3 CAGR), indicando que a nivel nacional el IoT se encuentra en plena fase de inspección y reconocimiento.

Teniendo en cuenta lo anterior, la implementación de una RIC en conjunto con un sistema mecatrónico presenta un aporte en el uso de las tecnologías que busquen ayudar a satisfacer las necesidades de las personas, especialmente en Colombia. Esto permite una exploración y una eventual aplicación de sistemas implementados en conjunto con IoT, que tienen un alto potencial y un sinfín de aplicaciones y usos, ya sea desde sistemas complejos en fábricas o establecimientos militares, hasta aplicaciones en la vida cotidiana, en entornos tan comunes como los hogares.

En términos de aplicación, el presente proyecto presenta una alternativa más eficiente, accesible y económica para conformar un sistema de seguridad, que incluso podría ser más fácil de manejar para los usuarios finales (e.g., habitantes de una residencia). La red

inalámbrica de cámaras, cuyo eje principal de funcionamiento es el módulo ESP32 CAM, cuenta con la facilidad de operación, el reducido tamaño comparado con otros sistemas embebidos, la conveniente integración con una cámara OV2640 y el potencial empleo de ésta como sensor: Es posible adaptar este dispositivo como detector de rostros, incluyendo en su programación librerías que habilitan esta funcionalidad. Así mismo, el sistema mecatrónico que se incorpora a la RIC, con funciones de actuación y el movimiento de puertas en base a la respuesta de las cámaras, presenta ciertas ventajas en las que destacan el bajo costo de sus elementos, el tamaño del sistema, entre otras. El sistema completo pretende detectar rostros con la RIC y emitir una alarma a un administrador del sistema, éste decidirá el manejo de las puertas dando instrucciones al sistema mecatrónico, que finalmente generará movimiento con la actuación de motores incorporados.

1.2. Problemática

En la actualidad, establecimientos cerrados hacen uso de sistemas de vigilancia para ofrecer un seguimiento al entorno y garantizar la seguridad. En Colombia, pequeños y medianos negocios cuentan con sistemas de CCTV, dado que son accesibles en precio y disponibilidad [11]. Sin embargo, la eficiencia de este tipo de sistemas ha sido puesta a prueba en diferentes circunstancias (robos, vandalismo, entre otros) mostrando que en ocasiones, la videovigilancia no es suficiente.

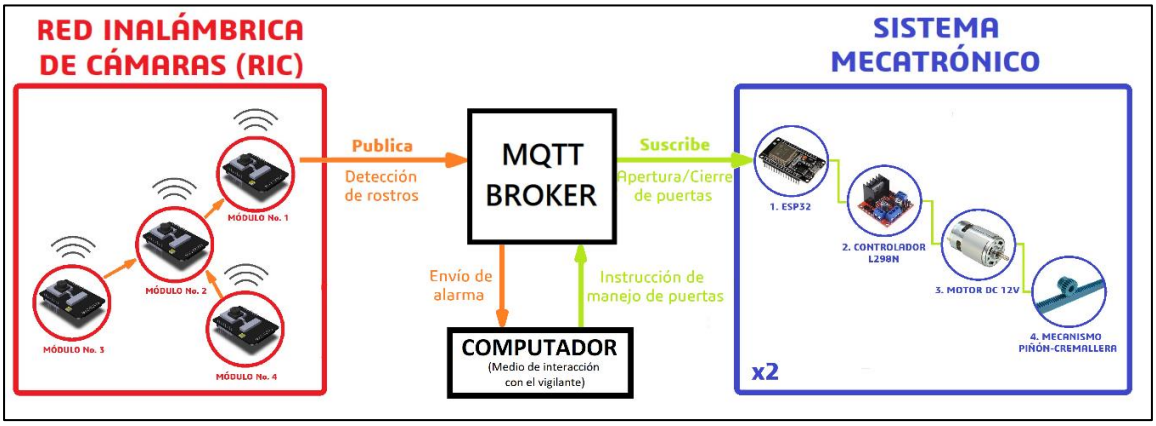
Específicamente, no sólo se habla de la detección y grabación por medio de cámaras de video (funciones realizadas por los sistemas de vigilancia convencionales), sino también la necesidad de incorporar otro tipo de características que refuercen esas funciones. De este modo, se considera una red inalámbrica de cámaras que, además de tener conectividad inalámbrica vía WiFi, cuenta con funcionalidades adicionales tales como la integración con sistemas mecatrónicos de actuadores.

Utilizando una red inalámbrica de cámaras en conjunto con el sistema mecatrónico de actuadores se puede abarcar una funcionalidad adicional de la seguridad en una residencia: la apertura y cierre remoto de puertas; el objetivo de esta funcionalidad adicional es bloquear el acceso a determinados espacios en la residencia. Para manejar

la apertura y cierre de puertas se propone diseñar un sistema mecatrónico de actuadores con la capacidad de generar movimiento traslacional a la puerta. Dicho sistema mecatrónico estará conectado a la red inalámbrica de cámaras con el fin de controlar el bloqueo de espacios de forma remota y teniendo en cuenta la información suministrada por las cámaras. Con esta configuración las cámaras cumplen la función de reconocimiento de apoyo, detectando la presencia de personas y enviando alarmas a un vigilante, quién decidirá al final el manejo de las puertas. La integración de la red inalámbrica de cámaras, el sistema mecatrónico de actuadores y la interacción con el usuario es posible gracias al protocolo MQTT (de siglas MQ Telemetry Transport), que asegura la conectividad entre diferentes dispositivos de forma inalámbrica.

MQTT permite la integración los subsistemas RIC y sistema mecatrónico, además de permitir la interacción de ambos con un administrador de la red (usuario). Específicamente, el MQTT broker es un 'mediador' entre los subsistemas y el administrador, que se encarga de manejar la transmisión y recepción de información. De esta forma en la Figura 1.1, se muestra cómo la RIC y el sistema mecatrónico se integran por medio de MQTT, Siendo este último el medio por el cual el usuario puede decidir la operación del sistema.

Figura 1.1. Arquitectura MQTT que integra la red inalámbrica de cámaras y el sistema mecatrónico.



Recordemos que MQTT es un protocolo que trabaja con el paradigma de publicación y subscripción; así, la RIC se encarga de publicar instrucciones para la apertura y cierre de puertas y el sistema mecatrónico se suscribe a dichas publicaciones y ejecuta las instrucciones.

Finalmente, es necesario garantizar el correcto funcionamiento del sistema conjunto. En este trabajo se pretende corregir las deficiencias de los sistemas de vigilancia convencional a través de la implementación de uno alternativo. Para esto, es necesario evaluar el desempeño del sistema propuesto por medio de la medición e interpretación de sus parámetros de rendimiento. De un lado, se evaluará el desempeño del sistema mecatrónico mediante el análisis de la velocidad de traslación de la puerta y el torque del motor DC. Por otro lado, se evaluará el rendimiento de la RIC a través del análisis de su latencia, cantidad de paquetes perdidos, intensidad de señal recibida (RSSI) y consumo de energía.

1.3. Objetivos

1.3.1. Objetivo general

Implementar una red inalámbrica de cámaras que integrada a un sistema mecatrónico permite el monitoreo de entornos residenciales y el manejo remoto de puertas usando MQTT.

1.3.2. Objetivos específicos

- Implementar una red inalámbrica de cámaras con nodos reales para monitorear entornos residenciales.
- Diseñar un sistema mecatrónico que maneje remotamente la apertura y cierre de puertas a escala, y posteriormente, extrapolarlo a escala real por medio de una simulación en SolidWorks y Matlab.
- Integrar la red inalámbrica de cámaras y el sistema mecatrónico para el manejo remoto de puertas usando MQTT.
- Reconocer el rostro de los transeúntes con el fin de generar una alarma para apoyar la labor del vigilante.
- Evaluar la solución propuesta midiendo la velocidad de traslación de la puerta, el torque del motor, la latencia, la cantidad de paquetes perdidos, la intensidad de señal recibida (RSSI) y el consumo de energía.

1.4. Alcance

En el marco de la carrera de estudio (Ingeniería Mecatrónica), se considera el componente electrónico, que junto con la programación, puede satisfacer los estudios y análisis propuestos para la red inalámbrica de cámaras. No obstante, el componente mecánico será implementado en el sistema mecatrónico, que incluye el funcionamiento de motores DC que manejen la apertura y cierre de puertas.

La RIC se realizará en un entorno residencial, lo que significa que no se evaluarán otro tipo de espacios en las que la implementación y el comportamiento de este sistema sería completamente diferente, como por ejemplo, zonas externas como bosques, calles, campos abiertos, entre otros.

La residencia de estudio escogida en este caso es la del autor principal, debido a que, por la situación actual, la movilización a otros sitios está altamente restringida; además, porque cumple con los requisitos para llevar a cabo el proyecto: Facilidad de obtención de los datos planteados, un espacio propicio para la construcción y montaje de la RIC, e incluso la emulación de usuario final del sistema implementado.

Por limitaciones en el presupuesto, la red inalámbrica de cámaras contará con 4 módulos ESP32-CAM ya que estos, si bien son los sistemas embebidos con propiedades más acordes a las necesidades del proyecto, también son los componentes de valor más elevado, como se observa en la tabla de presupuesto más adelante.

2. Capítulo 2. Marco teórico

Cada aspecto que abarca la realización de este proyecto, desde su concepción hasta su aplicación, parte de conceptos teóricos que permiten el entendimiento de cada elemento, método y herramienta utilizados. En este capítulo se definirán los conceptos necesarios para comprender los principios en la implementación de una red inalámbrica de cámaras para el monitoreo de entornos residenciales y el manejo remoto de puertas usando MQTT.

De esta manera, el componente teórico del sistema propuesto se divide en tres conceptos clave: Sistema mecatrónico, red inalámbrica de cámaras y MQTT.

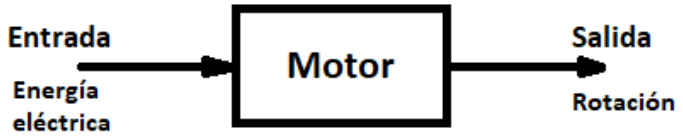
2.1. Sistema mecatrónico

Se puede definir un sistema mecatrónico en base a los dos términos a los que hace referencia: sistema y mecatrónica.

2.1.1. Sistema

En ingeniería, un sistema se puede interpretar como un conjunto de componentes o dispositivos conectados entre sí, que cumplen la función de realizar un proceso. Este proceso consiste en la transformación de una señal de entrada, y generar como resultado una señal de salida.

Figura 2.1. Ejemplo de un sistema.



Adaptado de [11]

La figura 2-1 muestra un diagrama de bloques de un sistema que consiste en un motor, el cual transforma la señal de entrada (Energía eléctrica), mediante un proceso (inducción electromagnética), y generando una señal de salida (Energía mecánica – rotación del motor). [11]

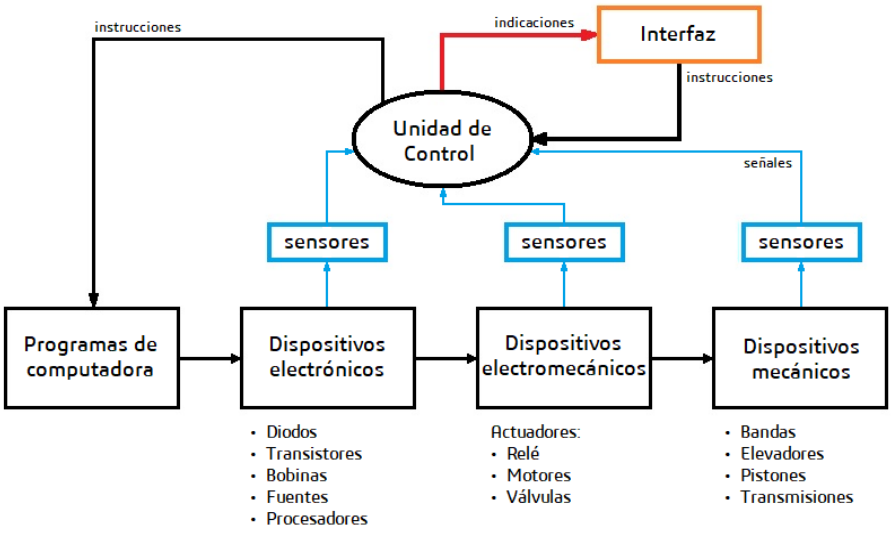
2.1.2. Mecatrónica

El término mecatrónica hace referencia a un campo de la ingeniería que abarca diferentes disciplinas, tales como la ingeniería eléctrica, mecánica, de sistemas y de control. Su función principal es aplicar todos estos conocimientos y con ellos crear productos que integren elementos eléctricos y mecánicos, mediante la programación de dispositivos de control.

La ingeniería mecatrónica se encarga de resolver problemas relacionados con el diseño y disposición de circuitos que pueden ser análogos o digitales, mecanismos formados por elementos mecánicos, sistemas fundamentados en microcontroladores (sistemas embebidos, por ejemplo), así como sensores y actuadores. [12]

De esta forma, un sistema mecatrónico se puede expresar como la unión de los componentes presentados en la Figura 2.2:

Figura 2.2. Componentes de un sistema mecatrónico.



Adaptado de [13]

La Figura 2.2 incluye además el tipo de elementos que se pueden utilizar para cada uno de los bloques que conforman un sistema mecatrónico. También se observa la forma en que los componentes deben ir conectados, evidenciando una retroalimentación del sistema en conjunto.

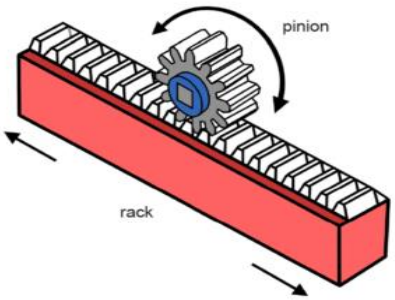
2.1.3. Componentes del sistema mecatrónico

Esta sección especifica las funciones y características de cada uno de los bloques en la figura. Se establecen componentes a las siguientes categorías: Sistema mecánico, condicionamiento e interfaz de la señal de entrada, arquitecturas de control digital, condicionamiento e interfaz de señal de salida y pantallas gráficas. Para cada uno de ellos, se enfatizará en los componentes utilizados en el presente proyecto.

2.1.3.1. **Dispositivos mecánicos**

También llamados mecanismos, son aquellos con la capacidad de convertir movimiento. Un sistema mecánico cambia la forma en que se genera un movimiento, a otro tipo con características diferentes [14]. Por ejemplo, el mecanismo piñón-cremallera transforma un movimiento rotacional (a partir del piñón) en uno traslacional (con la cremallera) y viceversa, como se muestra en la Figura 2.3:

Figura 2.3. Mecanismo piñón-cremallera.



Tomado de [15]

En este mecanismo, las piezas se acoplan por medio de dientes que poseen en su superficie, estos encajan entre sí realizando un esfuerzo de empuje y de esta manera

transmiten el movimiento. La relación de movimiento rotacional y traslacional se describe con las siguientes ecuaciones de desplazamiento y de velocidad:

- Desplazamiento de cremallera:

$$\Delta R_{cremallera} = r(\Delta\theta) = \frac{(d_{piñón})(\Delta\theta_{piñón})}{2} \quad (2.1)$$

Donde:

r y $d_{piñón}$ son el radio y el diámetro del piñón, respectivamente y

$\Delta\theta_{piñón}$ es la distancia de rotación del piñón, en radianes.

- Velocidad lineal de cremallera:

$$v_{cremallera} = \omega_{piñón} r_{piñón} = \frac{(d_{piñón})(\omega_{piñón})}{2} \quad (2.2)$$

Donde:

$\omega_{piñón}$ es la velocidad angular del piñón.

Así mismo, existe una entrada y una salida de movimiento, que depende de la pieza acoplada a una fuerza motriz. En el caso del sistema mecatrónico propuesto se genera una fuerza electromotriz a partir de un motor DC, cuyo eje se mueve de forma rotacional. Este eje se acopla al piñón que lleva este mismo tipo de movimiento, dejando así al piñón como entrada de movimiento, como elemento conductor. De esta forma, la salida de movimiento es la cremallera como elemento conducido, que fijada a la puerta lleva el movimiento traslacional al desplazarla de forma horizontal [16].

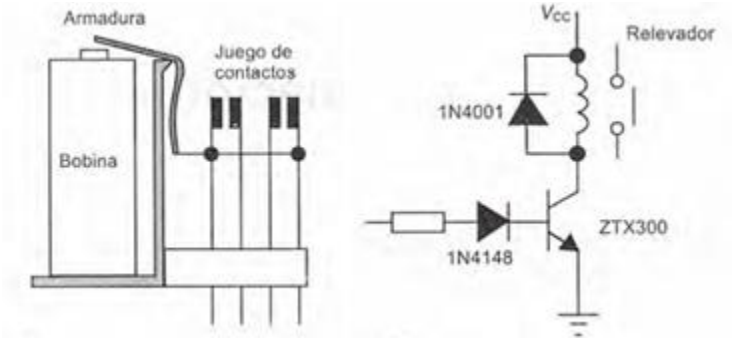
2.1.3.2. **Dispositivos electromecánicos**

En un sistema mecatrónico, los dispositivos electromecánicos se desempeñan como actuadores que operan a partir de las indicaciones dadas por el sistema de control. En este sentido, se dividen en dos tipos: dispositivos de conmutación y sistemas motrices.

- **Dispositivos de conmutación:** Que por lo general son interruptores, son elementos que se utilizan ya sea para recibir señales de entrada (sensores) o también para encender o apagar componentes eléctricos (actuadores). Esta última función es la que

se tiene en cuenta al incorporar estos dispositivos en sistemas mecatrónicos. Un ejemplo común es el relevador o relé, que es un interruptor manejado eléctricamente y cuya estructura se muestra en la Figura 2.4:

Figura 2.4. Relevador: Forma física (izq.) y esquemático (der.).



Tomado de [14]

- **Sistemas motrices:** Que se clasifican principalmente en motores de corriente directa (CD) y de corriente alterna (CA), son elementos que generan movimiento rotacional al aplicar sobre ellos una corriente. Por este motivo, son aquellos manejados por los sistemas de control para fines de manejo de posición y velocidad [14]. En la Figura 2.5, se observa un motor de CD, el cuál será utilizado en el proyecto para transformar una señal eléctrica proveniente de un controlador, en movimiento rotacional.

Figura 2.5. Motor DC.

Tomado de [17]

2.1.3.3. Dispositivos electrónicos

Los dispositivos electrónicos son la combinación de diferentes elementos electrónicos que se conectan entre sí para formar lo que se denomina como un circuito. Así mismo, son

considerados elementos electrónicos aquellos cuya fabricación resulta de la creación de interfaces de tamaño microscópico, con la utilización de materiales conocidos como semiconductores [14]. Un ejemplo de un dispositivo electrónico es el módulo L298N, que incluyendo diferentes componentes basados en semiconductores es capaz de controlar el movimiento de un motor de CD. La figura 2-6 muestra la forma de este dispositivo, que también es aplicado en el sistema mecatrónico propuesto:

Figura 2.6. Módulo L298N.

Tomado de [18]

2.1.3.4. Unidad de control

La unidad de control, como su nombre lo dice, es aquella encargada de controlar o manejar de forma inteligente las operaciones del sistema al cual esté conectado, en base al número de entradas y salidas que tenga. Por lo general, los elementos con capacidad de realizar tareas de este tipo son los microcontroladores, que son circuitos integrados complejos basados en circuitos digitales, permitiendo así su programación por medio de un software incorporado en el mismo [14]. En el presente proyecto, se maneja el módulo ESP32 y ESP32-CAM (Figura 2.7), ambos microcontroladores que permiten el manejo de señales de entrada y salida, siendo estas inalámbricas y eléctricas, respectivamente.

Figura 2.7. Módulo ESP32 (izq) y Módulo ESP32-CAM (der).

Tomado de [19]

2.1.3.5. Programación de microcontroladores

La programación de estos dispositivos se realiza por medio de herramientas como simuladores o emuladores, que para el caso de los microcontroladores, es un programa o aplicación que permite editar sus propiedades internas tales como el estado de las entradas y salidas, datos en la memoria, entre otros. Este proceso se realiza escribiendo software y compilándolo por lo general en un ordenador, para luego ser descargado en la memoria del microcontrolador en un lenguaje conocido como código de máquina.

Al escribir software, se emplean diferentes comandos cuya estructura depende de la herramienta de programación utilizada. De esta forma, se facilita la interacción entre el programador y el dispositivo.

2.2. Red inalámbrica de cámaras

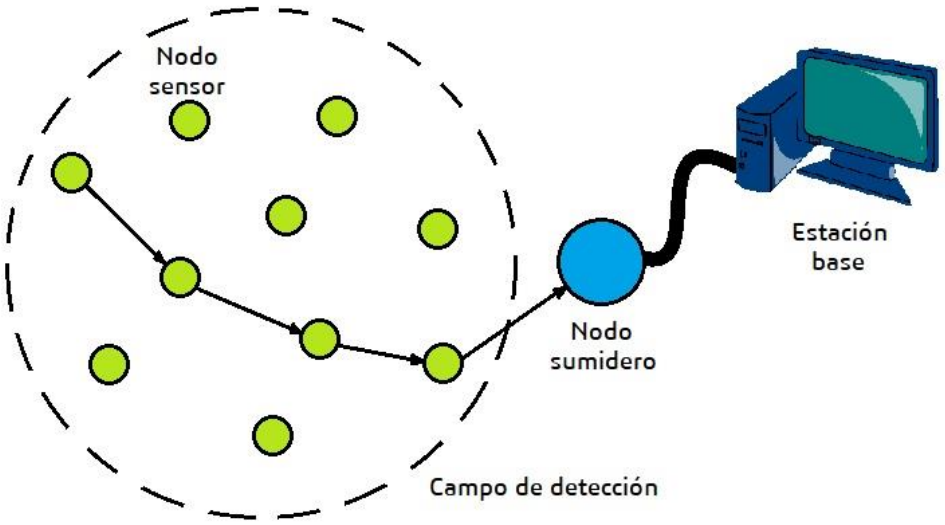
Una red inalámbrica de cámaras consiste en la implementación de las denominadas redes inalámbricas de sensores aplicadas a sistemas de vigilancia, más concretamente vigilancia por grabación de video que utiliza cámaras como dispositivos sensores [20]. La relación entre estos dos sistemas, RIS y sistemas de vigilancia por video, es complementaria y cada una puede ser desarrollada por separado; sin embargo, su aplicación en conjunto resulta en un sistema enfocado al monitoreo con cámaras, con cualidades de una red inalámbrica de sensores. A continuación, se definen los dos sistemas mencionados con anterioridad.

2.2.1. Red inalámbrica de sensores

Se define a una red inalámbrica de sensores (RIS abreviado) como un conjunto de sensores que monitorean un espacio físico amplio, y que se conectan entre sí de forma inalámbrica. Sus aplicaciones se basan principalmente en la medición de magnitudes físicas presentes en el entorno físico, como humedad, presión, intensidad lumínica, temperatura, entre otros [21].

Una RIS se compone por nodos, que dependiendo de su disposición pueden ser de detección o tipo sumidero. Estos nodos se ubican en un área extensa, denominada campo de detección. La información es enviada desde los nodos a una estación base que recolecta toda la información de la RIS, y la dispone a un usuario según la configuración de la red. Estos elementos se distribuyen como se presenta en la Figura 2.8.

Figura 2.8. Arquitectura de una RIS.



Adaptado de [21]

- **Nodos:** Este término corresponde a cada uno de los sensores en la RIS. Existen dos tipos de nodos: Los nodos sensores, que son aquellos que reciben la información del entorno y se conectan entre sí para formar una topología en la Red; y los nodos sumidero, que además de cumplir las funciones de los anteriores, también se emplean

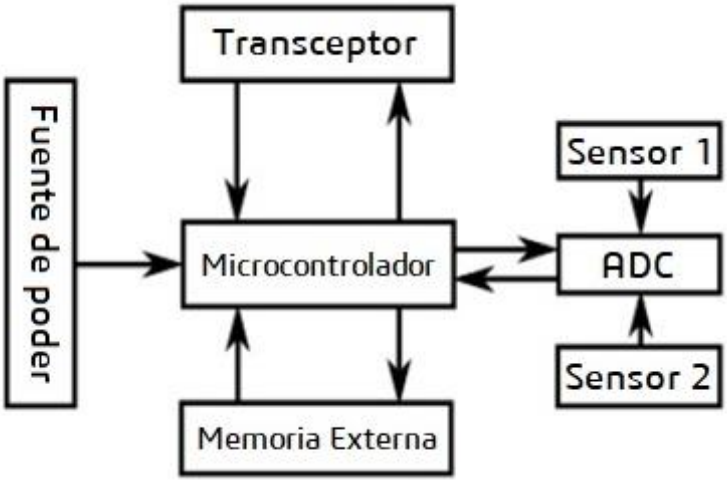
como el punto de llegada de toda la información proveniente de otros nodos, para ser enviada a la estación base [21].

Un nodo se compone de cuatro componentes básicos:

- Unidad de detección, que a su vez se compone de un sensor y de un convertidor análogo a digital (ADC)
- Unidad de procesamiento, que permite la comunicación y ejecutar las funciones del nodo.
- Unidad transceptora, que se encarga de la transmisión y recepción de la información.
- Unidad de alimentación, con la que los componentes del nodo son energizados.

Estos componentes pueden representarse como un diagrama de bloques, tal como se muestra en la Figura 2.9:

Figura 2.9. Diagrama de bloques de un nodo en una RIS.



Adaptado de [21]

- **Campo de detección:** Consiste en el espacio destinado a la detección por parte de la red inalámbrica de sensores. Este espacio presenta diversas condiciones ambientales y geográficas, que condicionan características de la RIS como son la ubicación de sus

nodos, el parámetro a detectar por los sensores, entre otros [21]. Por ejemplo, el presente proyecto maneja como campo de detección un entorno residencial y el parámetro de detección es la imagen, con cámaras como elemento sensor.

- **Estación base:** La estación base es el elemento que tiene como función principal la recolección de toda la información proveniente de los sensores que componen una red inalámbrica de sensores, aunque también puede encargarse de visualización de la información o su análisis. Por lo general, una estación base es un dispositivo que cuenta con una mayor capacidad computacional y capacidad de memoria que un nodo de la RIS, como un microcontrolador de alta capacidad o un computador mismo.

  Así como los nodos, la estación base se compone de diferentes elementos, que son:

  - Ordenador anfitrión. Puede ser un computador personal, que cuenta con las especificaciones mencionadas con anterioridad. Corresponde al hardware de la estación base.
  - Software de la estación base. Se puede definir como un programa en el ordenador anfitrión, que permite disponer de la información de la red de sensores para diferentes aplicaciones como almacenamiento, envío y transmisión, entre otros.
  - Un transceptor o nodo de puerta de enlace. A diferencia de un nodo sensor, este nodo no tiene la función de detección, más bien cumple la función de ofrecer una interfaz entre la red de sensores y la estación base [22].

2.2.2. Sistema de video vigilancia

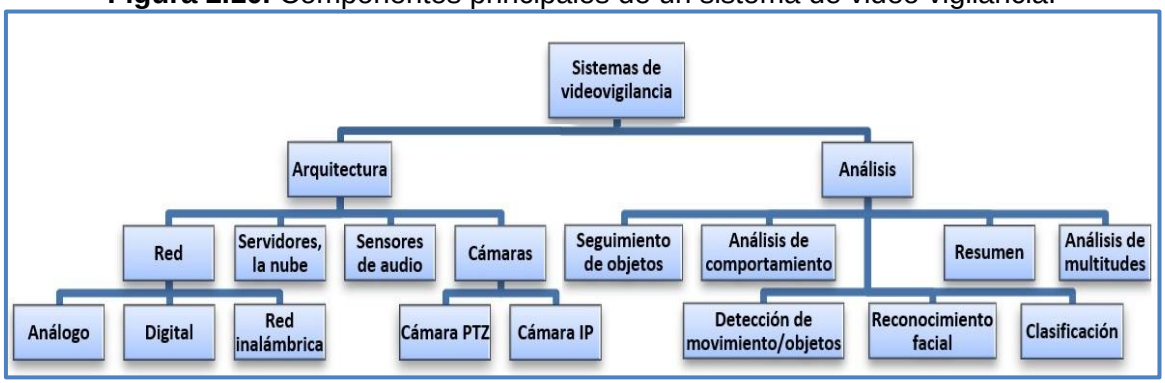
Un sistema de video vigilancia se describe como el conjunto de elementos necesarios para realizar el monitoreo de espacios, que pueden ser públicos o privados, haciendo uso de cámaras. Su aplicación tiene el objetivo principal de garantizar la seguridad a través de la reacción o la predicción de eventos posibles en el espacio monitoreado [23].

En sus inicios, los sistemas de video vigilancia se basaban únicamente en la detección de una señal (imagen) por medio de cámaras, que era transmitida por cables hacia un monitor donde es mostrada, o también hacia un dispositivo con capacidad de almacenamiento de

información. En la actualidad, estos sistemas han evolucionado al punto en que cuentan con la capacidad de analizar y automatizar toda la información obtenida por las cámaras, gracias a la adición de elementos y modernización de aquellos que los conforman.

Dadas las condiciones de operación y aplicaciones de un sistema de videovigilancia, este se puede describir desde dos aspectos importantes: La arquitectura del mismo, que corresponde principalmente al hardware o elementos físicos que lo estructuran; y la capacidad de análisis que posee, haciendo referencia a su capacidad de procesamiento [23]. La Figura 2.10 presenta de forma más detallada estas características en un sistema de video vigilancia:

Figura 2.10. Componentes principales de un sistema de video vigilancia.



Adaptado de [23]

2.2.2.1. Arquitectura de los sistemas de video vigilancia

De acuerdo a la función principal de un sistema de video vigilancia, se han creado diferentes métodos que pueden cumplir esta función según el tipo de elementos empleados y su disposición en el sistema, que pueden ser clasificados en tres tipos: Sistema análogo, digital, o por red.

- **Sistema de vigilancia análogo:** Siendo los primeros en ser implementados, este tipo de sistemas han sido utilizados durante más de 20 años. Emplea la tecnología de procesamiento de señales análogas como modelo base para la transmisión, intercambio y grabación de imagen; con elementos de envío de información como cables coaxiales y fibra óptica.

- **Sistema de vigilancia digital:** En un sistema de videovigilancia digital las señales de entrada también son análogas, sin embargo, sufren una conversión a señales digitales las cuales serán aquellas procesadas por los elementos que componen este sistema. Teniendo en cuenta lo anterior, la diferencia con respecto a un sistema de vigilancia análogo es la forma en que las señales se trasmiten, se controlan y se almacenan.

- **Sistema de vigilancia por red:** En este caso, el sistema está basado en el procesamiento digital de señales, y se diferencia a un sistema de vigilancia digital en que incluye además elementos como cámaras de red o cámaras IP, que son digitales. En un sistema por red, se puede realizar funciones de transmisión de señales, control y almacenamiento de video, e incluso el manejo de cada uno de los elementos que componen la red (cámaras y sensores) [23].

2.2.2.2. Análisis en los sistemas de video vigilancia

Otro aspecto importante de los sistemas de video vigilancia además de su arquitectura, es su capacidad de observar, detectar y clasificar los escenarios en un espacio monitoreado. Existen tareas que cumplen con estas características, brindando recursos que también son empleados en el análisis de información. Por ejemplo, tareas que destacan son: detección de movimiento, clasificación y reconocimiento de objetos, seguimiento de objetos, reconocimiento de la acción humana, detección y reconocimiento facial, conteo de multitudes, re identificación de personas, entre otras.

La implementación de cada una de las aplicaciones mencionadas depende de factores como el objeto de interés en el espacio monitoreado, condiciones de entorno alrededor de los objetos como forma, color, etc.; el movimiento del objeto, que incluye su velocidad y trayectoria, y otros factores que en general condicionan la detección por cámaras [23].

2.3. MQTT

MQTT es un protocolo de transporte de mensajes que se basa en la publicación/subscripción de mensajes entre clientes y servidor. Se caracteriza por ser ligero, tener una mejor transferencia de información en comparación con otros protocolos

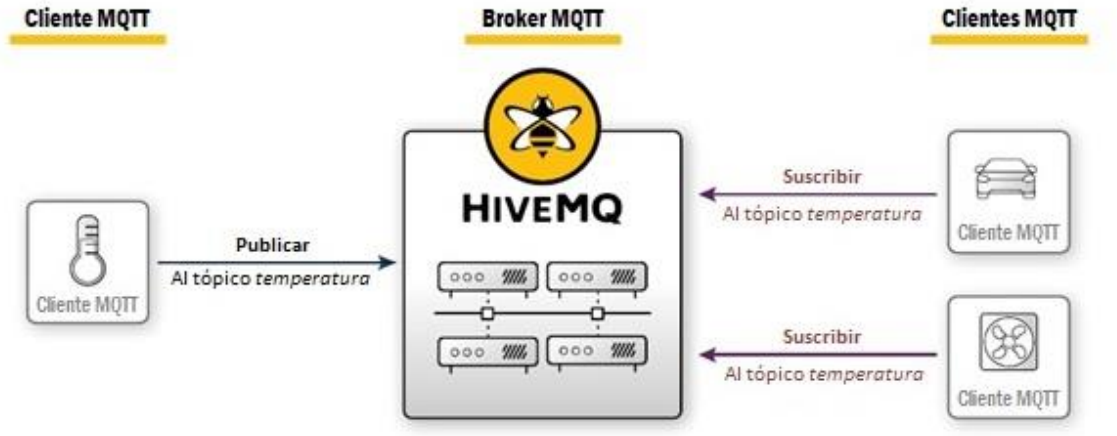
de comunicación (HTTP, por ejemplo) y estar diseñado de tal forma que su implementación sea sencilla. Esto permite su aplicación en entornos como la comunicación máquina a máquina (M2M) e incluso en el internet de las cosas (IoT). Esto último es la razón para implementar este protocolo en el presente proyecto.

El acrónimo MQTT hace referencia a Transporte de Telemetría MQ (MQ Telemetry Transport, en inglés), donde MQ corresponde a la serie de un producto desarrollado por IBM específicamente para soportar el uso de este protocolo. Por otro lado, este acrónimo puede emplearse directamente como el nombre del mismo [24].

2.3.1. Modelo Publicación/Suscripción del MQTT

Este modelo, también denominado pub/sub, es una alternativa al modelo cliente-servidor, donde el cliente se comunica directamente con su destino final. En su lugar, la arquitectura pub/sub desacopla el cliente que emite el mensaje (publicador) con el cliente o clientes que lo reciben (suscriptores), por lo tanto, nunca tienen comunicación directa entre ellos. De esta manera, se incluye un tercer componente llamado bróker, que se encarga de gestionar los mensajes recibidos de los publicadores y distribuirlos de manera adecuada a los suscriptores. La Figura 2.11 muestra un ejemplo de la aplicación de este modelo:

Figura 2.11. Arquitectura pub/sub del MQTT.



Adaptado de [24]

2.3.1.1. Filtrado de mensajes

Una característica importante del modelo pub/sub es la capacidad de filtrar los mensajes y controlar su envío o recepción a los diferentes clientes. En este caso, el componente encargado de realizar dicha tarea es el bróker, que administra los mensajes de tal modo que los clientes suscritos reciban los mensajes que les correspondan, según su conveniencia [24]. Existen tres formas en el que el bróker filtra los mensajes:

- **Filtro basado en el tema o tópico:** Es común que el envío de un mensaje tenga un asunto o tema, que en el modelo pub/sub se conoce como tópico. Los tópicos sin embargo no son líneas de texto, sino más bien cadenas con una estructura jerárquica definida. Este filtro toma el tópico del mensaje como criterio de suscripción por parte de los clientes.

- **Filtro basado en el contenido:** El filtrado por contenido se enfoca en lo que sería el cuerpo del mensaje, donde el criterio de suscripción corresponde a un grado de coincidencia con respecto al contenido del mensaje. Este tipo de filtro tiene la desventaja de que el cliente debe saber el contenido el mensaje con anterioridad, por lo que este último no puede ser encriptado o cambiado con facilidad.

- **Filtro basado en el tipo:** Este filtro hace referencia al tipo o clase de mensaje. El bróker maneja en estos casos tipos de mensaje como exception o sub-type, y sólo es aplicado cuando se usan lenguajes de programación orientados a objetos.

2.3.2. Elementos que conforman MQTT

Son tres los elementos que hacen parte de la transmisión de mensajes con MQTT: clientes, bróker y el establecimiento de la conexión entre ellos. A continuación, se describe con mayor detalle cada uno de ellos:

2.3.2.1. Cliente MQTT

Un cliente en MQTT puede ser cualquier dispositivo que sea capaz de correr una librería MQTT y de conectarse a un MQTT bróker en una red. La variedad de dispositivos aptos

para ser clientes puede ir desde algo pequeño como un microcontrolador hasta un servidor completo. Dependiendo de las tareas y funciones que les sean asignadas, pueden ser clientes publicadores o clientes suscriptores: los publicadores son aquellos que emiten un mensaje y los suscriptores los que los reciben [24].

Sin embargo, existen condiciones para que un dispositivo funciones como cliente MQTT. La primera es que debe funcionar bajo el modelo TCP/IP, que tiene cualquier máquina capaz de comunicarse en la red; la segunda es que debe entender un lenguaje de programación como Android, Arduino, C, C++, Java, entre otros, para correr la librería de cliente MQTT.

Entre las aplicaciones que puede tener un cliente MQTT se pueden mencionar:

- Puesta a prueba de un servidor MQTT en un ordenador, con capacidad de visualización para analizar su comportamiento.
- Conexión inalámbrica de un dispositivo con recursos limitados y una librería de mínimo tamaño.

2.3.2.2. Bróker MQTT

El bróker MQTT es un servidor que puede ser montado en un dispositivo con conexión a la red. Es el núcleo del protocolo pub/sub, dado que es el encargado de ejecutar prácticamente todas las funciones que involucran el empleo del mismo: recibe todos los mensajes, los filtra, decide qué cliente se suscribe y se encarga de enviarles los mensajes. Además, el MQTT bróker puede autenticar y autorizar los clientes para publicar o suscribir mensajes.

Para que un bróker pueda funcionar correctamente, tiene que cumplir los siguientes requisitos:

- **Debe ser altamente escalable.** Al incrementar la cantidad de clientes que maneja el bróker MQTT la carga en el mismo aumenta de forma significativa. La escalabilidad asegura que el bróker tenga un rendimiento óptimo sin importar la cantidad de clientes conectados.

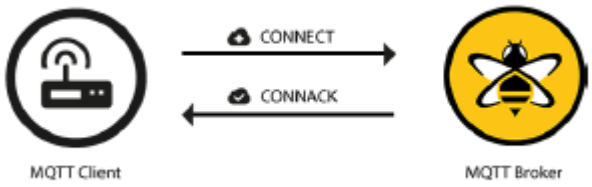
- **Fácil de monitorear.** En este caso, el uso de un computador permite una mayor interacción con el bróker, facultando al usuario de configurarlo y visualizar de forma más detallada todos los procesos que se están llevando a cabo.
- **Resistente a fallos.** Como cualquier sistema de comunicaciones, es necesario que el bróker MQTT tenga la capacidad de seguir funcionando a pesar de presentarse fallos o errores en alguno de sus componentes.

2.3.2.3. Conexión MQTT

Para que exista una conexión entre un cliente y el bróker se requiere que ambos cuenten con la pila TCP/IP, que es un paquete de protocolos de comunicación utilizado ampliamente. Este requisito se debe a que el protocolo MQTT está basado en el TCP/IP.

Cuando se va a iniciar una conexión, el cliente envía un mensaje de conexión denominado CONNECT al bróker. El bróker le responde enviando un mensaje de reconocimiento conocido como CONNACK y un código de estado. La conexión entre estos componentes, una vez iniciada, se mantiene hasta que el cliente envía un mensaje de desconexión o también hasta que la conexión falla. La Figura 2.12 representa un el proceso de conexión entre el cliente y el bróker:

Figura 2.12. Proceso de conexión entre el cliente y el bróker con MQTT.



Tomado de [24]

2.3.3. Publicación, suscripción y cancelar suscripción en MQTT

En el modelo pub/sub, las acciones principales que realizan los clientes son precisamente las de publicar un mensaje o suscribirse a él; sin embargo, también se puede cancelar la

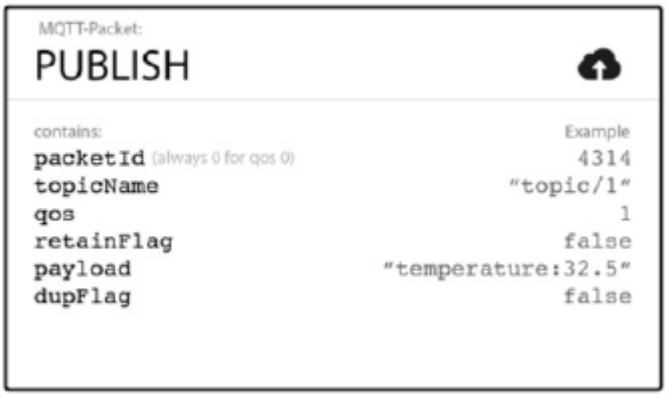
suscripción para dejar de recibir mensajes [24]. A continuación, se describe con más detalle en qué consiste cada una de las acciones mencionadas con anterioridad.

2.3.3.1. Publicar

Para que un cliente pueda funcionar como publicador, primero debe estar conectado al bróker y luego emitir un mensaje de tipo PUBLISH que debe tener un tópico para que así el bróker pueda establecer filtros a los clientes suscriptores. El publicador decide el tipo de mensaje que va a transmitir, que puede tener formatos como datos binarios, de texto e incluso archivos XML o JSON.

Un mensaje PUBLISH tiene diferentes atributos, que se pueden ver en la Figura 2.13.

Figura 2.13. Tarjeta con atributos de un mensaje PUBLISH (a la izquierda).



Tomado de [24]

- **Packet ID:** Corresponde al identificador del mensaje que es transmitido entre los clientes y el bróker. Generalmente el identificador es un número (4314 en la figura) y tanto el bróker como la librería del cliente son los encargados de asignarlo al mensaje.

- **Topic Name:** Es el nombre con el que se designa el tópico del mensaje. Consiste de una cadena jerárquicamente estructurada con barras diagonales (/) como separadores. En la figura 2-16 aparece como "*topic/1*".

- **QoS:** Es un número que indica la Calidad de Servicio (Quality of Service en inglés) del mensaje. Puede tener tres valores: 0,1 o 2; entre menor sea su valor habrá mayor garantía de que el mensaje llegue a su destino.

- **Retain Flag:** Es una bandera (también llamada etiqueta) que define si el mensaje es almacenado en el bróker con un tópico determinado, lo que significa que cualquier cliente que se suscriba a ese tópico recibirá el mensaje retenido por el bróker.

- **Pay Load:** Es el contenido del mensaje. Ya que el protocolo MQTT es independiente de los datos que maneja, se puede enviar imágenes, texto de cualquier tipo de codificación, todo tipo de datos en binario, entre otros.

- **Dup Flag:** Es una bandera que indica si el mensaje es original o un duplicado; este último sucede cuando el mensaje original no llega al recipiente de destino (ya sea bróker o cliente) y requiere de un reenvío, como copia del mensaje inicial.

2.3.3.2. Suscribir

Es importante que existan clientes que deseen suscribirse a mensajes para así completar esa cadena de comunicación entre clientes a través del bróker MQTT. Para que un cliente pueda suscribirse a un mensaje, necesita enviar un mensaje tipo SUBSCRIBE al bróker. A diferencia de un mensaje PUBLISH, este tipo de mensaje sólo cuenta con dos atributos que son:

- **Packet ID:** De igual forma que con el mensaje publicado, es un número que identifica el mensaje en concreto. Así mismo, la asignación de este identificador está a cargo del bróker o del cliente en cuestión.

- **Lista de suscripciones:** Un cliente puede contar con múltiples suscripciones en un solo mensaje de tipo SUBSCRIBE. Por lo tanto, se crea una lista con todas las suscripciones disponibles, donde cada una tiene un tópico y un nivel de Calidad de Servicio QoS.

Una vez el cliente haya enviado el mensaje SUBSCRIBE al bróker, este tiene que confirmar la suscripción enviando de vuelta un mensaje denominado SUBACK, con el que el cliente reconoce su suscripción a un tópico determinado. Este mensaje de confirmación tiene dos atributos también: El Packet ID, que es el mismo del mensaje SUBSCRIBE, y una lista de códigos de retorno. La cantidad de códigos de retorno en el mensaje depende de la cantidad de suscripciones que haya escogido el cliente.

2.3.3.3. Cancelar la suscripción

Para que un cliente cancele una suscripción a un tópico, necesita seguir exactamente el mismo proceso que cuando se suscribió en primer lugar. La diferencia radica en el tipo de mensajes involucrados. En primer lugar, el cliente envía un mensaje llamado UNSUBSCRIBE, que llega al bróker y tiene los mismos atributos que el mensaje SUBSCRIBE.

Luego el bróker tiene que confirmar esa cancelación de suscripción enviando de vuelta otro mensaje denominado UNSUBACK. Sólo cuenta con un atributo (Packet ID, que es el mismo que tiene el mensaje UNSUBSCRIBE) y cuando llega al cliente, se confirma la desuscripción.

2.3.4. Tópicos y su estructura

Un tópico es de los atributos más importantes de un mensaje, ya que de este depende la recepción del mensaje por el bróker y su posterior clasificación para distribuirlos a los clientes correspondientes. Su estructura se compone de una cadena UTF-8 (Formato de Transformación Unicode de 8 bits) que posee una jerarquía de varios niveles. Un tópico puede tener uno o varios niveles. Cada nivel del tópico es separado por una barra diagonal (/) [24], como se muestra en la Figura 2.14:

Figura 2.14. Estructura de un tópico.

Adaptado de [24]

Los tópicos en MQTT se caracterizan por ser muy ligeros en tamaño, pueden ser aceptados por el bróker sin inicialización previa siempre y cuando sean válidos, y los clientes son los que actúan de acuerdo al tópico, no al contrario, al momento de tener que realizar una publicación o una suscripción.

2.3.4.1. Comodines

Un comodín es un carácter que incluido en un tópico permite a un cliente suscribirse a varios tópicos de manera simultánea; sin embargo, sólo funciona para clientes que deseen suscribirse, mas no aquellos que publican un mensaje. Existen dos comodines: de nivel simple y de nivel múltiple.

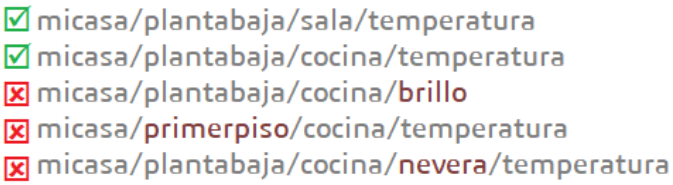
- **Nivel simple:** Se utiliza el símbolo más (+) para reemplazar un nivel de un tópico. La Figura 2.15 muestra un ejemplo de un tópico aplicando este comodín.

Figura 2.15. Tópico con comodín de nivel simple.

Adaptado de [24]

Un tópico con un comodín de nivel simple permite a un cliente suscribirse a todos los tópicos que se encuentren en el mismo nivel jerárquico del comodín. Para el caso del ejemplo anterior, un cliente se suscribirá a cualquier tópico que se encuentre en el lugar del símbolo +. La Figura 2.16 muestra diferentes casos donde puede aplicar el comodín:

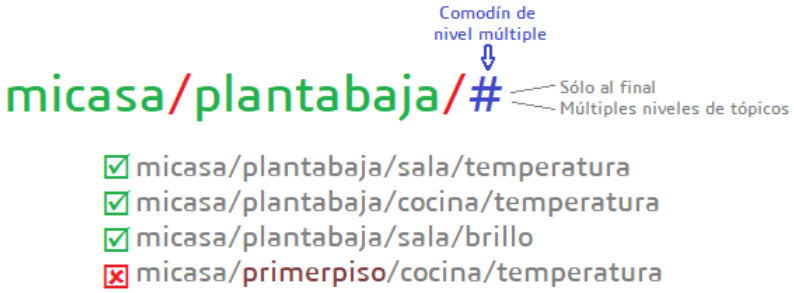
Figura 2.16. Resultados de una suscripción con el tópico de la Figura 2.15.



Adaptado de [24]

- **Nivel múltiple:** Este comodín utiliza el símbolo numeral (#) y cubre en este caso todos los niveles de jerarquía iguales e inferiores al nivel donde se aplique este comodín. Se requiere que este símbolo esté ubicado como último carácter en el tópico y que sea precedido por un símbolo (/). La Figura 2.17 muestra un ejemplo de aplicación y diferentes casos donde se observa la validez del comodín [24]:

Figura 2.17. Tópico con comodín de nivel múltiple y casos de aplicación.



Adaptado de [24]

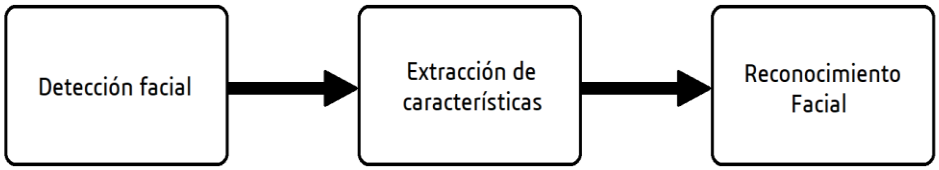
2.4. Reconocimiento Facial

El reconocimiento facial es un término que en términos generales se define a sí sólo: Consiste en la acción de reconocer el rostro por parte de un sistema. Sin embargo, los conceptos clave “reconocimiento” y “facial” tienen muchos aspectos que deben ser

considerados para entender a profundidad el empleo de esta tecnología en dispositivos principalmente orientados a la vigilancia, así como su funcionamiento.

En primer lugar, la acción de “reconocer” en un sistema con la función de reconocimiento facial tiene intrínsecamente varias tareas que siguen un orden determinado, y se dividen en tres principales: Detección o segmentación del rostro, extracción de características y finalmente el propio reconocimiento facial [25]. La Figura 2.18 representa en un diagrama de bloques el orden de estos procesos en un sistema convencional de reconocimiento facial.

Figura 2.18. Diagrama de bloques de un sistema de reconocimiento facial.



Adaptado de [25]

Por otra parte, se tiene el concepto “facial”. Esto hace referencia a que el rostro humano será el criterio de análisis en el sistema de reconocimiento. El rostro es uno de los rasgos que hace al ser humano único, con capacidad de mostrar información como edad, género, raza e incluso el estado mental en base a expresiones faciales, lo que lo convierte en un buen parámetro para procesos de identificación de personas. El rostro, junto con otros rasgos biométricos como las huellas dactilares y el iris ocular, permiten el monitoreo y control de personas en espacios determinados, con la diferencia de que el primero puede ser obtenido sin el consentimiento de la persona. Esto hace al reconocimiento facial adecuado para realizar labores de vigilancia [26].

2.4.1. Procesos presentes en el reconocimiento facial

De acuerdo a la figura anterior, existen tres procesos que componen en conjunto al reconocimiento facial. Estos se explican con mayor detalle, a continuación:

2.4.1.1. Detección de rostros

La detección de rostros implica una capacidad por parte del sistema de identificar si existe o no un rostro según la información que pueda obtener, la cual proviene de sensores como cámaras, principalmente. Además de la presencia de un rostro, la detección tiene que determinar también la posición del mismo. Existen dos categorías sobre las formas de lograr la detección de rostros [25]

- **Enfoque basado en características:** Este método se basa en la comparación de características faciales previamente obtenidas y las características faciales extraídas por detección. Algunas de estas características pueden ser los ojos, la boca, nariz o color de piel.

- **Enfoque basado en la imagen:** También conocido como método basado en la apariencia, este enfoque trata de emparejar la imagen obtenida con otra que se encuentra en una base de datos. El sistema de detección toma la información de la base de datos y crea una referencia para identificar un rostro en una imagen.

2.4.1.2. Extracción de características

Consiste en el proceso de conversión de una imagen que contenga un rostro, en un conjunto de características faciales que permitan distinguir a una persona en específico. Cuando el proceso de extracción de características es lo suficientemente bueno, no es necesaria toda la imagen sino más bien esas características extraídas que permitirán el reconocimiento facial. Existen tres métodos para la extracción de características [25].

- **Método genérico:** Es aquel que se basa en curvas, líneas y bordes.

- **Método basado en plantillas de características:** El cual se emplea para detectar características faciales como ojos, pestañas, boca, etc.

- **Método de emparejamiento estructural:** El cual tiene en cuenta restricciones geométricas en las características faciales.

2.4.1.3. Reconocimiento facial

Este último proceso consiste en el propio reconocimiento de un rostro con sus características, en distinción tanto del ambiente como de otros rostros. Existen tres métodos frecuentes para lograr el reconocimiento facial:

- **Método basado en el conocimiento:** Es aquel que se basa en el conocimiento humando sobre los rostros y sus características faciales para su reconocimiento. Estos conocimientos son convertidos en una serie de reglas que los dispositivos pueden aplicar, sin embargo, dada la complejidad de un rostro sólo es recomendado el uso de este método con entradas de información simples.

- **Método de características invariantes:** Ciertas características faciales no cambian con alteraciones en el ambiente como orientación del rostro, iluminación, entre otros; por lo que este método aprovecha estas características como criterio de reconocimiento facial. Estas características pueden ser la distancia ojo a ojo, ángulo de la quijada, tamaño de la nariz, etc.

- **Método basado en plantillas:** Este método se basa en la comparación de las imágenes de entrada con otras imágenes de referencia, para así reconocer rostros. Los rostros son convertidos en funciones, que son guardadas dando lugar a las plantillas de imágenes [25].

2.4.2. Características del reconocimiento facial

Existen tres características que definen todos los aspectos involucrados en el reconocimiento facial:

2.4.2.1. Taxonomía del reconocimiento facial

Un sistema de reconocimiento facial se clasifica en dos grupos principales: los que emplean métodos basados en imágenes y los basados en video. El primero realiza el reconocimiento a partir de la apariencia física, mientras que el segundo utiliza la variación en el tiempo de la apariencia física además de la dinámica del rostro [26].

2.4.2.2. Bases de datos

Desde que se empezó a emplear sistemas con capacidad de reconocimiento facial, se establecían bases de datos que consistían en un banco de imágenes que servían de referencia para comparación e identificación de rostros. Inicialmente se contaban con cientos de imágenes en las bases de datos, actualmente se pueden obtener millones, o miles de videos (para sistemas basados en video) [26].

2.4.2.3. Métricas de evaluación

Como cualquier sistema de detección, el sistema de reconocimiento facial requiere de métricas que permitan determinar su rendimiento en comparación con otros sistemas [26]. Algunas de estas métricas son:

- **Tasa de Coincidencias Falsas:** Abreviado FMR en inglés, es el porcentaje de muestras que resultan falsas con respecto a un total de detecciones.

- **Tasa de No Coincidencias Falsas:** Abreviado FNMR, es el porcentaje de muestras verdaderas que fueron descartadas de forma incorrecta.

- **Precisión:** Es el porcentaje de muestras correctamente clasificadas.

- **Tasa de Aceptación Verdadera:** GAR en inglés, es el porcentaje de muestras identificadas como rostros correctamente.

- **Tasa de Detección:** Es el porcentaje de muestras rechazadas correctamente por no ser rostros.

3. Capítulo 3. Planteamiento de la solución

En este capítulo se plantea el proceso a seguir para lograr cada uno de los objetivos planteados en el capítulo 1. Este proceso se divide en 5 fases, una por cada objetivo, con el fin de organizar el desarrollo del proyecto de acuerdo a las diferentes bases teóricas presentes en el proyecto.

3.1. Implementación de una red inalámbrica de cámaras con nodos reales en un entorno residencial.

Para implementar la RIC se establecieron ciertos requisitos para garantizar una operación adecuada en un entorno residencial: desde la disposición de la red en el espacio, las cámaras a utilizar, la conexión entre cada nodo de la red, la programación de los mismos para que funcionen de acuerdo a la solución propuesta y la puesta a prueba del sistema.

Estos requisitos delimitan los procedimientos necesarios para la implementación de una RIC y se ajustan al concepto y elementos de una Red Inalámbrica de Sensores mencionados en el capítulo 2.2.1. Partiendo de lo anterior, de bases teóricas y de investigación sobre una Red Inalámbrica de Cámaras, se plantea el siguiente proceso para la implementación de este sistema:

3.1.1. Definir el espacio de trabajo

Para comenzar a trabajar en esta fase se define un entorno residencial, que en este caso es la residencia del autor principal dadas las facilidades de acceso y condiciones de trabajo para el desarrollo del proyecto. Consiste en un área de 150 m^2 concentrada en un solo piso, con habitaciones y pasillos distribuidos.

Este espacio será la referencia para la distribución de los nodos de la red, que tendrá en cuenta los espacios disponibles, ubicaciones estratégicas y obstáculos como paredes o puertas. Este espacio será el campo de detección de la Red Inalámbrica de Cámaras.

3.1.2. Seleccionar los elementos que conforman la RIC

Los elementos que se consideran en esta fase de selección están basados en el esquema general mostrado de una RIS, en la Figura 2.8 del capítulo 2. A continuación se indican las especificaciones que deberían tener los elementos de la Red:

- **Nodos sensores:** Dispositivo con conexión inalámbrica vía Wifi, que posea una cámara como elemento de detección y que sea programable para aplicar las funciones especificadas en el planteamiento del problema del capítulo 1.
- **Nodo sumidero:** Además de tener las mismas especificaciones de los nodos sensores, este nodo debe contar con funciones que permitan la conexión al MQTT Bróker, con el que se integrará la RIC al sistema mecatrónico.
- **Punto de acceso:** Siendo el componente central de la red, debe ser capaz de realizar tareas de enrutamiento de datos y redirección de datos entre los elementos de la red y el MQTT broker, mediante conexión por Wifi.
- **MQTT Broker:** Debe ser un dispositivo capaz de recibir toda la información de los sensores, y a su vez que cuente con espacio de almacenamiento para implementar el MQTT Broker.

De acuerdo a lo anterior, se definen los componentes a utilizar:

3.1.2.1. Nodos sensores y nodo sumidero.

La selección de este componente consistió en evaluar diferentes sistemas embebidos y comparándolos según las propiedades de mayor relevancia para el proyecto. La Tabla 3.1 presenta la siguiente información para tres sistemas embebidos diferentes:

Tabla 3.1. Comparación de tres sistemas embebidos diferentes para selección de nodos en la RIC.

SISTEMA EMBEBIDO	COSTO	VELOCIDAD DE PROCESADOR	MEMORIA ROM	MEMORIA RAM	ADC	CÁMARA
Raspberry Pi 3	$ 168500	1.2 – 1.4 GHz	Flash 4 GB	1 GB	SI	NO
ESP32-CAM	$ 56000	Hasta 160 MHz	EEPROM 448 KB	520 KB	SI	SI
Arduino ATMega2560	$ 45000	16 MHz	Flash 256 KB	8 KB	SI	NO

Y de acuerdo a la información obtenida se selecciona el módulo ESP-32 CAM como nodo de la red, debido a su bajo costo en comparación con los demás sistemas embebidos (considerando la cámara incluida) y las aplicaciones que son suficientes para los fines del proyecto.

3.1.2.2. Punto de acceso.

Teniendo en cuenta la ubicación del desarrollo del proyecto, la única opción disponible es el módem residencial, sin embargo cumple con todas las condiciones para llevar a cabo la implementación de la RIC.

3.1.2.3. MQTT Broker.

El mejor dispositivo acondicionado para cumplir con las especificaciones es un computador. En este caso se selecciona directamente el computador portátil del autor principal, por fácil disponibilidad y tener aplicaciones más que suficientes para llevar a cabo las funciones de la Red Inalámbrica de Cámaras.

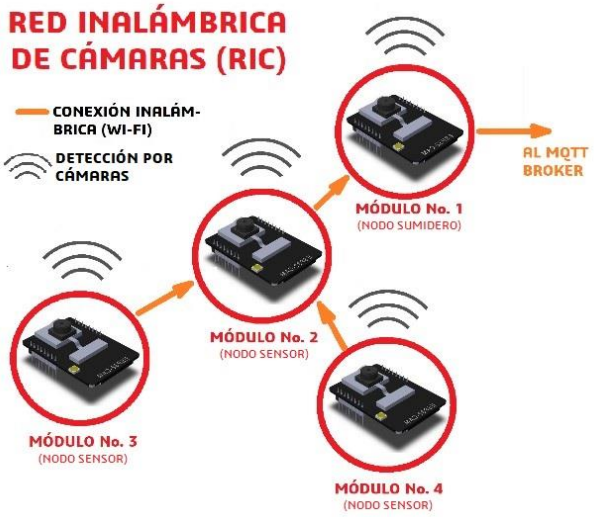
Se tiene en cuenta que el MQTT Broker es un protocolo de envío y recepción de mensajes que bajo el modelo publicar/suscribir, administra la información de los diferentes clientes (dispositivos conectados) recibiendo sus mensajes y enviándolos a otros clientes.

3.1.3. Interconexión de nodos en la red

Para que los nodos de la red puedan comunicarse entre sí es necesario que sean capaces de interconectarse de forma inalámbrica, esto se logra con la característica del módulo ESP32 – CAM de tener conectividad vía Wifi, sin embargo, hace falta una programación en estos dispositivos para que puedan cumplir con esta función.

Por otra parte, se define la topología de la red inalámbrica de cámaras, que será de tipo malla dadas las condiciones del espacio de trabajo (cantidad de nodos y distribución en el campo de detección). La Figura 3.1 muestra la interconexión de los nodos planteada para el entorno residencial seleccionado:

Figura 3.1. Distribución de la RIC siguiendo una topología de malla.



3.1.4. Programación de los módulos ESP32 – CAM

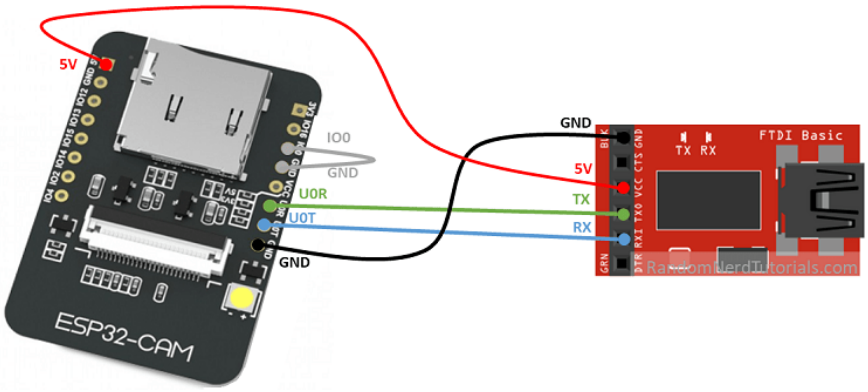
Los módulos deben ser programados para que desempeñen todas las funciones que se proponen en el capítulo 1.2:

- Detección de rostros a partir de la cámara como elemento sensor
- Comunicación entre módulos como nodos de la RIC
- Capacidad de envío de alarma al realizar una detección
- Integración con el MQTT Broker
- Cambio de modo de operación para optimizar consumo de energía

Sin embargo, para esta sección sólo se considera la comunicación entre los nodos de la red inalámbrica de sensores, mientras que las demás funciones corresponden a secciones posteriores. De este modo, se realiza el siguiente proceso:

- Inicialmente se toma uno de los módulos ESP32-CAM y se conecta a un computador, mediante un adaptador USB a TTL. El diagrama de conexiones correspondiente se muestra en la Figura 3.2:

Figura 3.2. Interconexión entre el módulo ESP32-CAM y el adaptador USB a TTL.



Tomado de [27]

- Una vez conectado el módulo, se emplea el software Arduino desde el computador para realizar la programación. Se toma como referencia el código en [28] para operar varios módulos ESP32 como una red en malla y se realizan modificaciones como incluir el modelo de la tarjeta empleada e introducir parámetros de red (Nombre de red y contraseña) en la conexión, con el fin de que el código sea adecuado para los módulos de trabajo ESP32-CAM. El código realizado se encuentra en el Anexo A, donde se observa el empleo de la librería painlessMesh, que es la que hace posible la conexión de diferentes módulos ESP como una red inalámbrica en topología de malla [29].

En secciones posteriores, la señal de entrada es un rostro detectado por la cámara de un nodo, comunicándose con otro nodo que puede ya sea reenviar la información o en caso de ser un nodo sumidero, enviar directamente el mensaje al MQTT Broker.

3.2. Diseño del sistema mecatrónico

El diseño del sistema mecatrónico se realiza tomando como eje principal la función que debe cumplir una vez esté construido: La apertura y cierre de puertas a escala de acuerdo a instrucciones dadas por la Red Inalámbrica de Cámaras. En este sentido, se plantea inicialmente un sistema que tiene una señal de entrada y una señal de salida, tal como se muestra en la Figura 3.3:

Figura 3.3. Diagrama de bloques con la Entrada y salida del Sistema Mecatrónico.

De esta manera, se establecen condiciones iniciales que sirven como punto de partida para definir todos los procesos que se tienen que seguir, desde la concepción del sistema planteado en la Figura 3.5, selección de los elementos que lo componen, el diseño de algunos de ellos, su disposición para implementación conjunta y finalmente la puesta en operación del sistema mecatrónico.

3.2.1. Selección de componentes

De acuerdo a la teoría en el numeral 2.1 del capítulo 2, un sistema mecatrónico debe tener los siguientes elementos:

- Elemento mecánico, que define la señal de salida en el sistema en forma de movimiento mecánico. Para este caso se toma un mecanismo piñón-cremallera, que por sus propiedades es adecuado para recibir el giro de un motor (movimiento rotacional) y convertirlo en desplazamiento de una puerta (movimiento traslacional). Se considera el diámetro, número de dientes y relación torque - velocidad del piñón como

criterios de selección de acuerdo a la disponibilidad en el mercado, realizando una comparación de ellos en la Tabla 3.2:

Tabla 3.2. Características de diferentes piñones para selección.

OPCIÓN	DIÁMETRO (mm)	NÚMERO DE DIENTES	RELACIÓN VELOCIDAD v – TORQUE τ
1	12	11	v muy alta – τ muy bajo
2	17	15	v alta – τ bajo
3	24	22	v – τ balanceado
4	35	33	v baja – τ alto
5	46	45	v muy baja – τ muy alta

Teniendo en cuenta que el desplazamiento establecido es pequeño (10 cm aproximadamente), el torque debe ser el mayor posible para manejar la carga de la puerta y que el motor escogido más adelante es de CD y por consiguiente presenta una alta velocidad angular, se selecciona el piñón de la opción 1, con las menores dimensiones y número de dientes y ofrece un torque de entrada considerable.

Por otra parte, la cremallera cuenta con dimensiones predeterminadas, de longitud de 8cm y un total de 26 dientes, con acople a otras cremalleras, por lo que en este caso se toman dos cremalleras acopladas, con el fin de tener una libertad de desplazamiento de 16 cm y con 52 dientes.

- Elemento electromecánico, que convierte la energía eléctrica en movimiento mecánico. En este caso, corresponde a un motor DC, y teniendo en cuenta la Tabla 3.3, el dispositivo más adecuado en este caso es un motor DC-12V, que tiene como entrada energía eléctrica y una salida de movimiento rotacional y se acopla al dispositivo electrónico:

Tabla 3.3. Tabla comparativa de diferentes motores eléctricos.

MOTOR	COSTO*	COMPATIBILIDAD	CARACTERÍSTICA ESPECIAL
DC 12V	$ 10000	SI	Sólo cuenta con 2 pines, gira de forma continua.
Paso a paso	$ 8600	SI	El movimiento se limita por etapas.
Servomotor	$ 8500	NO	Incluye sistema de control interno.

El motor seleccionado es de modelo es FC-280PC, con características que se observan en el Anexo B. En su punto máximo, cuenta con un torque de 42.2 mN*m, por lo que este valor será de referencia para especificar la carga máxima que puede presentar la puerta a desplazar.

- Elemento electrónico, que genere una salida de voltaje con variaciones que dependerán de la señal de entrada: el mensaje recibido del MQTT. Un elemento que cumple con esta condición es la tarjeta L298N, que recibe una señal eléctrica y logra acoplar el elemento electromecánico con características de movimiento y velocidad específicas. En la Tabla 3.4 se consideran características como precio y tamaño con respecto a otros elementos de este tipo, que llevan a la selección del L298N.

Tabla 3.4. Comparación de controladores de motor.

ELEMENTO	COSTO	TAMAÑO	INCLUYE
L9110	$ 8200	3 x 4 cm	Integrado más estructura con distribución de pines
L293D	$ 3000	1 x 2 cm	Sólo el circuito integrado
L298N	$ 9500	4 x 4 cm	Integrado más estructura con distribución de pines

- Elemento de entrada. Dadas condiciones como ser un sistema inalámbrico e integrarse a un servidor MQTT se considera el módulo ESP32, que tiene conectividad Wifi y posee pines de salida los cuales se pueden energizar a partir de la señal recibida. Además, se tiene en cuenta que ya se ha seleccionado un dispositivo de la misma familia (ESP32-CAM) para la Red Inalámbrica de Cámaras, por lo que se puede garantizar la compatibilidad entre los sistemas.

- Elemento de salida. Corresponde a la puerta que se va a desplazar, y cuyas propiedades físicas son determinantes para la operación del sistema mecatrónico. En este caso se establecen unas dimensiones de 10cm x 10cm x 1cm, siendo 1cm el espesor adecuado para fijar la puerta a la cremallera. De este modo, el material de la puerta será el factor de diseño y la característica a determinar.

- Fuente de alimentación. En el caso ideal, el sistema se energiza por una fuente independiente de baterías de larga duración, conectadas en serie para tener una salida de 12V. Sin embargo, al tratarse de un prototipo, se selecciona una fuente de alimentación variable fijando un voltaje de 12V.

Una vez escogidos todos estos elementos, se determina su conexión para que en conjunto puedan transformar la señal de entrada (instrucción del MQTT Broker) en una señal de salida deseable (movimiento mecánico traslacional). La Figura 3.4 muestra con detalle la conexión de los elementos del sistema mecatrónico:

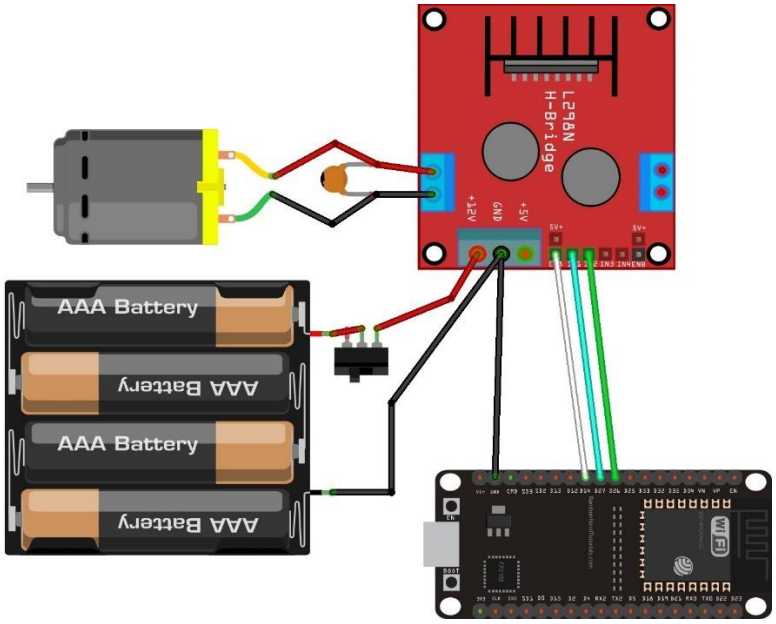
Figura 3.4. Conexión de elementos en el sistema mecatrónico.

3.2.2. Construcción del sistema

En este paso se tienen definidos todos los elementos del sistema mecatrónico, por lo que se procede a realizar la conexión entre ellos. Se toma como referencia el tutorial de [30]

donde se muestra el diagrama de conexiones del módulo ESP32, el controlador de motor L298N, el motor DC de 12V y así mismo la fuente de alimentación al sistema. Este diagrama se puede observar en la Figura 3.5:

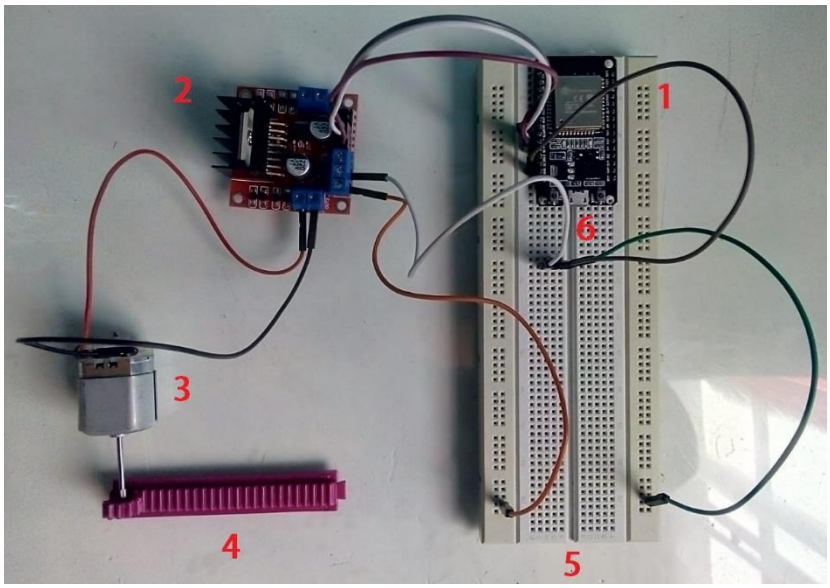
Figura 3.5. Diagrama de conexión de elementos con la ESP32.



Tomado de [30]

Sin embargo, dado que el armado inicial del sistema tiene el objetivo verificar su funcionamiento, se omite la conexión del switch y del condensador de $0.1\ \mu F$. De este modo, sólo hacen parte del sistema los componentes seleccionados con anterioridad. Posteriormente, se conecta al sistema por el método de acople mecánico, encajando el eje del motor en el orificio del piñón. La unión resultante de todos estos elementos se observa en el montaje de la Figura 3.6:

Figura 3.6. Montaje del sistema mecatrónico.



De esta forma, la Figura anterior tiene una disposición de elementos que se describe a continuación:

- **1** es el módulo ESP32, que conecta los pines D14, D26 y D27. El pin D14 habilita un canal PWM para regular la velocidad de motor y los pines D26 y D27 permiten establecer un nivel alto o bajo de voltaje para controlar la dirección del motor. El pin GND también se conecta para

- **2** es el controlador de motor L298N, conectado a los pines de la ESP32 y cuyas señales de salida le permiten al controlador ajustar la velocidad y dirección del motor, elemento con el que se conecta también por sus terminales positivo y negativo. El controlador es quien recibe la alimentación de la fuente de voltaje por sus entradas de 12V y GND.

- **3** es el motor DC de 12V. Este elemento electromecánico es quien recibe la señal de la ESP32 a través del L298N, convirtiéndola en movimiento rotacional en su eje. Este eje está conectado al piñón de **4**.

- **4** es el mecanismo piñón-cremallera, acoplado por el piñón a **3** por acople mecánico. El piñón encaja con la cremallera convirtiendo el movimiento rotacional en traslacional.

- **5** es la entrada de alimentación, que se conecta a la fuente y energiza los elementos que emplean 12V para su funcionamiento.

- **6** es la entrada del conector del ESP32 con el computador, donde se alimenta el módulo con 5V y también se recibe el código con el que se programan los pines del módulo.

Este montaje presenta la construcción del sistema mecatrónico en base a sus elementos físicos, sin embargo, hace falta determinar las variables de interés y programar el módulo ESP32 para realizar tareas como recibir mensajes de MQTT y enviar señales al controlador L298N, con el propósito de establecer el comportamiento del sistema.

3.2.3. Cálculo de variables de interés

El sistema mecatrónico tiene como objetivo mover una puerta a partir del giro de un motor DC de 12V. En este sentido se determinan los parámetros de entrada y de salida, que son el torque del motor τ_m y la carga o peso de la puerta W_p.

Anteriormente se definió el torque τ_{max} de $42.2\ mN \cdot m$ en el cual el sistema trabaja con el mayor torque posible según la hoja de especificaciones, por lo que se define:

$$\tau_{max} > \tau_m$$

Lo que significa que τ_m está condicionado a τ_{max} para vencer una carga. Este es el punto de partida al determinar las demás variables de interés. En este sentido, se define la siguiente ecuación:

$$\tau_m = F_c \cdot r_p = P_m/\omega_p \tag{3.1}$$

Donde F_c es la fuerza tangencial ejercida sobre la cremallera, r_p el radio del piñón, P_m es la potencia del motor y ω_p es la velocidad angular del piñón. Esto significa que existen dos maneras de obtener τ_m, además de la tabla del Anexo B. A continuación, se plantea τ_m en términos de ω_p.

3.2.3.1. Torque del motor τ_m

En el diagrama del Anexo B, se observa que el torque del motor es inversamente proporcional a la velocidad angular del mismo, y sus valores describen una recta o función lineal. De este modo, se toma la ecuación de la recta para hallar τ_m.

Para hallar la pendiente se define:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.2}$$

Donde x_2 y x_1 son valores finales e iniciales del eje x (correspondientes al torque) y y_2 y y_1 son valores finales e iniciales del eje y (valores de velocidad angular). Se toman estos puntos de la gráfica y se resuelve:

$$\boldsymbol{m} = \frac{5000 - 12500}{25 - 0} = \frac{-7500}{25} = \boldsymbol{-300}$$

Ahora, se toma la ecuación de la recta en función de y:

$$y = mx + b$$

Y reemplazando por las variables de estudio:

$$\omega_p = m\tau_m + b$$

Donde b es el punto de intersección de la recta con el eje y (12500). ω_p se halla a partir de la ecuación 2.2:

$$v_c = \omega_p r_p = \frac{(d_p)(\omega_p)}{2}$$

Donde v_c es la velocidad de cremallera y se puede definir también como:

$$v_c = \frac{d_c}{t} \tag{3.3}$$

La cual está representada en términos del desplazamiento de cremallera d_c (fijado a $10\ cm$, a partir del largo de la puerta) y el tiempo de desplazamiento t (establecido en $5\ s$), obteniendo así:

$$\boldsymbol{v_c} = \frac{d_c}{t} = \frac{10 \times 10^{-2} m}{5\ s} = 2 \times 10^{-2}\ m/s = \boldsymbol{2cm/s}$$

Con v_c obtenido se determina el siguiente valor desconocido r_p, que siendo el radio de piñón efectivo se toma como:

$$r_p = \frac{d_p}{2} \tag{3.4}$$

Resolviendo primero para r_p:

$$r_p = \frac{d_p}{2} = \frac{10.5\ mm}{2} = 5.25\ mm$$

Y luego despejando para ω_p:

$$\omega_p = \frac{v_c}{r_p} = \frac{2 \times 10^{-2} \cancel{m}/s}{5.25 \times 10^{-3} \cancel{m}} = 3.81\, rad/s$$

Se hace la conversión de rad/s a rev/min (Unidad de manejo de la gráfica):

$$3.81 \frac{\cancel{rad}}{\cancel{s}} \cdot \frac{60\ \cancel{s}}{1\ min} \cdot \frac{1\ rev}{2\pi\ \cancel{rad}} = 36.38\ rev/min$$

Finalmente, se despeja τ_m y se halla en la ecuación de la recta:

$$\boldsymbol{\tau_m} = \frac{\omega_p - b}{m} = \frac{36.38 - 12500}{-300} = \boldsymbol{41.54\ mN \cdot m}$$

Adicionalmente, se puede obtener la potencia P_m a partir de con los valores obtenidos de la ecuación 3.1:

$$\boldsymbol{P_m} = \tau_m \cdot \omega_p = 41.54\ mN \cdot m \cdot 3.81\ rad/s = 0.158\ W = \boldsymbol{158\ mW}$$

3.2.3.2. Fuerza tangencial F_c

Con la definición de τ_m en la ecuación 3.1 se puede despejar F_c para obtener su valor:

$$F_c = \frac{\tau_m}{r_p} = \frac{41.54\ mN \cdot \cancel{m}}{5.25 \times 10^{-3} \cancel{m}} = 7.91\ N$$

A partir de la fuerza tangencial se pueden obtener otras variables, como se plantea en la ecuación:

$$F_c = m \cdot g \cdot \mu + m \cdot a + F_e \quad (3.5)$$

Donde:

- m es la masa de la puerta a mover
- g es la constante de la gravedad ($9.81\ m/s^2$)
- μ es el coeficiente de fricción. Para simulación se omite este valor.
- a es la aceleración para el movimiento de la carga, y teniendo en cuenta que el movimiento deseado tiene una velocidad constante, se define $a = 0$.
- F_e son las fuerzas externas adicionales, para este caso son de 0.

$$F_c = m_p \cdot g$$

Así se procede a obtener las variables desconocida m.

3.2.3.3. Masa de la puerta m_p

Ya conociendo las demás variables de la ecuación 3.3 reducida, se despeja m de la siguiente forma:

$$F_c = m_p \cdot g$$

$$m_p = \frac{F_c}{g}$$

Resolviendo para m:

$$\boldsymbol{m_p} = \frac{7.91\ N}{9.81\ m/s^2} = \frac{7.91\ Kg \cdot \cancel{m/s^2}}{9.81 \cancel{m/s^2}} = 0.806\ Kg = \mathbf{806\ g}$$

Finalmente, con el valor de la masa m, se puede determinar el material de la puerta a utilizar por el cálculo de la densidad ρ.

3.2.3.4. Densidad de la puerta ρ

La ecuación 3.7 presenta la densidad en función de la masa m y el volumen V:

$$\rho = \frac{m_p}{V} \tag{3.6}$$

Y teniendo el Volumen expresado en términos de la longitud l, el ancho w y la altura h en la ecuación 3.6,

$$V = l \cdot w \cdot h \tag{3.7}$$

Se determina la densidad ρ:

$$\rho = \frac{m_p}{l \cdot w \cdot h} = \frac{806\ g}{0.1m \cdot 0.01m \cdot 0.1m} = 8060000\ g/m^3 = 8060\ \boldsymbol{Kg/m^3}$$

Esta densidad corresponde a la densidad de diseño $\rho_{diseño}$, sirviendo como referencia para determinar el material de la puerta, según el criterio:

$$\rho_{puerta} < \rho_{diseño}$$

Ya que una densidad igual o mayor a $\rho_{diseño}$ implicaría una carga mayor a la que puede mover el motor.

3.2.3.5. **Material de la puerta**

De acuerdo a lo anterior, se elabora la Tabla 3.5 comparando de diferentes materiales posibles para la elaboración de la puerta a desplazar:

Tabla 3.5. Comparación entre diferentes materiales según su densidad.

Material	**Densidad (Kg/m^3)**	**Propiedades especiales**
Madera	350 - 1050	Fácil manejo y conformado
Acero	7850	Material resistente – mayor peso
Acrílico	1180	Liviano - accesible
Aluminio	2698	Metal liviano – requiere tratamiento
Vidrio	2500	Frágil – difícil manejo

De acuerdo a esta tabla, se selecciona la madera como material de la puerta por su facilidad de acceso y de conformado, a comparación de los demás materiales.

Además, su densidad de $350\,a\,1050\ Kg/m^3$ es menor a la densidad de diseño $\rho_{diseño}$ de $8060\,Kg/m^3$.

3.2.3.6. Recuento de las unidades

Todas las variables especificadas y obtenidas por cálculos se muestran la Tabla 3.6:

Tabla 3.6. Variables obtenidas en el sistema mecatrónico.

Magnitud	Símbolo	Magnitud	Tipo de dato
Torque del motor	τ_m	$41.54\,mN \cdot m$	Calculado
Radio del piñón	r_p	$5.25\,mm$	Calculado
Fuerza tangencial de cremallera	F_c	$7.91\,N$	Calculado
Constante de gravedad	g	$9.81\,m/s^2$	Fijo
Desplazamiento de la puerta	d_c	$10\,cm$	Fijo
Tiempo de desplazamiento	t	$5\,s$	Fijo
Velocidad angular del motor	ω_p	$3.81\ rad/s$	Calculado
Velocidad lineal de cremallera	v_c	$2\ cm/s$	Calculado
Masa de la puerta	m_p	$806\,g$	Calculado
Volumen de puerta	V	$100\,cm^3$	Calculado
Densidad de diseño de la puerta	$\rho_{diseño}$	$8060\,Kg/m^3$	Calculado

De la tabla también se pueden deducir los parámetros de entrada para garantizar el funcionamiento del sistema mecatrónico, como el torque y las propiedades de la puerta. Todos estos valores y procedimientos serán la base del desarrollo de la extrapolación a escala relacionado en la sección 3.3.

3.2.4. Programación del módulo ESP32

Para esta sección se considera la programación del módulo ESP32 únicamente en sus funciones de controlar la velocidad y dirección de giro del motor. La programación para recibir mensajes de MQTT se especifica en el capítulo de Integración de sistemas con MQTT. En este sentido, se toma como referencia el código en [30] y se realizan las siguientes operaciones:

3.2.4.1. **Declaración de variables**

Se declaran tres variables que se vinculan a los pines a utilizar del ESP32, que son los pines D14, D26 y D27. Además, se declaran otras cuatro variables que hacen referencia a las propiedades de la señal PWM a generar, como la frecuencia del pulso (menor a la frecuencia de reloj del ESP32), selección de canal PWM (de 0 a 15), la resolución del ciclo útil (1 a 16 bits), y un valor inicial de ciclo útil. La Figura 3.7 presenta la porción de código correspondiente a la declaración de variables:

Figura 3.7. Declaración de variables asociadas a diferentes pines.

```
//DECLARACIÓN DE VARIABLES//
// Asignar variables a pines determinados
int motor1Pin1 = 27;
int motor1Pin2 = 26;
int enable1Pin = 14;

// Establecer propiedades de PWM
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 200;
```

3.2.4.2. **Escritura de la función de inicialización *void setup()***

En esta función se inserta el código que establece las condiciones iniciales de trabajo del módulo ESP32, considerando que solo se ejecuta una vez. En este caso, se configuran los pines D14, D26 y D27 como salidas de señal, también se asigna un canal PWM al pin D14 con la función *ledcAttachPin* y se establecen las variables que definen las propiedades de la señal PWM con la función *ledcSetup*. La Figura 3.8 muestra las configuraciones realizadas:

Figura 3.8. Configuración de la función *void setup().*

```
void setup() {
  // Configurar pines como salidas:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  // configurar funciones del PWM
  ledcSetup(pwmChannel, freq, resolution);

  // Vincular el canal PWM al pin GPIO que será controlado
  ledcAttachPin(enable1Pin, pwmChannel);
```

Se tiene en cuenta que el pin D14 está habilitado para emplearse como generador de señales PWM y los pines D26 y D27 se pueden disponer como entradas y salidas de niveles de voltajes altos y bajos, como lo presenta el diagrama de pines del ESP32 en el Anexo D.

3.2.4.3. Escritura de la función principal *void loop()*

Esta función contiene todo el código relacionado con la operación del módulo ESP32, y en ella se escribe todos los comandos que cambian el comportamiento de los pines declarados con anterioridad. Específicamente, se habla de cómo se consigue el cambio de dirección y velocidad del motor:

- **Cambio de dirección:** Los pines D26 y D27 cambian sus niveles de voltaje de salida, donde D26 con nivel alto y D27 con nivel bajo permiten el giro del motor en sentido horario y viceversa, para un giro en sentido antihorario.

- **Cambio de velocidad:** El pin D14 vinculado a un canal PWM ofrece un pulso de salida que dependiendo de su ciclo útil hará que el motor se mueva a una velocidad determinada. El valor de ciclo útil se determina en la sección de evaluación de variables del sistema mecatrónico.

- **Duración de movimiento:** Se establece con un valor de tiempo *t* incluido en el retraso generado por el comando *delay(t)*, en milisegundos. Este tiempo se determina con la velocidad del motor al realizar el movimiento y el desplazamiento de la puerta

planteado. Este parámetro se determina en la sección de evaluación de variables del sistema mecatrónico.

La Figura 3.9 presenta el código en la función principal para establecer un control en la dirección, velocidad y duración de movimiento del motor:

Figura 3.9. Configuración de la función *void loop()*.

```
//SECCIÓN PARA CONTROL DE VELOCIDAD Y POSICIÓN DEL MOTOR
void loop() {
//Fijar variables dependientes: ciclo útil (0-255) y duración de movimiento t
    ledcWrite(pwmChannel, dutyCycle);
    dutyCycle = 20; //Valor de referencia, dutyCycle se determina más adelante
    t = 2000 //Valor de referencia, t se determina más adelante

// Movimiento del motor en sentido horario
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
    Serial.print("Ciclo útil: ");
    Serial.println(dutyCycle);
    delay(t);

     // Movimiento del motor en sentido antihorario
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);
    Serial.print("Ciclo útil: ");
    Serial.println(dutyCycle);
    delay(t);
  }
```

De esta forma queda preparado el código para mover el motor en función de variables de entrada de ciclo útil (directamente relacionado con la velocidad de giro) y tiempo de duración de movimiento (De acuerdo al desplazamiento deseado). Este código se sube al módulo ESP32 desde un computador, conectándose a través del conector en **6** de la Figura 3.6.

Así, con la conexión de componentes y configuración del ESP32 como elemento asociado al control del movimiento, el sistema mecatrónico puede operar de forma independiente con el objetivo de realizar la apertura y cierre de puertas a escala.

3.3. Extrapolación a escala real por medio de SolidWorks y Matlab

Para hacer del sistema mecatrónico un dispositivo aplicable a escala real es necesario definir cuáles son los parámetros que cambian con respecto al sistema a escala. En este caso, corresponde a las dimensiones y material de la puerta que son mayores a comparación de la puerta a escala, la cual requiere del cambio de otros parámetros y elementos con el fin de garantizar su movimiento. En la Tabla 3.7 se contemplan los ajustes de los diferentes componentes del sistema mecatrónico, así como el cambio en los parámetros asociados a cada uno de ellos, en caso de que sea requerido:

Tabla 3.7. Cambios en componentes del sistema mecatrónico.

COMPONENTE		¿REQUIERE CAMBIOS?	PARÁMETROS ASOCIADOS	CAMBIOS
Módulo ESP32-CAM		SI	Dirección y velocidad del motor	Cambio de variables de velocidad y tiempo de desplazamiento
Controlador L298N		NO	Ninguno	Ninguno
Motor DC 12V		SI	Torque y velocidad angular	Implementar un motor de mayor capacidad según se requiera
Mecanismo	Piñón	NO	Velocidad angular	Ninguno
	Cremallera	SI	Distancia de desplazamiento	Incrementar longitud de cremallera

Teniendo en cuenta lo anterior, se emplea el software Matlab para realizar una simulación del sistema que tiene en cuenta todos los parámetros relacionados e implementa entradas y salidas de movimiento emulando los componentes de velocidad del motor y carga de la puerta, en base a elementos mecánicos presentes como el mecanismo piñón cremallera.

Ya con claridad de los conceptos necesarios para extrapolar el sistema mecatrónico a escala real, se sigue el siguiente procedimiento:

3.3.1. Modelado del mecanismo en SolidWorks

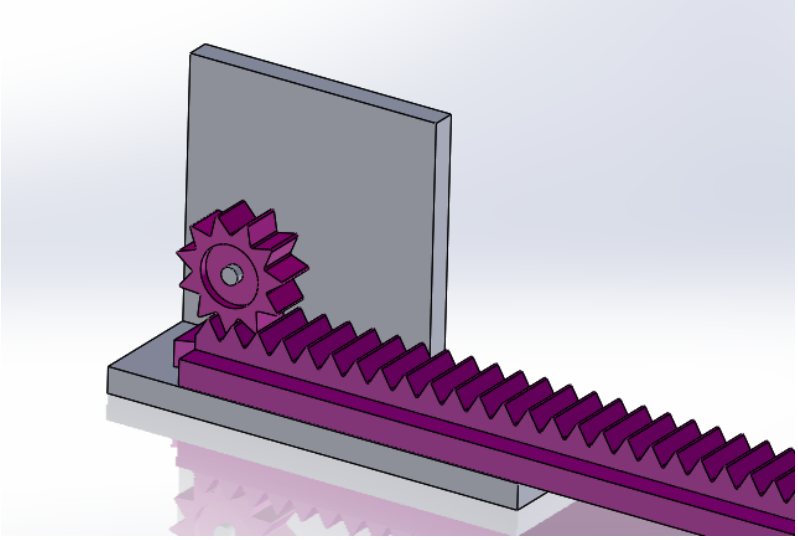
Para poder realizar la simulación del sistema, es necesario el modelado previo de los elementos mecánicos en SolidWorks que luego son exportados a Matlab. Este modelado

depende de las medidas de las dimensiones y demás propiedades físicas relevantes del piñón y la cremallera del mecanismo, presentadas a continuación:

- **Piñón:** Diámetro de 12 mm, 11 dientes (mantiene igual al modelo a escala)
- **Cremallera:** Longitud de 820 mm, 272 dientes (con ajuste a puerta a escala real)

Con estas dimensiones, se modelan las piezas en SolidWorks y con su ensamblaje resulta el mecanismo modelado de la Figura 3.10:

Figura 3.10. Modelado en SolidWorks del mecanismo piñón cremallera.



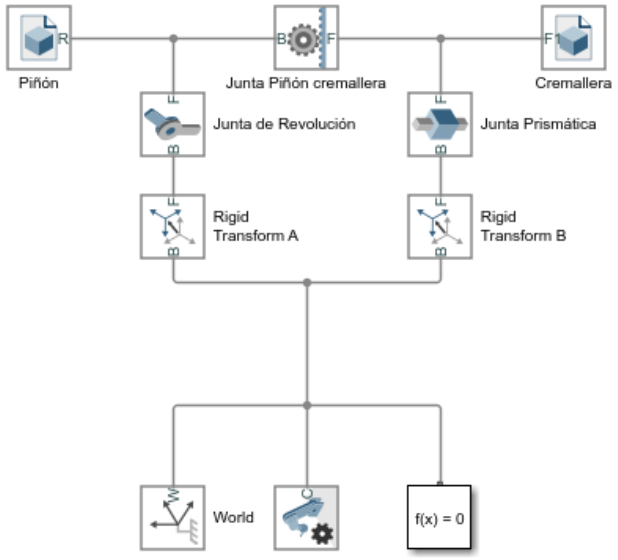
La Figura incluye una base o soporte (Pieza en color gris) que es fija y se requiere para establecer las relaciones de posición que acoplan el piñón con la cremallera en SolidWorks.

3.3.2. Exportar el modelo a Matlab

El ensamblaje en SolidWorks se exporta por medio de la herramienta *Simscape Multibody Link* instalado previamente en ambos programas. En SolidWorks, los archivos de las piezas son convertidos a un formato compatible con Matlab, de extensión *.step*, mientras que el ensamblaje se convierte a un archivo de extensión *.xml*.

En Matlab, se busca la ubicación de los archivos convertidos y se ejecuta el comando *smimport ('Archivo.xml')* Donde 'Archivo' es el nombre del ensamblaje. Al aplicar este comando, las piezas son exportadas a Simulink y con ellas se arma un diagrama de bloques que describe el funcionamiento de las piezas, como se observa en la Figura 3.11:

Figura 3.11. Diagrama de bloques del mecanismo piñón cremallera.

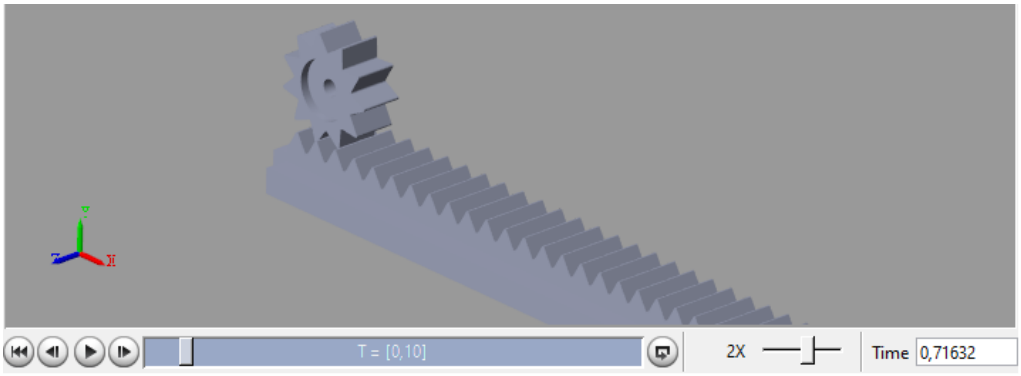


La función de los bloques más importantes se describe a continuación:

- **Piñón,** que contiene la pieza del piñón.
- **Cremallera,** que contiene la pieza de cremallera.
- **Junta Piñón cremallera,** que acopla el piñón y la cremallera en base al radio del círculo de paso del piñón.
- **Junta de Revolución,** que describe el movimiento rotacional del piñón y
- **Junta Prismática,** que describe el movimiento traslacional de la cremallera.

Con este diagrama de bloques, también llamado modelo en Simulink, se puede realizar la simulación del mecanismo, mostrando las piezas acopladas como se observa en la Figura 3.12:

Figura 3.12. Piezas del mecanismo en entorno de simulación de Matlab.



3.3.3. Criterio de extrapolación a escala real

La extrapolación a escala real se basa en el cambio de especificaciones en los elementos de entrada y de salida que son el motor DC y la puerta, respectivamente. Se requiere detallar las variables asociadas a cada uno de estos elementos e incorporarlas con una fórmula producto del sistema de ecuaciones en la sección anterior. Los parámetros de cambio son:

- **Puerta a escala real:** Dimensiones de $4\ cm \times 82\ cm \times 192\ cm$, se selecciona material madera con ρ de $400\ \ Kg/m^3$. También se requiere la carga de la puerta W_p.
- **Motor DC 12V:** Velocidad angular ω_p y torque τ_m definidos a partir de las nuevas propiedades de la puerta.
- **Desplazamiento de Cremallera:** Longitud que corresponde con el ancho de la puerta, es decir de $82\ cm$.

Para el torque del motor τ_m se define una fórmula del torque en función de la densidad, empleando las ecuaciones de la sección 3.2.3 de la siguiente forma:

a. Ecuación inicial $$\tau_m = F_c \cdot r_p$$

b. Reemplazar F_c $$\tau_m = \left(m_p \cdot g\right) \cdot r_p$$

c. Reemplazar m_p $$\tau_m = [(\rho \cdot V) \cdot g] \cdot r_p$$

d. Reemplazar datos invariantes $$\tau_m = [(\rho \cdot V) \cdot 9.81\, m/s^2] \cdot 5.25 \times 10^{-3} m$$

e. Operar y establecer fórmula $\tau_m = (\rho \cdot V) \cdot 51.52 \times 10^{-3}\, m^2/s^2$

De esta manera, se define τ_m en función de los parámetros de la puerta. Ya definidos ρ y V, se halla el torque τ_m:

$$\tau_m = \left(400\, kg/m^3 \cdot (4 \times 10^{-2} m \cdot 82 \times 10^{-2} m \cdot 192 \times 10^{-2} m)\right) \cdot 51.52 \times 10^{-3}\, m^2/s^2$$

$$\tau_m = (400\, kg/\cancel{m^3} \cdot 62.976 \times 10^{-3} \cancel{m^3}) \cdot 51.52 \times 10^{-3}\, m^2/s^2$$

$$\tau_m = 25.19\, kg \cdot 51.52 \times 10^{-3}\, m^2/s^2$$

$$\boldsymbol{\tau_m = 1.297\, kg \cdot m^2/s^2 = 1.29\, N \cdot m}$$

Con el valor de torque resultante se puede definir el tipo de motor a emplear, el cual debe tener la capacidad de generar como mínimo un torque de $1.29\, N$. En este caso, se selecciona el motor modelo 8DCG12-30-50 de alta capacidad de torque, con hoja de especificaciones en el Anexo C.

A partir de la curva de rendimiento de la hoja de especificaciones se puede obtener la velocidad angular del motor ω_p, empleando la ecuación de la recta 3.2:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{0 - 3500}{15.5 - 0} = \frac{-3300}{15.5} = -212.9$$

Luego, aplicando la ecuación de la recta en función de y (eje de velocidad angular), se tiene:

$$y = mx + b$$

Y Reemplazando:

$$\omega_p = m\tau_m + b$$

$$\omega_p = -212.9 \cdot 13.15 + 3300$$

$$\boldsymbol{\omega_p = 500.365 \approx 500\ rev/min}$$

Se tiene en cuenta que la curva de rendimiento maneja el torque en $kg \cdot cm$, por lo que se hizo la conversión de $1.29\, N \cdot m$ a $13.15\, kg \cdot m$.

Ya teniendo τ_m y ω_p se obtienen las demás variables de entrada, que son la masa de la puerta m_p y el tiempo de desplazamiento t.

Para el valor de la masa de puerta m_p, se despeja la ecuación 3.6:

$$m_p = \rho \cdot V$$

Y se resuelve:

$$m_p = 400\ kg/m^3 \cdot (4 \times 10^{-2}m \cdot 82 \times 10^{-2}m \cdot 192 \times 10^{-2}m)$$

$$m_p = 400\ kg/\cancel{m^3} \cdot 62.796 \times 10^{-2}\cancel{m^3}$$

$$\boldsymbol{m_p = 25.19\ kg}$$

Para el valor del tiempo de desplazamiento t se halla primero la velocidad lineal de cremallera de la ecuación 2.2:

$$v_c = \omega_p r_p = \left(500 \frac{\cancel{rev}}{\cancel{min}} \cdot 5.25 \times 10^{-3}m\right) \cdot \frac{2\pi\ rad}{1\ \cancel{rev}} \cdot \frac{1\ \cancel{min}}{60\ s}$$

$$\boldsymbol{v_c} = 0.27\ m/s = \boldsymbol{27\ cm/s}$$

Finalmente, con el valor de v_c se halla t despejándolo en la ecuación 3.3:

$$\boldsymbol{t} = \frac{d_c}{v_c} = \frac{0.82\ \cancel{m}}{0.27\ \cancel{m}/s} = \boldsymbol{3.037\ s}$$

En la simulación del sistema a escala real, los datos que requiere el sistema para funcionar son ω_p y m_p.

3.3.4. Inserción de parámetros en el modelo de Matlab

Ya preparado el entorno de simulación y obtenido los parámetros de entrada, se procede a insertar los parámetros de entrada y de salida de acuerdo a las siguientes consideraciones:

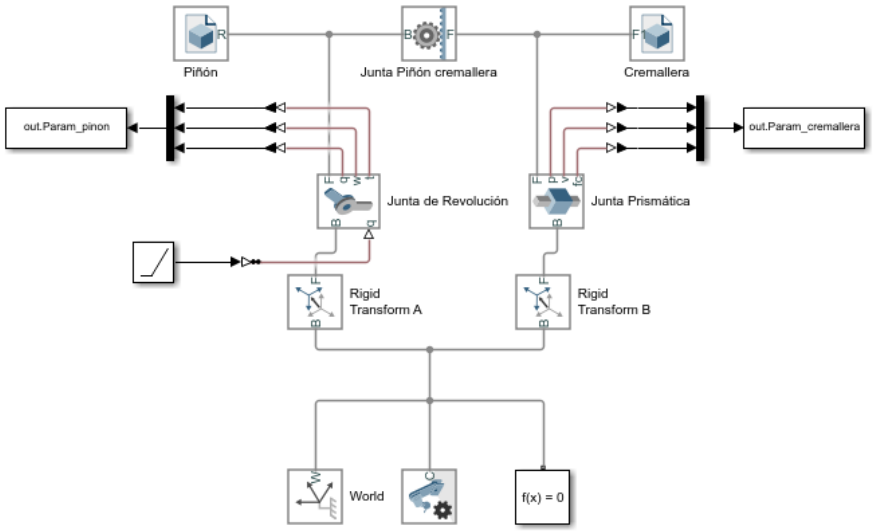
- Los parámetros de entrada corresponden a la velocidad angular ω_p, que para simulación se establece como una señal de tipo rampa introducida a la Junta de

revolución. El parámetro restante corresponde a la masa de la puerta m_p que en simulación se establece en las propiedades mecánicas de la cremallera.

- Los parámetros de salida son todas aquellas variables medibles que pueden obtenerse a partir de las juntas de revolución y prismáticas, tales como posición, velocidad, torque resultante y fuerza tangencial con respecto al tiempo. A estas salidas se conectan bloques denominados *To workspace*, que convierten las señales de salida en matrices de datos numéricos que pueden emplearse en la realización de gráficas.

Y de este modo, se realiza la conexión de bloques de entrada y de salida al modelo del mecanismo:

Figura 3.13. Diagrama de bloques con señales incorporadas de entrada y de salida.



En este punto todo el entorno de simulación está preparado, y de aquí se puede realizar la evaluación de las diferentes variables del sistema mecatrónico simulado.

3.4. Integración de la Red Inalámbrica de Cámaras y el sistema mecatrónico empleando MQTT

Para poder integrar los dos sistemas trabajados anteriormente se requiere de dos procedimientos principales: Montar el servidor MQTT y su acople con ambos sistemas. En el caso de la RIC se modifica la programación del módulo ESP32-CAM correspondiente al nodo sumidero, que recibe la información de los demás nodos. El sistema mecatrónico sigue un procedimiento parecido, con la diferencia de que la programación recae sobre el módulo ESP32, que es el que recibe la señal de entrada para dicho sistema. En este sentido, se comienza con el montaje del MQTT Broker.

3.4.1. Montaje del MQTT Broker

El MQTT Broker consiste de un servidor que gestiona todos los mensajes que son enviados y recibidos por parte de diferentes dispositivos inalámbricos con conexión a la red. Para este proyecto se tiene como consideración principal su instalación, que se realiza en el computador personal del autor principal por facilidad de acceso y cumplimiento de requisitos de operación mencionados en la sección 2.3.2.

Existen diferentes tipos de broker que manejan el protocolo MQTT, como el HiveMQ o el EMQTT, sin embargo, se escoge el Mosquitto, dado que se instala fácilmente y puede ser operado directamente desde el símbolo del sistema del computador, a diferencia de los dos primeros que requieren de acceso por medio de un navegador de internet.

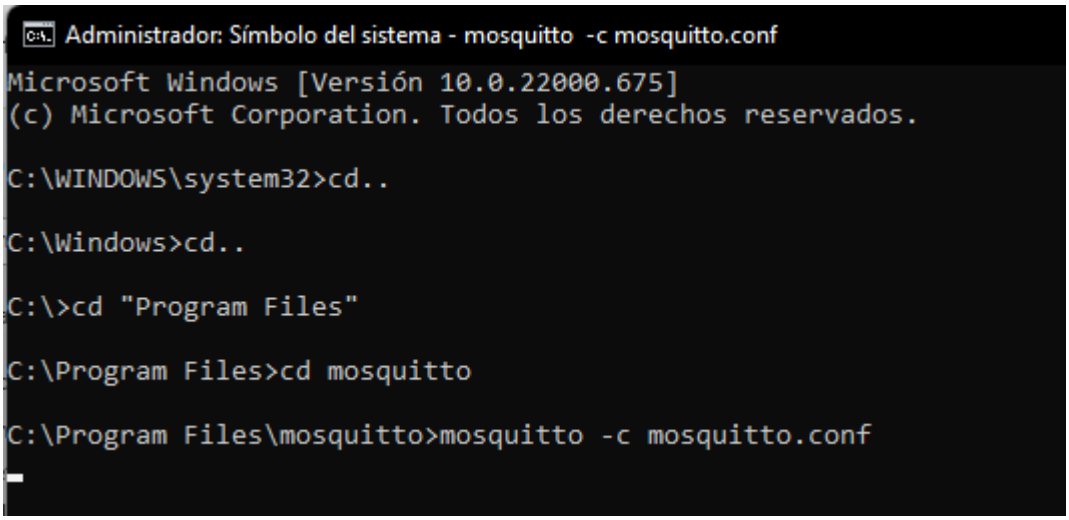
La instalación del broker Mosquitto se realiza como cualquier otro programa en el computador, y su puesta en funcionamiento se realiza de la siguiente manera:

- Se abre el símbolo de sistema y se busca la ubicación del programa instalado llegando a la carpeta de destino llamada ‘mosquitto’ y

- Se ejecuta el comando ‘*mosquitto -c mosquitto.conf*’, el cual enciende el servidor MQTT para poder ser utilizado.

La Figura 3.14 muestra los comandos realizados para cumplir con las dos tareas antes mencionadas:

Figura 3.14. Símbolo de sistema como herramienta de manejo del MQTT broker.



En este punto el MQTT broker ya está en operación en el computador, por lo que el siguiente paso es establecer el envío y recepción de mensajes de los módulos de la RIC y el sistema mecatrónico.

3.4.2. Acople con la Red Inalámbrica de Cámaras

La integración por parte de la RIC se basa en la programación del módulo que va a transmitir la información, donde el módulo es el ESP32-CAM que actúa como nodo sumidero y la información corresponde a la alarma al vigilante al realizarse una detección. Teniendo clara esta consideración inicial, se realiza las siguientes modificaciones:

- El código de programación del ESP32-CAM mencionado en la sección 3.1.4, así como se planteó, sólo funciona como una red de cámaras y reconocimiento facial pero no está asociado al MQTT broker. La asociación correspondiente se logra con dos condiciones: Que la detección de rostros genere algún tipo de señal y que esta pueda ser transmitida como un mensaje al MQTT broker.

- Para generar una señal a partir de la detección de rostros, se modifica inicialmente la porción de código asociada a la detección, en este caso creando una variable booleana “deteccion”, que se exporta al código principal con aplicaciones como condicional. Este proceso se observa en la Figura 3.15:

Figura 3.15. Porciones de código para generar una señal de respuesta a la detección: En código de reconocimiento facial (arriba) y en código principal (abajo).

```
if (net_boxes || fb->format != PIXFORMAT_JPEG){
    if(net_boxes){
        detected = true;
        if(detected == true){
            deteccion = true;
        }
        if(recognition_enabled){
            face_id = run_face_recognition(image_matrix, net_boxes);
        }
        fr_recognize = esp_timer_get_time();
        draw_face_boxes(image_matrix, net_boxes, face_id);
        dl_lib_free(net_boxes->score);
        dl_lib_free(net_boxes->box);
        dl_lib_free(net_boxes->landmark);
        dl_lib_free(net_boxes);
    }
```

```
void loop() {

  // ************************SECCIÓN MQTT************************
  mqttClient.loop();
  mesh.update();

  if(myIP != getlocalIP()){
    myIP = getlocalIP();
    Serial.println("My IP is " + myIP.toString());

    if (mqttClient.connect("painlessMeshClient")) {
       if (deteccion==true){
        Serial.print("Detectado");
        mqttClient.publish("painlessMesh/from/gateway","Detectado");
        mqttClient.subscribe("painlessMesh/to/#");
        }
        deteccion = false;
        delay(1000);
    }
```

En este espacio de código se inserta todas las líneas de comandos referentes al envío del mensaje al MQTT broker, que se detallan a continuación.

- El mensaje en el monitor de recursos *“Detectado”* indica que hubo una detección por parte de la Red Inalámbrica de Cámaras, en este caso como confirmación de que se realiza la detección y así mismo su interpretación como un mensaje.

- La sección *mqttClient.publish* corresponde al código de acople entre el módulo ESP32-CAM y el MQTT broker, que realiza una publicación al tópico *painlessMesh/from/Gateway*.

- El servidor MQTT broker tiene una suscripción al tópico *painlessMesh/from/#*, lo que significa que apenas el va a recibir el mensaje del módulo ESP32-CAM apenas sea publicado. En este caso, el mensaje es “Detectado”.

- Este mensaje corresponde a una alarma visual definitiva que el vigilante podrá observar, para dar así lugar a la decisión de abrir o cerrar las puertas, mediante una instrucción que se detalla en la siguiente sección.

3.4.3. Acople con el sistema mecatrónico

El sistema mecatrónico logra su comunicación con el MQTT broker a través del módulo ESP32. A su vez, este dispositivo controla la dirección y velocidad del motor, como se observó en la sección 3.2. La integración con MQTT permite al sistema mecatrónico actuar a partir de mensajes enviados por el vigilante, en forma de instrucciones, desde el computador. La implementación de esta funcionalidad se basó en la programación del módulo ESP32, adicionando comandos de comunicación con MQTT a la sección de código que determina la velocidad y dirección del motor, como se observa en la Figura 3.16:

Figura 3.16. Código de parámetros del motor ajustado a respuesta de MQTT.

```
void loop()
{
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
    t = 5000; //Tiempo de referencia t=5s
  if (mqttClient.subscribe("painlessMesh/to/open")){
    ledcWrite(pwmChannel, dutyCycle);
    dutyCycle = 2;
    digitalWrite(motor1Pin1, HIGH); // Movimiento del motor en sentido horario
    digitalWrite(motor1Pin2, LOW);
    Serial.print("Ciclo útil: ");
    Serial.println(dutyCycle);
    delay(t);
    dutyCycle = 0;
  }
  if (mqttClient.subscribe("painlessMesh/to/close")){
    ledcWrite(pwmChannel, dutyCycle);
    dutyCycle = 2;
    digitalWrite(motor1Pin1, LOW); // Movimiento del motor en sentido antihorario
    digitalWrite(motor1Pin2, HIGH);
    Serial.print("Ciclo útil: ");
    Serial.println(dutyCycle);
    delay(t);
    dutyCycle = 0;
  }
```

La Figura muestra dos bloques de comandos asociados a los escenarios posibles en la actuación del sistema mecatrónico: Apertura y cierre de puertas a partir de la dirección de movimiento del motor.

Cada bloque de comandos se hace efectivo si el módulo ESP32 logra suscribirse al tópico especificado. Por ejemplo, si el vigilante desea abrir la puerta, él publica un mensaje concreto de tópico *painlessMesh/to/open.* Al ser publicado este mensaje el ESP32 ya puede suscribirse al tópico, y al estar suscrito, se cumple la condición que finalmente permite al módulo ejecutar las condiciones de movimiento del motor.

3.5. Reconocimiento del rostro de los transeúntes con el fin de generar una alarma para apoyar la labor del vigilante

Para que la RIC pueda reconocer rostros y posteriormente genere una alarma se implementó un código que pone en funcionamiento la propiedad de detección y reconocimiento facial incorporada en el módulo ESP32-CAM. De esta manera, se llevó a cabo el siguiente procedimiento:

El primer paso consistió en programar el módulo en Arduino con un código ya establecido, que incluye, entre otras, las siguientes librerías:

- **esp_camera.h:** Que permite la compatibilidad del módulo ESP32 con sensores de imagen, como la cámara OV2640 empleada en este proyecto.
- **camera_index.h:** Contiene un arreglo de bits en hexadecimal, con el objetivo de visualizar la imagen de la cámara en una página web.
- **esp_http_server.h:** Permite montar una página web con propiedades del código empleado, en este caso para configurar las características de la captura de imagen.
- **camera_pins.h:** Donde se establecen los pines disponibles del módulo según el modelo y especificaciones.
- **fb_gfx.h, fd_forward.h y fr_forward.h:** Librerías que incorporan el reconocimiento facial, siendo las dos últimas para detección y reconocimiento, respectivamente.

Al ser un código ya conformado las ediciones en el mismo corresponden a las propiedades del módulo y de conexión. La Figura 3.17 muestra los campos de códigos a modificar:

Figura 3.17. Porción de código con propiedades de la RIC.

```
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"

const char* ssid = "Nombre de red";
const char* password = "Contraseña";
```

Donde el primer campo marcado corresponde al modelo de cámara utilizado (Ai Thinker) y el segundo campo indica el nombre y contraseña de la red a la cual está conectado el módulo ESP32-CAM.

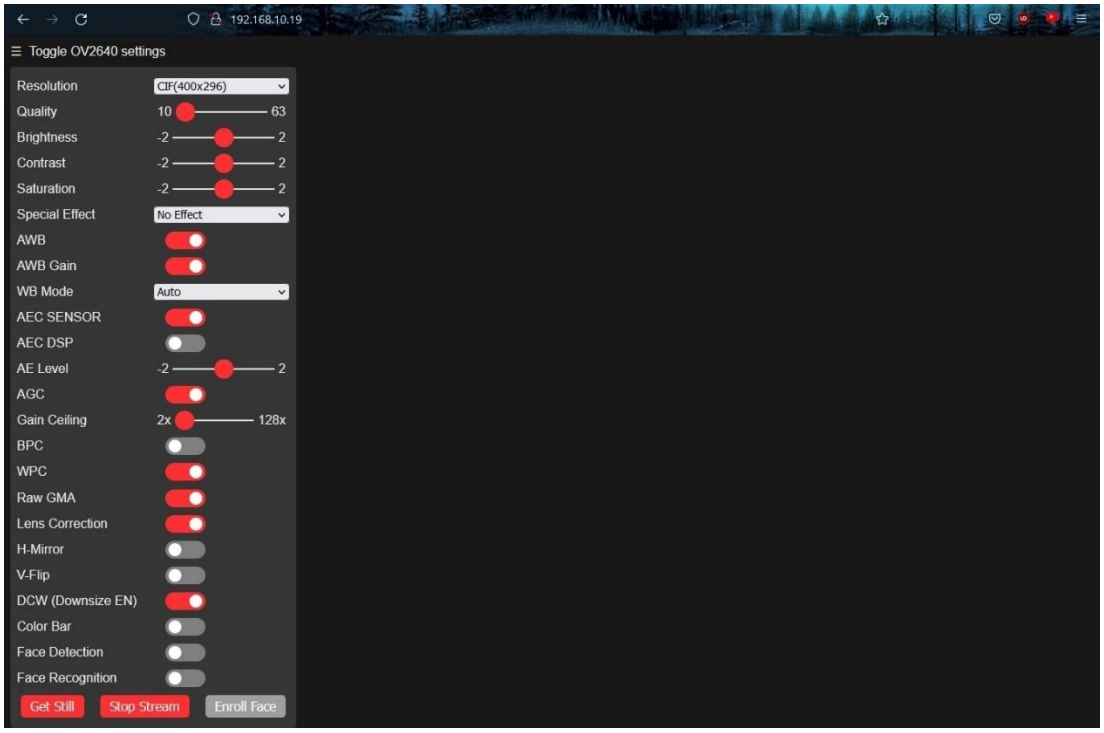
Una vez modificados estos campos se subió el código al módulo, que posteriormente se puso en funcionamiento. Se abrió el monitor de recursos, para observar información relevante como el estado de conexión a la red y como la IP asignada al dispositivo, tal como se evidencia en la Figura 3.18:

Figura 3.18. Monitor de recursos con información de conexión a la red.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4

.
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.10.19' to connect
```

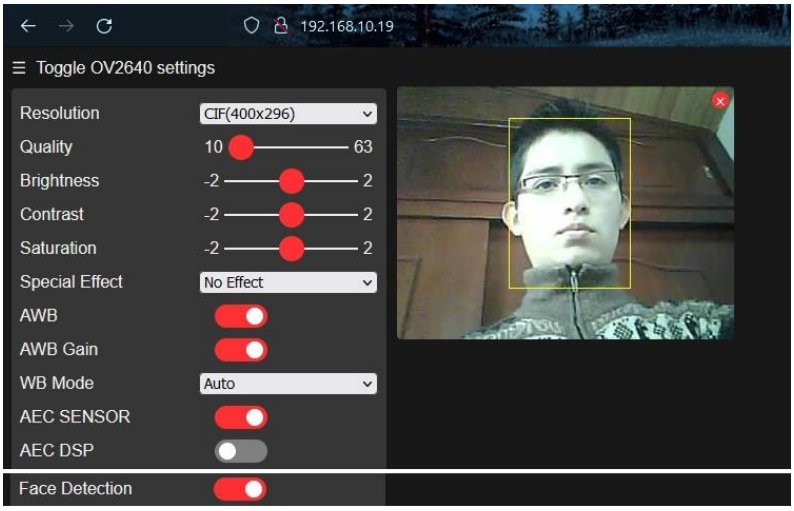
La dirección obtenida (http://192.168.10.19) al ser puesta en cualquier navegador de internet, permite la configuración de la cámara y sus propiedades a través de una página web. La Figura 3.19 muestra la interfaz generada:

Figura 3.19. Interfaz de configuración de la cámara en ESP32-CAM.



Desde aquí es posible el empleo de la cámara y su aplicación para reconocimiento de rostros. Se selecciona el botón “Start Stream” para comenzar la transmisión de imagen y se habilita la opción “Face Detection” para comenzar la detección facial. En la Figura 3.20 se observa el resultado de esta acción tomando al autor principal como referencia:

Figura 3.20. Puesta en funcionamiento de la cámara y detección facial.

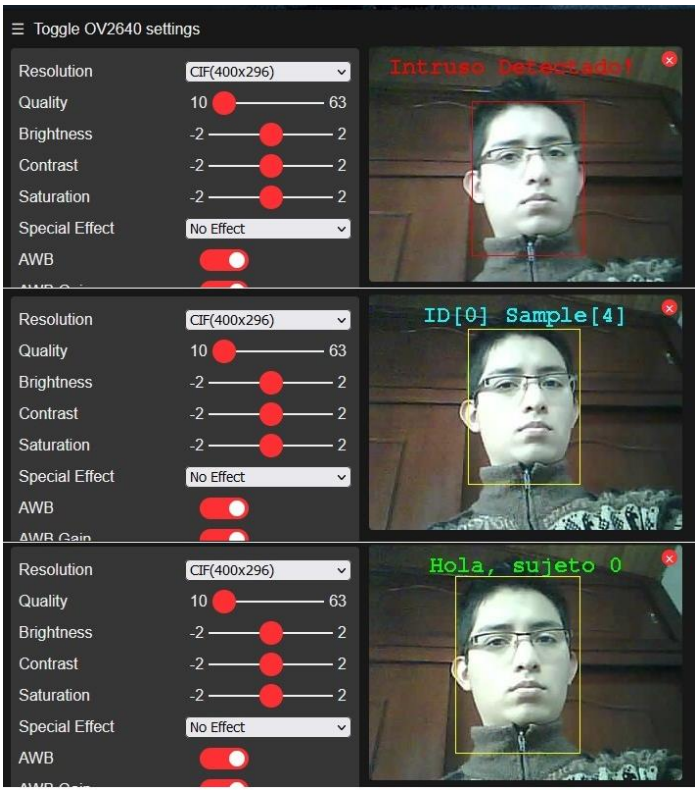


Así mismo, la interfaz hace evidente la detección de un rostro remarcándolo con un cuadrado de color amarillo.

El siguiente paso fue implementar el reconocimiento facial. Se habilitó la opción "*Face Recognition*" de la interfaz, teniendo en cuenta que la cámara está funcionando y la opción de detección facial está activada. La cámara inmediatamente tomó el rostro en imagen como no identificado, dado que en el momento no se encuentra en la base de datos del ESP32-CAM. De esta manera, se selecciona la opción "*Enroll Face*", la cual permite a la cámara tomar muestras del rostro que son guardadas como referencia.

Al finalizar la toma de muestras la cámara reconoce el rostro a partir de las capturas de referencia, finalizando el proceso. La Figura 3.21 presenta los diferentes resultados conforme se siguieron los pasos de reconocimiento facial:

Figura 3.21. Proceso de reconocimiento facial. Rostro no identificado (arriba), Toma de muestras (mitad) y Rostro ya reconocido (abajo).



3.6. Evaluación de la solución propuesta midiendo la velocidad de traslación de la puerta, el torque del motor, la latencia, la cantidad de paquetes perdidos, la intensidad de señal recibida (RSSI) y el consumo de energía.

Esta fase consiste en la preparación de los elementos que componen el sistema integral de la solución propuesta, para ser evaluados a partir de la medición de parámetros que describen sus propiedades operacionales, como el comportamiento de los diferentes dispositivos y el rendimiento que ellos proporcionan. Estos parámetros se contemplan en base a los sistemas principales: La Red Inalámbrica de Cámaras y el sistema mecatrónico. Los parámetros que se analizan en el primer sistema son la latencia, cantidad de paquetes perdidos, intensidad de señal recibida (RSSI) y consumo de energía; mientras que en el segundo sistema se toma la velocidad de traslación de la puerta, el torque del motor y otros parámetros correspondientes al mecanismo piñón-cremallera.

3.6.1. Parámetros del sistema mecatrónico

La medición de parámetros del sistema mecatrónico se fundamenta en la implementación de la simulación de la sección 3.3, preparando el modelo con entradas y salidas para así obtener las variables de interés.

Esta preparación consiste en tomar cada una de las señales de salida y convertirlas en conjuntos de datos numéricos que harán las veces de muestras, y serán evaluados en función del tiempo, magnitud que también es convertida en un conjunto de datos.

Para la cremallera se toman parámetros de posición, velocidad y fuerza tangencial, mientras que para el piñón se tienen parámetros de posición, velocidad angular y torque; todos ellos en función del tiempo. Como consideración especial se toman las muestras de la velocidad angular en función de las muestras del torque. El criterio de obtención de muestras es el siguiente:

- Se toma un tiempo de muestreo de $10\ s$, suficiente para mostrar el comportamiento de todas las variables.

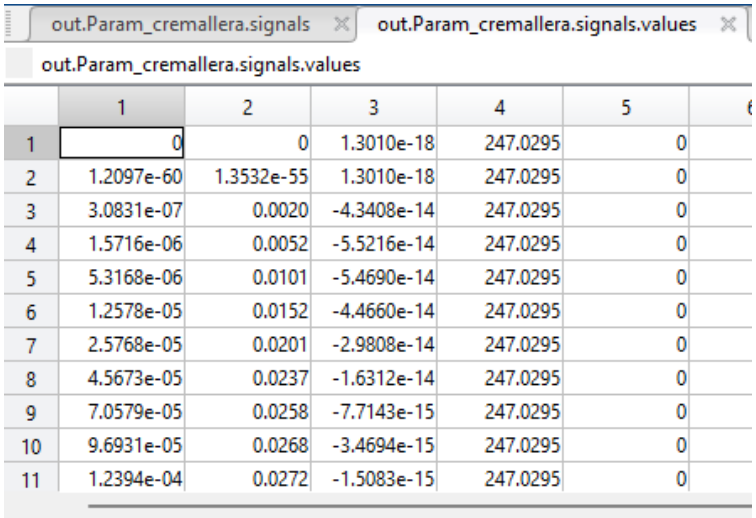
- Se realizan dos simulaciones del modelo, una presenta los valores de entrada del sistema mecatrónico con la puerta a escala y otra tiene como referencia los valores obtenidos en la extrapolación a escala real. Se realizará una comparación entre ambas simulaciones.

- Dado que las muestras se toman de una simulación no es necesaria la repetición de pruebas del mismo tipo, ya que el resultado será el mismo.

- Se tendrá en cuenta los valores calculados en la sección 3.2.3 y 3.3.3, para evaluar la coherencia de los valores calculados con los medidos.

A partir de lo anterior se realizan las simulaciones, cada una de ellas generando dos matrices de datos, una del piñón y otra de la cremallera. Estas matrices quedan distribuidas en hojas de cálculo, como se ve en la Figura 3.22, que presenta las muestras resultantes en la cremallera:

Figura 3.22. Conjunto de datos de salida de la cremallera, exportados desde el modelo en Simulink.



out.Param_cremallera.signals | out.Param_cremallera.signals.values

out.Param_cremallera.signals.values

	1	2	3	4	5
1	0	0	1.3010e-18	247.0295	0
2	1.2097e-60	1.3532e-55	1.3010e-18	247.0295	0
3	3.0831e-07	0.0020	-4.3408e-14	247.0295	0
4	1.5716e-06	0.0052	-5.5216e-14	247.0295	0
5	5.3168e-06	0.0101	-5.4690e-14	247.0295	0
6	1.2578e-05	0.0152	-4.4660e-14	247.0295	0
7	2.5768e-05	0.0201	-2.9808e-14	247.0295	0
8	4.5673e-05	0.0237	-1.6312e-14	247.0295	0
9	7.0579e-05	0.0258	-7.7143e-15	247.0295	0
10	9.6931e-05	0.0268	-3.4694e-15	247.0295	0
11	1.2394e-04	0.0272	-1.5083e-15	247.0295	0

En la figura tres de las cinco columnas representan los valores de una variable de la cremallera: La primera contiene datos de posición, la segunda de velocidad lineal y la cuarta la fuerza tangencial en la cremallera.

Como último proceso de preparación, se toman esos datos para obtener las gráficas y se crea un script insertando las instrucciones que se ven en la Figura 3.23:

Figura 3.23. Script para generar gráficas de las distintas variables.

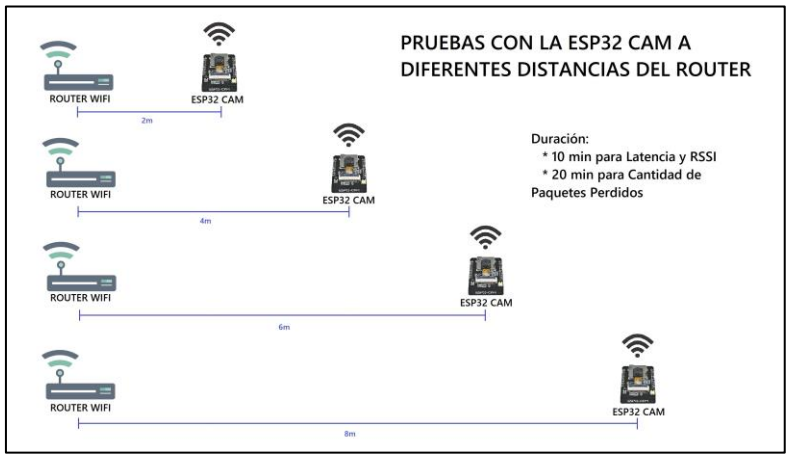
```
plot (out.Param_cremallera.time,out.Param_cremallera.signals.values(:,"# de columna"))

title("Titulo de gráfica",'Fontsize',18)
xlabel("Etiqueta eje x",'Fontsize',18)
ylabel("Etiqueta eje y",'Fontsize',18)
```

La Figura muestra la plantilla del script a ejecutar, donde reemplaza el número de columna por el correspondiente en la hoja de cálculo de la Figura 3.21, además de colocar las respectivas leyendas y títulos de gráfica haciendo referencia a la variable a mostrar.

3.6.2. Parámetros de la Red Inalámbrica de Cámaras

La medición de parámetros de la RIC se basa en la toma de muestras en función del tiempo, entre dos nodos de la red de los cuales uno es un módulo ESP32-CAM y el otro es el router de internet. En el caso de la latencia, RSSI y cantidad de paquetes perdidos, la prueba de medición consiste en poner en funcionamiento el módulo a diferentes distancias del router, con tiempos de prueba que se indican en la Figura 3.24:

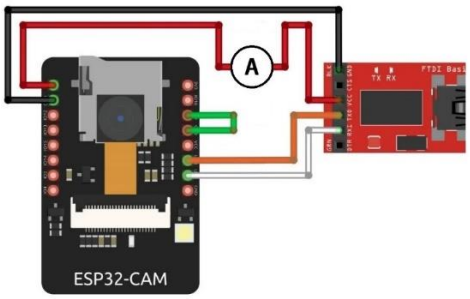
Figura 3.24. Metodología de la toma de muestras para latencia, RSSI y cantidad de paquetes perdidos.



Por otro lado, la medición de consumo de energía realiza la toma de muestras de forma externa, tomando un amperímetro y ubicándolo en serie con la conexión del ESP32-CAM

y el adaptador USB a TTL de la Figura 3.8 La conexión adicional del amperímetro se observa en la Figura 3.25:

Figura 3.25. Amperímetro conectado en serie con el módulo ESP32-CAM,



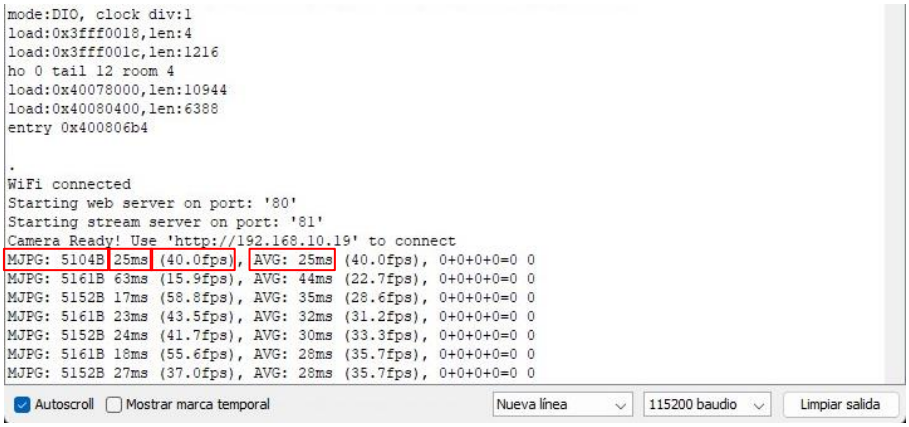
Adaptado de [30]

Ya planteado el principio de medición de parámetros para la RIC, se procede a obtener las muestras para cada una de las variables como se detalla a continuación:

3.6.2.1. Medición de latencia

Inicialmente se realiza la conexión del módulo al computador de acuerdo a la Figura 3.3. Luego se programa el módulo ESP32-CAM en Arduino con código referente a la transmisión de imagen, es puesto en operación y finalmente se abre el monitor de recursos, donde se genera un conjunto de datos que como se indica en la Figura 3.26:

Figura 3.26. Monitor de recursos con datos referentes a la transmisión de imagen.

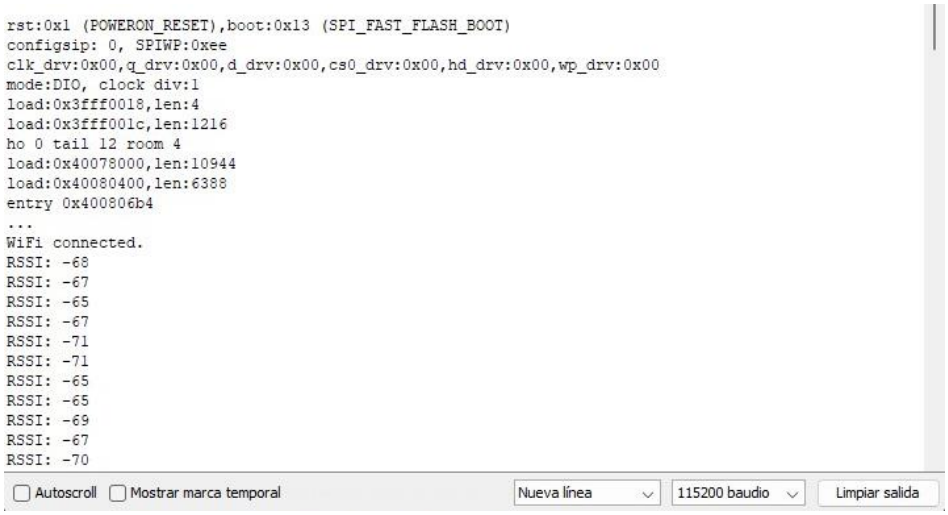


Los datos incluidos por la Figura 3.26 son, de izquierda a derecha: 1) cantidad de cuadros por segundo de la cámara (FPS), 2) tamaño de imagen resultante por cuadro (en bytes), 3) latencia (variable de interés, en ms) y 3) el promedio de latencia de acuerdo al total de muestras tomadas (en ms).

3.6.2.2. Medición de RSSI

Se realiza el mismo procedimiento que en la medición de latencia, con la diferencia de que código empleado en este caso es específico para obtener el RSSI. Además, el intervalo de tiempo entre cada muestra es de 0.2 s. Su visualización se permite también el monitor de recursos, como se observa en la Figura 3.27:

Figura 3.27. Monitor de recursos generando valores de RSSI, en dB.



3.6.2.3. Medición de paquetes perdidos

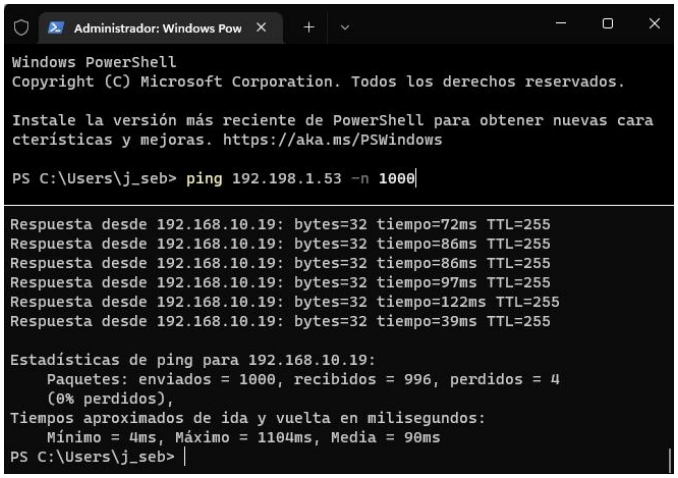
La medición de esta variable se logra programando el módulo ESP32-CAM con un código que haga un ping desde el módulo hacia el router, con el objetivo de mostrar una dirección ip en el monitor de recursos, como se ve en la Figura 3.28:

Figura 3.28. Información del monitor de recursos al hacer un ping a internet.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4

Connecting to WiFi
.....................................
WiFi connected with ip 192.168.1.53
Pinging host www.google.com
Success!!
```

Esta IP (192.168.1.53) se utiliza en el símbolo de sistema del computador, que finalmente mostrará información de envío y recepción de paquetes entre los nodos de la red. En este programa, se define un comando que realiza el ping a la ip conocida un total de 1000 veces, que serán las muestras de cantidad de paquetes enviados. Al terminar la tarea el programa muestra el total de paquetes enviados, recibidos y perdidos, siendo estos últimos los datos de interés. La Figura 3.29 detalla el empleo del símbolo de sistema en el proceso de toma de muestras:

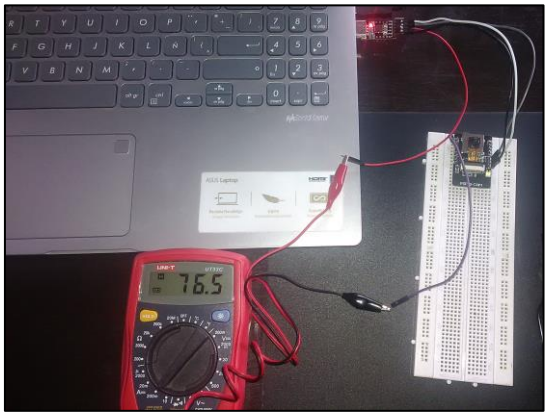
Figura 3.29. Símbolo de sistema como herramienta para obtener la cantidad de paquetes perdidos.



3.6.2.4. Medición de consumo de energía.

Como se mencionaba anteriormente, la medición de este parámetro se realiza utilizando un amperímetro en serie con el módulo ESP32-CAM. En la Figura 3.30 se evidencia la conexión correspondiente, al momento de hacer las pruebas de medición:

Figura 3.30. Conexión en serie del amperímetro, para medición de corriente en la ESP32-CAM.



El criterio de comparación para este caso es, a diferencia de los parámetros anteriores (distancia), el modo de operación del módulo. Se insertan tres códigos diferentes al módulo, uno para cada prueba: Uno corresponde al modo de operación de sueño profundo, donde se espera un consumo bajo de energía; otro es el modo de envío de ping, que supone una mayor carga de trabajo y a su vez un mayor consumo de energía; y el modo de transmisión de imagen, donde se presenta el máximo consumo por la cantidad de datos que son transmitidos.

Para establecer un valor fiable de corriente en cada una de las pruebas, se realiza la recolección de muestras en un transcurso de 15 min, una cada 30 s. La Tabla 3.8 presenta los primeros 10 valores obtenidos (de un total de 30) para cada modo de operación:

Tabla 3.8. Consumo de corriente en las pruebas con el ESP32-CAM.

Consumo de corriente (mA)		
Sueño Profundo	**Envío de ping**	**Transmisión de imagen**
2,38	63,2	124
2,4	64,3	107,3
2,42	70,7	123,8
2,44	65,4	114,9
2,49	68,6	118,2
2,53	74,5	123,7
2,61	70,2	119
2,71	63,7	120,1
2,83	66,8	123,9
2,91	68,6	115,6

La Tabla 3.8 permite observar la diferencia notable en el consumo de corriente para cada modo de operación. Además, estos datos son utilizados posteriormente para cálculos de consumo de potencia y tiempo de autonomía del módulo ESP32-CAM.

4. Capítulo 4. Resultados

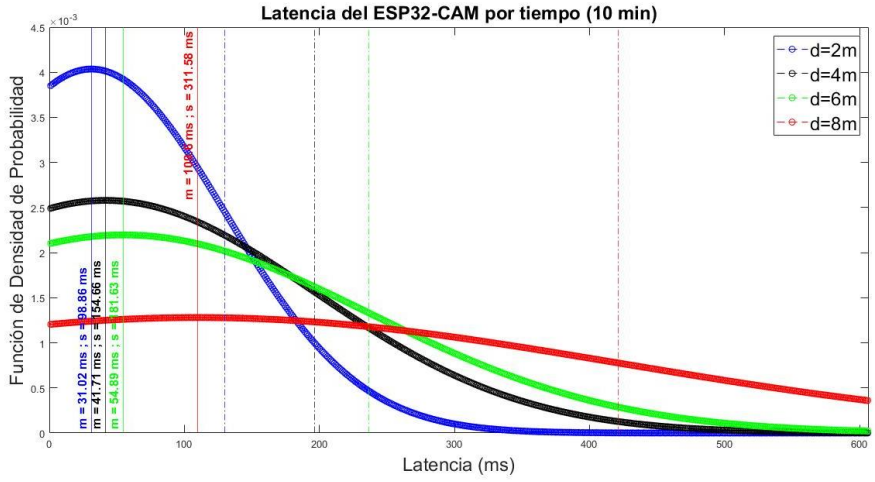
4.1. Resultados de monitoreo del entorno residencial (latencia, cantidad de paquetes perdidos, intensidad de señal recibida (RSSI))

Para cada uno de los parámetros relacionados con el monitoreo del entorno residencial, los resultados se basaron en la toma de muestras (valores numéricos), organizadas en hojas de cálculo de Excel. Posteriormente, los datos se exportaron a la herramienta Matlab, donde se escribe código que organiza la información y la presenta en gráficas que evidencian el comportamiento de cada parámetro bajo diferentes condiciones de prueba. Estos resultados corresponden a una conexión de dos nodos (ESP32-CAM y ordenador).

4.1.1. Resultados de Latencia

La Figura 4.1 presenta cuatro gráficas que organizan las muestras en una distribución normal, cada una corresponde a valores obtenidos de latencia con una distancia diferente entre los dispositivos empleados:

Figura 4.1. Latencia entre el ESP32-CAM y el ordenador por unidad de tiempo.



Cada gráfica, al ser una distribución normal, presenta un comportamiento que se explica a continuación:

- La forma característica de una distribución normal es de la denominada campana de Gauss, como se evidencia en la figura.

- El eje Y, que es la función de densidad de probabilidad de la latencia, indica la frecuencia con la que se repite cada valor de latencia obtenido. Por ejemplo, para la distancia d=2m, existe un pico en el eje Y de un valor aproximado de 31ms. Esto en una distribución normal denota una gran concentración de muestras con una latencia cercana a 31ms.

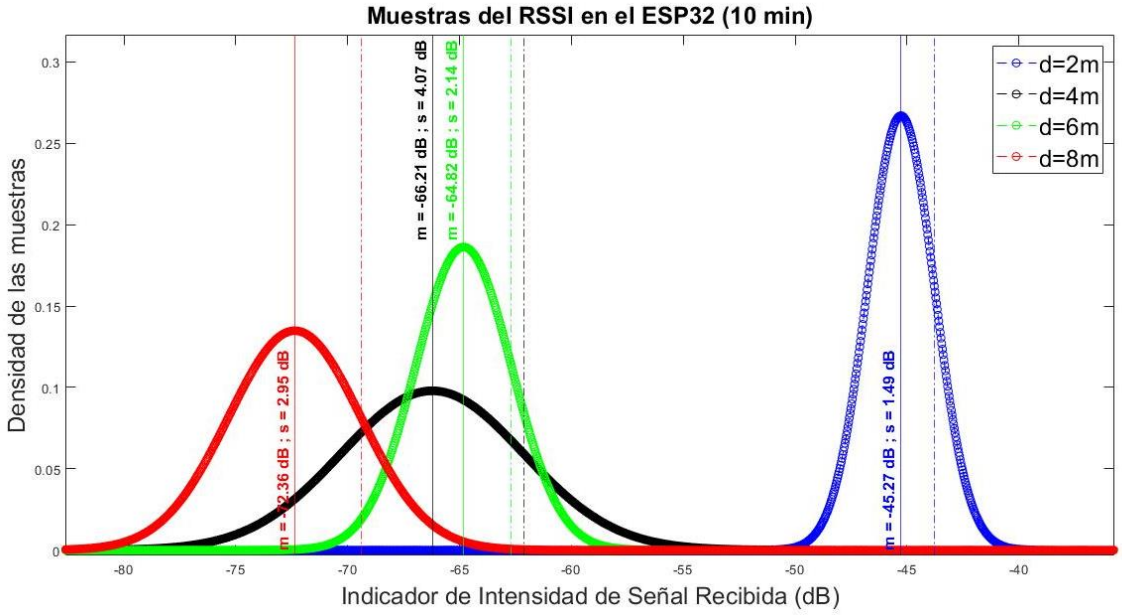
- La forma de las campanas de Gauss puede ser ancha o angosta, esto depende de la desviación estándar del total de muestras para cada gráfica. En la figura se hace muy evidente este comportamiento para cada una de las condiciones de prueba: En el caso de una distancia de 8m, una campana ancha indica una dispersión significativa en los valores de latencia. En comparación, con una distancia de 2m existe un grado de dispersión mucho menor.

A partir de lo anterior, se deduce que entre mayor sea la distancia entre dispositivos sus valores de latencia tendrán una desviación mayor, así como también el valor promedio de latencia. Desde la perspectiva del rendimiento de la RIC, un valor de latencia grande significa un deterioro en el canal de comunicaciones entre sus nodos, y viceversa para una latencia pequeña.

4.1.2. Resultados de RSSI

La Figura 4.2 presenta una distribución normal de las muestras de RSSI para cuatro pruebas diferentes, con la distancia como parámetro diferenciador en cada prueba realizada.

Figura 4.2. RSSI entre el ESP32-CAM y ordenador con respecto al tiempo.



Al igual que en la Figura 4-1, la figura 4-2 presenta cuatro pruebas distintas de toma de muestras de RSSI con gráficas de distribución normal. Por lo tanto, las consideraciones iniciales para esta figura son las mismas que en el caso anterior.

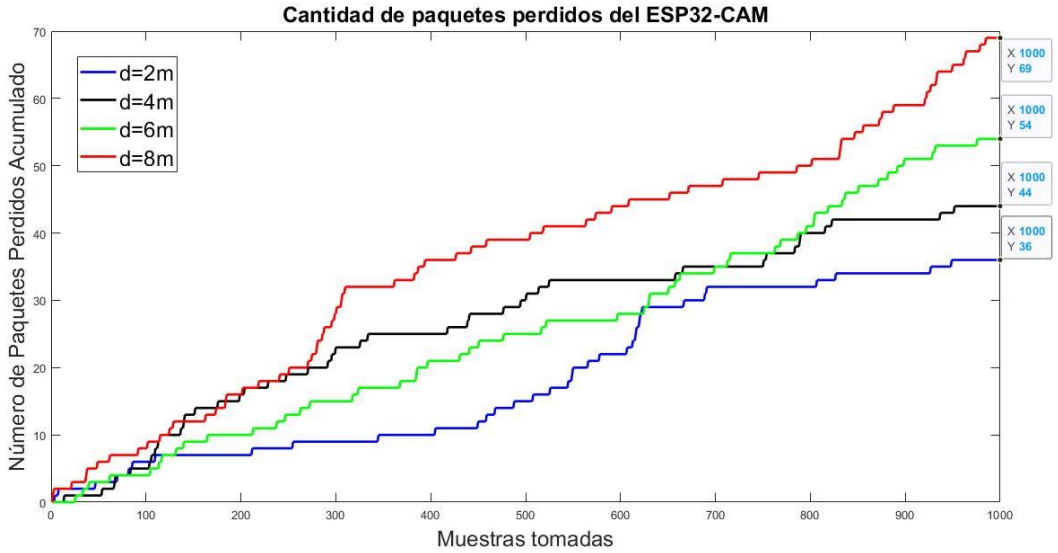
Para este caso, los valores del RSSI, expresados en decibeles negativos (-dB) son menores a medida que crece la distancia entre los nodos: Para una distancia de 2m, el RSSI promedio es de aproximadamente -45dB, continuando con valores de -66dB, -65dB y -72dB para las distancias de 4m, 6m, y 8 metros, respectivamente.

En una Red Inalámbrica de Cámaras, un valor mayor de RSSI indica una mayor intensidad de señal recibida, y la figura 4-2 presenta valores de RSSI más grandes para distancias entre nodos más cortas. Esta regla se cumple para todas las distancias a excepción de la prueba con una distancia de 4m, presentando resultados no esperados por factores típicos de un canal inalámbrico indoor (e.g., refracción, reflexión, interferencias en el canal inalámbrico y paredes). Lo anterior evidencia que la medición de este parámetro puede verse afectada por condiciones externas del entorno.

4.1.3. Resultados de paquetes perdidos

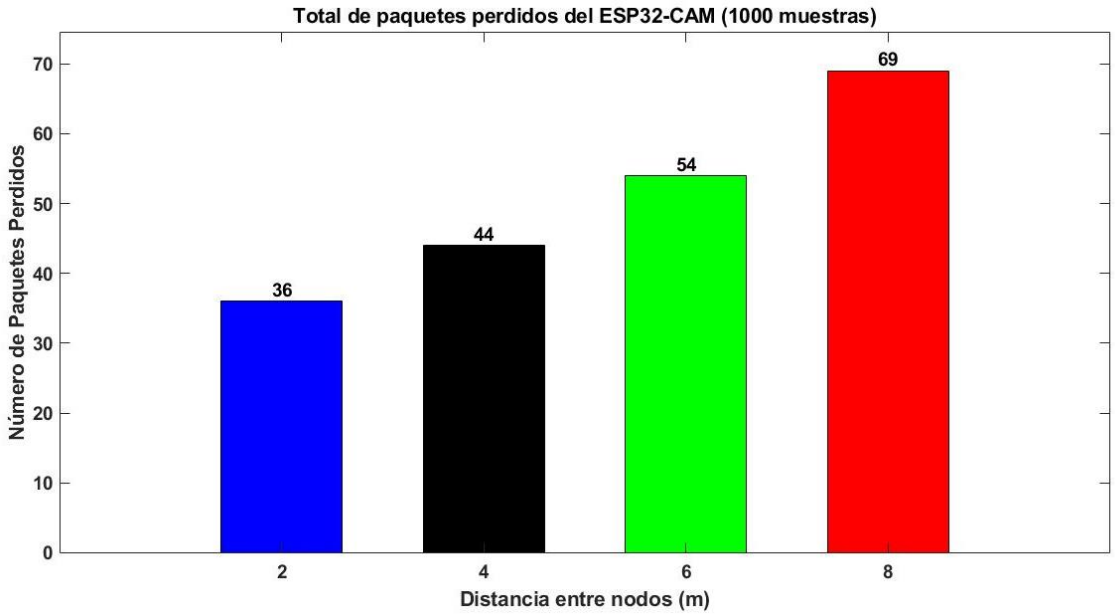
La Figura 4-3 presenta una gráfica del incremento de paquetes perdidos con respecto al total de paquetes transmitidos entre dispositivos. Se tomó un valor de referencia de 1000 paquetes totales, con la distancia como parámetro diferenciador para cada prueba.

Figura 4.3. Cantidad de paquetes perdidos entre el ESP32-CAM y el ordenador, para un total de 1000 paquetes.



Se observa en la figura 4-3 que la cantidad de paquetes perdidos en cada prueba va directamente relacionada con el parámetro de distancia, donde a mayor distancia se presenta una mayor pérdida de paquetes. La Figura 4.4 refleja este comportamiento con el total de paquetes perdidos para cada distancia medida:

Figura 4.4. Comparación entre el total de paquetes perdidos por prueba realizada.



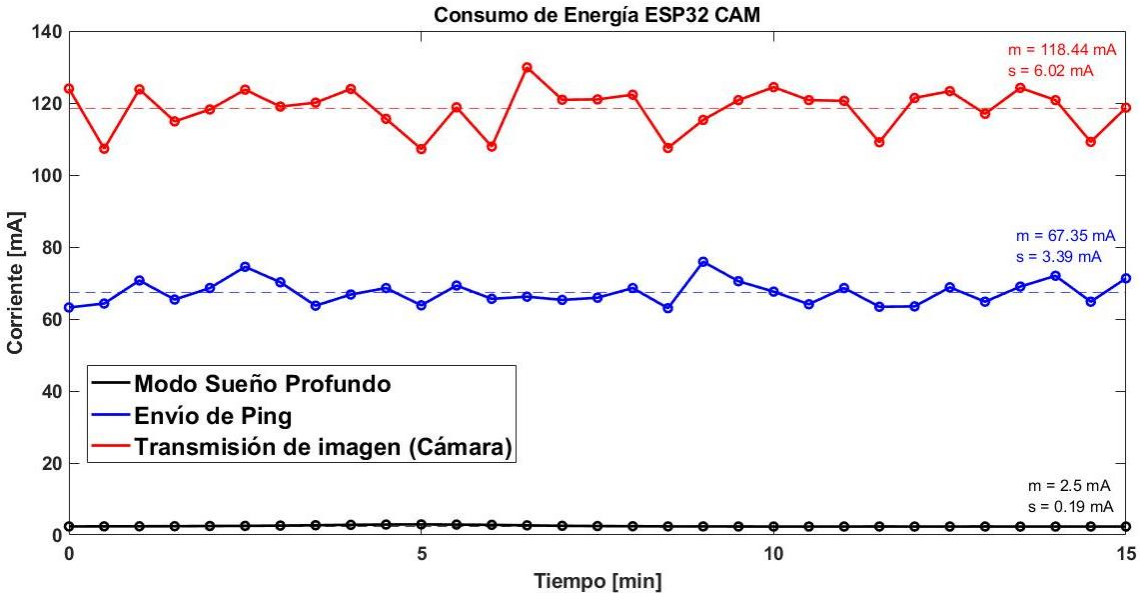
Así mismo, la figura muestra el total de paquetes perdidos para cada prueba realizada: 36 paquetes perdidos en la prueba de 2m, 44 paquetes para 4m, 54 paquetes para 6m y finalmente 69 paquetes perdidos para 8m.

En términos del rendimiento de la RIC, es desfavorable que exista una gran distancia entre sus nodos, ya que así crece la cantidad de información que se pierde al momento de ser transmitida.

4.1.4. Resultados de consumo de energía

Dado que las condiciones de prueba para el consumo de energía son diferentes a las de variables anteriores, se emplea como parámetro diferenciador el modo de operación del módulo ESP32-CAM. La Figura 4.5 presenta el consumo de energía (en mA) del módulo bajo tres modos de operación: sueño profundo, envío de ping y transmisión de imagen.

Figura 4.5. Consumo de energía de la ESP32-CAM en tres modos de operación.



La figura permite observar que los valores de corriente se hacen mayores al aumentar la carga de trabajo del ESP32-CAM: Cuando la ESP32-CAM opera en el modo de sueño profundo (bajo consumo), el consumo de corriente es de aproximadamente 2,5mA; al realizar un envío de ping (consumo medio) la corriente promedio aumenta a 67.35mA, y con el modo de Transmisión de imagen (alto consumo) los valores de corriente están cerca de los 118mA.

Teniendo en cuenta que los módulos ESP32-CAM van a cumplir sus funciones de forma inalámbrica, es necesario conocer su consumo para acondicionar una fuente de alimentación adecuada, así como para conocer su tiempo de operación. De esta manera, se considera una batería de 3,3 V como fuente de alimentación con capacidad de 1000 mAh, que proporciona los datos necesarios para obtener dos parámetros: Potencia en Watts y tiempo de operación en horas.

4.1.4.1. Potencia en la ESP32-CAM

Este parámetro se obtiene con la ecuación general de potencia:

$$P = V * I \tag{4.1}$$

Donde se considera la variable V como el voltaje de alimentación y la variable I como la corriente promedio, para así obtener:

- Modo sueño profundo:

$$\boldsymbol{P_{sleep}} = 3{,}3V * 2{,}5 \times 10^{-3}A = 8{,}25 \times 10^{-3}W = \boldsymbol{8{,}25mW}$$

- Modo envío de ping:

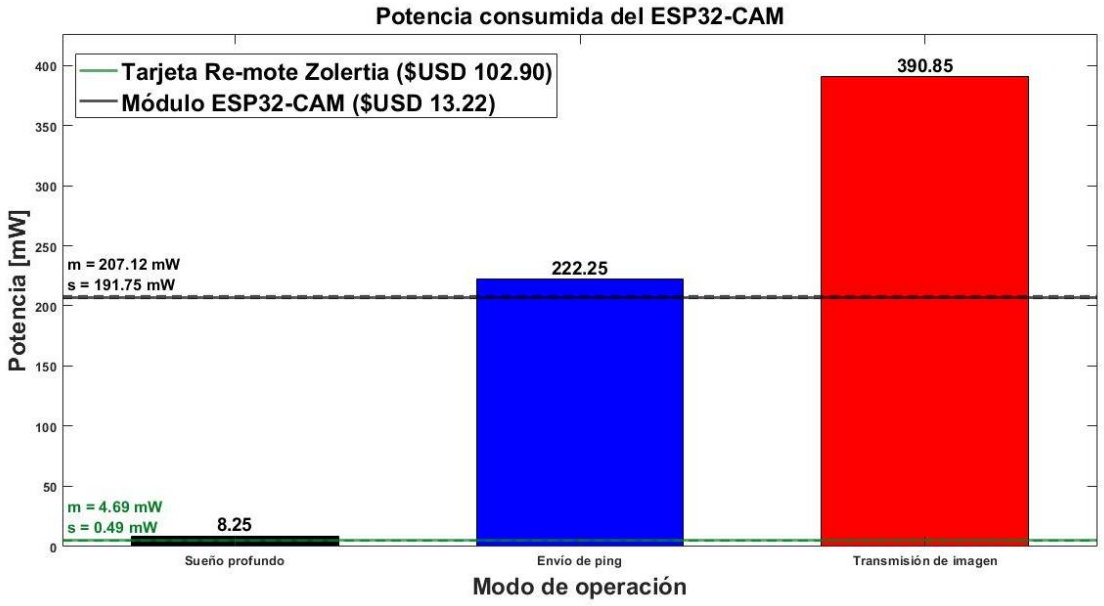
$$\boldsymbol{P_{ping}} = 3{,}3V * 67{,}35 \times 10^{-3}A = 222{,}25 \times 10^{-3}W = \boldsymbol{222{,}25mW}$$

- Modo transmisión de imagen:

$$\boldsymbol{P_{imagen}} = 3{,}3V * 118{,}44 \times 10^{-3}A = 390{,}85 \times 10^{-3}W = \boldsymbol{390{,}85mW}$$

Esta información se organiza en la siguiente gráfica de barras de la Figura 4.6:

Figura 4.6. Potencia consumida por el ESP32 - CAM, bajo tres modos diferentes de operación.



Donde se evidencia la diferencia de consumo de potencia con cada modo de operación, siendo el sueño profundo el más conveniente para emplear en el ESP32 – CAM para prolongar su tiempo de vida, parámetro que se analiza más adelante.

Por otra parte, la figura presenta una línea de consumo para una tarjeta Re-mote Zolertia, tarjeta de alta eficiencia energética tomada como elemento de comparación, que de acuerdo al artículo [31] presenta consumo de potencia promedio de apenas 4.69 mW con una desviación estándar de 0.49 mW, aproximadamente un 43% menos que el modo de operación de menor consumo para la ESP32 – CAM. Esto, sin embargo, implica un mayor coste que pone en evidencia la relación precio – rendimiento ($USD 14.7 del módulo ESP32 – CAM frente a los $USD 102.9 del Re-mote Zolertia).

4.1.4.2. **Tiempo de vida de la red**

Este parámetro, que determina el tiempo que durará el módulo ESP32 – CAM en operación, considera la capacidad de la batería que lo alimenta como variable condicionante. Para este caso, se establecen capacidades de 1000 mAh y 5000 mAh. Se aplica la ecuación 4.2:

$$t_{vida} = \frac{Capacidad\ Batería\ (mAh)}{Consumo\ (mA)} \tag{4.2}$$

Y de esta manera, se obtiene el tiempo de vida para cada modo de operación:

Con capacidad de 1000 mAh:

- Modo sueño profundo:

$$\boldsymbol{t_{operación-sleep}} = \frac{1000\ \cancel{mA}h}{2{,}5\ \cancel{mA}} = \mathbf{400\ \boldsymbol{h} \approx 16\ \boldsymbol{d}}$$

- Modo envío de ping:

$$\boldsymbol{t_{operación-ping}} = \frac{1000\ \cancel{mA}h}{67{,}35\ \cancel{mA}} = 14{,}85\ h \approx \mathbf{15\ \boldsymbol{h}}$$

- Modo transmisión de imagen:

$$\boldsymbol{t_{operación-imagen}} = \frac{1000\ \cancel{mA}h}{118{,}44\ \cancel{mA}} = 8{,}44\ h \approx \mathbf{8\ \boldsymbol{h}}$$

Con capacidad de 5000 mAh:

- Modo sueño profundo:

$$\boldsymbol{t_{operación-sleep}} = \frac{5000\ \cancel{mAh}}{2{,}5\ \cancel{mA}} = \boldsymbol{2000\ h \approx 83\ d}$$

- Modo envío de ping:

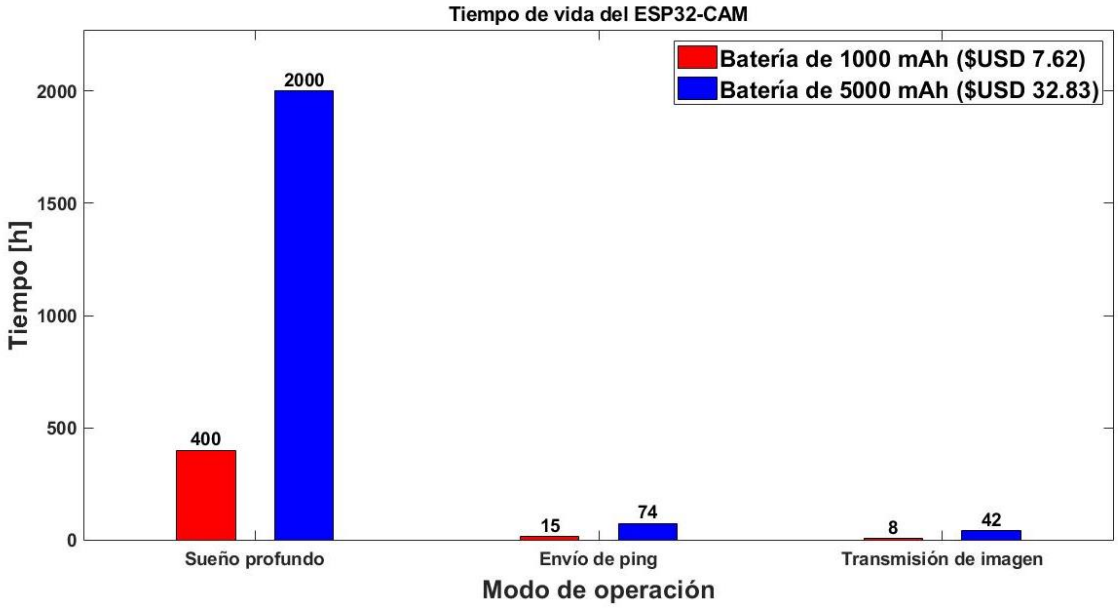
$$\boldsymbol{t_{operación-ping}} = \frac{5000\ \cancel{mAh}}{67{,}35\ \cancel{mA}} = 74{,}24\ h \approx \boldsymbol{74\ h \approx 3\ d}$$

- Modo transmisión de imagen:

$$\boldsymbol{t_{operación-imagen}} = \frac{5000\ \cancel{mAh}}{118{,}44\ \cancel{mA}} = 42{,}21\ h \approx \boldsymbol{42\ h \approx 1{,}75\ d}$$

Teniendo en cuenta estos valores obtenidos, se crea una gráfica de barras que organiza los resultados y permite realizar una comparación en el tiempo de vida del módulo usando diferentes baterías, como se observa en la figura 4-7:

Figura 4.7. Comparación del tiempo de vida de la ESP32-CAM con diferentes baterías.



De esta manera, la figura 4-7 muestra la gran diferencia entre la duración de ambas baterías, de acuerdo al consumo en cada modo de operación: En el modo de operación de sueño profundo, se tiene un tiempo de vida de 400h y 2000h, para las baterías de

1000mAh y 5000mAh, respectivamente; Para envío de ping una duración de batería de 15h (1000mAh) frente a 74h (5000mAh), y finalmente con transmisión de imagen se obtiene 8h (para 1000mAh) y 42h (para 5000mAh). Es de destacar los resultados con el modo de sueño profundo, donde se prolonga considerablemente el tiempo de vida de la red y se presenta el máximo tiempo de vida al emplear una batería de alta capacidad (5000 mAh). Así mismo, se puede apreciar el precio correspondiente para cada una de las baterías, donde una mayor capacidad implica un mayor costo (5000 mAh para un valor de $USD 32.83), pero mejor relación calidad – precio a comparación de la batería de 1000 mAh.

Desde el punto de vista del rendimiento de la Red Inalámbrica de Cámaras, lo ideal es asegurar la mayor autonomía posible de cada uno de sus elementos. Esto se logra teniendo el control sobre los dos parámetros considerados anteriormente: Una decente capacidad de batería y un bajo consumo del módulo al entrar en operación.

4.2. Resultados del sistema mecatrónico

Al igual que con los resultados del monitoreo a entornos residenciales, los parámetros del sistema mecatrónico se obtuvieron por la recolección de muestras. Sin embargo, como se observó en la sección 3.6.2, los resultados provienen de la simulación en Matlab. Las muestras en este caso son alojadas en hojas de cálculo generadas por el mismo software y su visualización sigue la misma metodología que con la RIC, por la obtención de gráficas. Para cada una de las simulaciones, del sistema mecatrónico a escala y del sistema a escala real, se evaluaron los mismos parámetros con el fin de establecer una comparación en ambos sistemas.

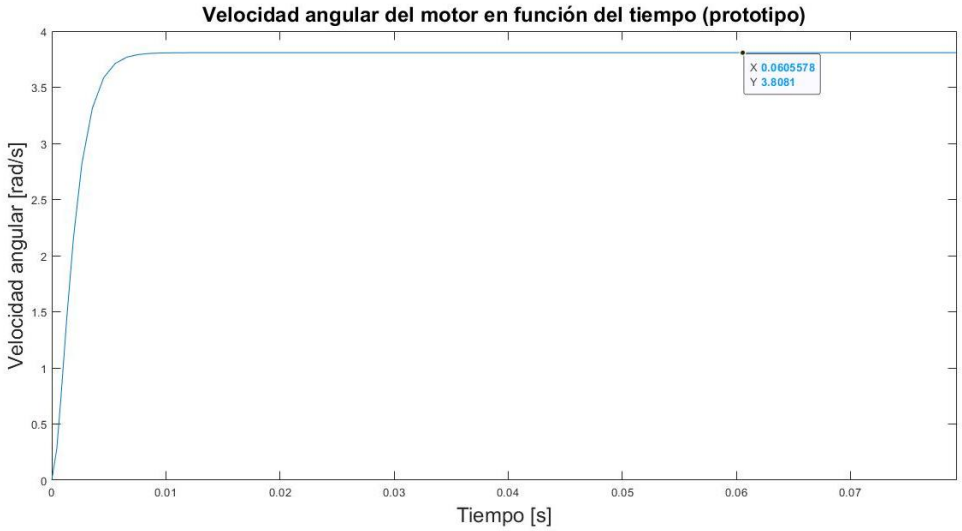
4.2.1. Resultados de la simulación a escala

Para esta simulación se fijan valores de velocidad angular $\omega_p = 3.81\ rad/s$ y la masa $m_p = 806\ g$, de acuerdo a los cálculos realizados. Estos parámetros de entrada permitieron la obtención de los parámetros de salida v_c, τ_m, F_c y d_c. De ellos se obtienen las siguientes gráficas:

4.2.1.1. Velocidad angular ω_p

La Figura 4.8 muestra la velocidad angular del motor ω_p en función del tiempo para el prototipo del sistema mecatrónico:

Figura 4.8. Gráfica de ω_p vs tiempo para el sistema a escala.

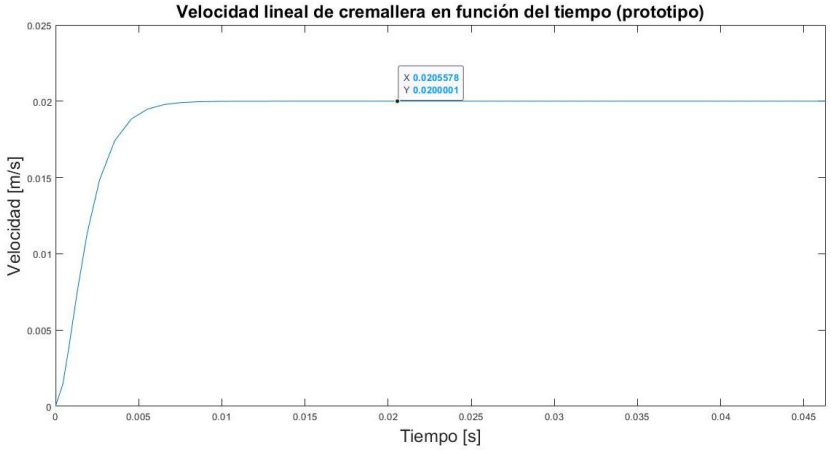


En la Figura anterior se observa cómo en los primeros $5\,ms$ existe un incremento de la velocidad angular de 0 a $3.81\ rad/s$, lo que significa un cambio de aceleración en ese periodo de tiempo. Este comportamiento se debe al arranque del sistema, que lleva al piñón del reposo a la velocidad angular requerida. Después de los $5\,ms$ ω_p se hace constante, indicando estabilidad con una velocidad angular constante y así mismo una aceleración angular de 0.

4.2.1.2. Velocidad lineal de cremallera v_c

La Figura 4.9 presenta la velocidad lineal de la cremallera como resultado de la transmisión de movimiento a partir del piñón:

Figura 4.9. Gráfica de v_c de la cremallera vs tiempo en el sistema a escala.

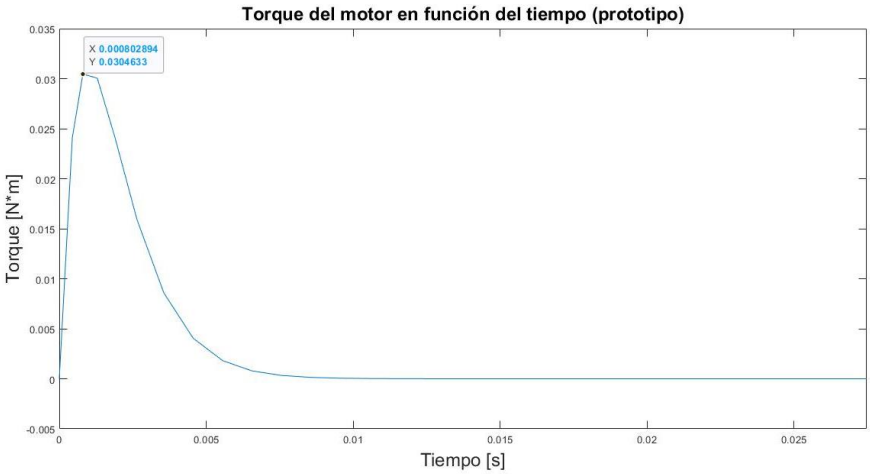


En este Figura se observa que desde el instante de $0\ s$ hasta aproximadamente $5\ ms$ existe un incremento de v_c hasta llegar al valor constante de $0.02\ m/s$. Al igual que ω_p, este incremento obedece al arranque del mecanismo desde el reposo, y su semejanza confirma la relación de velocidad angular del piñón con la velocidad lineal de la cremallera. También se evidencia que el valor de la velocidad constante de $0.02\ m/s$ tiene coherencia con los cálculos de esta misma magnitud en la sección 3.2.3.

4.2.1.3. Torque del motor τ_m

Para el sistema a escala, se obtiene un torque que se observa en la Figura 4.10:

Figura 4.10. Gráfica de torque del motor τ_m vs tiempo en el sistema a escala.

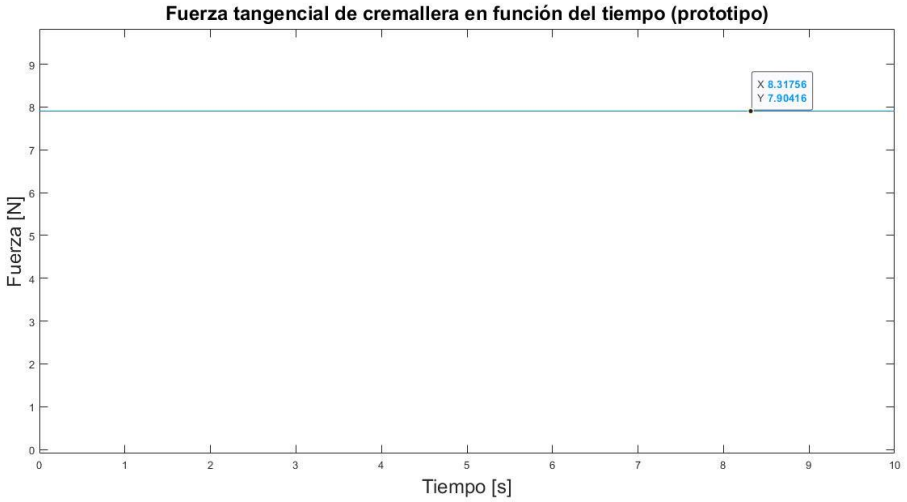


Donde el torque presenta un salto significativo al inicio de la prueba, llegando a aproximadamente $0.03\ N \cdot m$, para luego descender a 0. Este salto significa que el sistema requiere un torque inicial para ser desplazado, y conforme se establece la velocidad angular ω_p ingresada al sistema, el torque necesario para mantener en movimiento se hace menor, hasta ser casi nulo. Esto sucede en la simulación ya que no presenta condiciones físicas como fricción, que demandarían impulsos constantes al sistema para mantener la velocidad deseada.

4.2.1.4. Fuerza tangencial de cremallera F_c

Este parámetro se encuentra en la cremallera y dadas las circunstancias del mecanismo explicadas en la sección 3.2.3, también representa la carga que el sistema debe vencer para generar movimiento. La Figura 4.11 muestra este valor con el cambio en el tiempo:

Figura 4.11. Gráfica de Fuerza tangencial F_c vs tiempo en el sistema a escala.

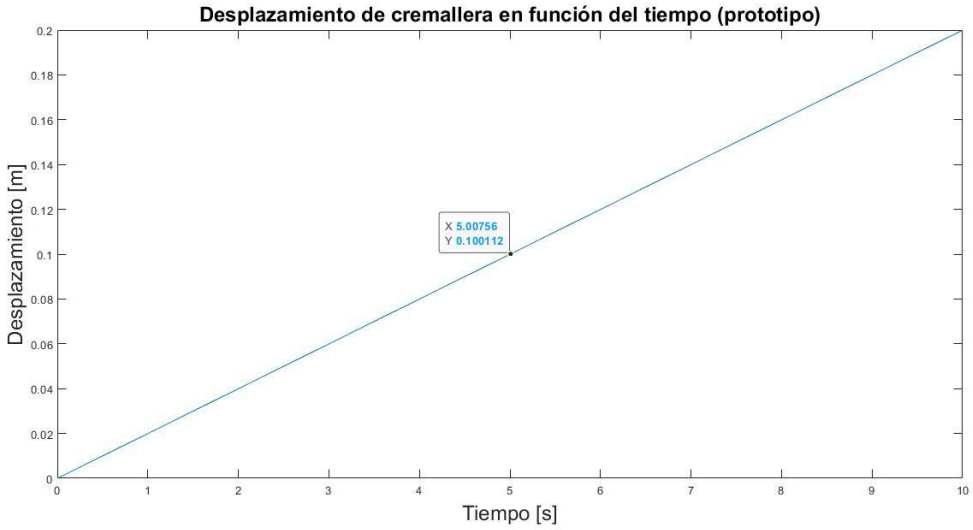


Se observa que la fuerza tangencial en todo momento es la misma, con un valor de $7.904\ N$. Esto obedece a que las variables que la generan (la masa del cuerpo $m_p = 806\ g$ y la gravedad $g = 9.81\ m/s^2$) son constantes también en todo el tiempo de prueba. Se tiene en cuenta que F_c no depende de parámetros del piñón como τ_m o ω_p, pero estos últimos si experimentan cambios a partir de una posible variación de F_c.

4.2.1.5. Desplazamiento de cremallera d_c

Esta variable describe el movimiento de la cremallera, y en la Figura 4.12 representa el cambio de la posición de esta pieza con respecto al tiempo:

Figura 4.12. Gráfica de desplazamiento d_c vs tiempo en el sistema a escala.



La Figura muestra el avance continuo de la cremallera con respecto al eje de referencia x. Esto implica que la razón de movimiento en función del tiempo es la misma en todo momento. También se observa un punto marcado con un desplazamiento aproximado de $0.1\ m$ a los $5\ s$, que corresponden a la longitud de la cremallera y el tiempo necesario para que sea desplazada en totalidad. Desde el punto de vista de un MRU, la trayectoria que describe la cremallera coincide con su velocidad lineal, donde una v_c constante implica un movimiento uniforme.

4.2.2. Resultados de la simulación a escala real

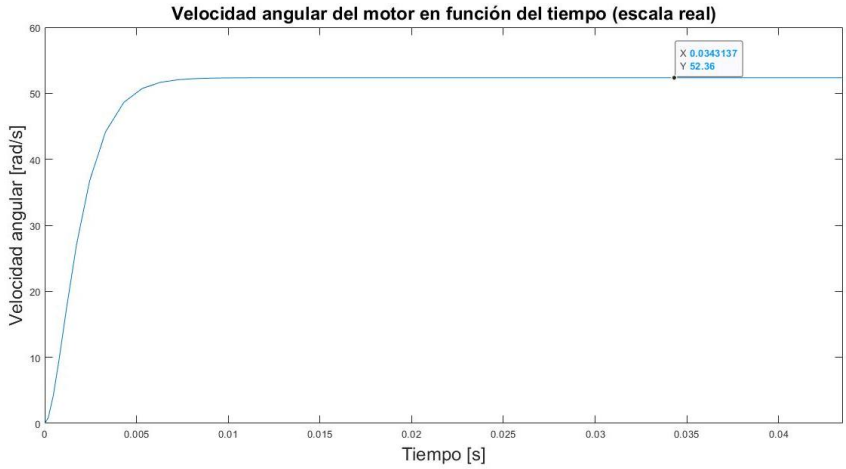
En esta simulación, cambian propiedades de la puerta como sus dimensiones $(4\ cm \times 82\ cm \times 192\ cm)$ y su masa ($m_p = 25.19\ Kg$), así mismo cambian propiedades del motor como su velocidad angular $\omega_p = 52.36\ rad/s$. A partir de estos nuevos datos, se obtienen los parámetros de salida v_c, τ_m, F_c y d_c.

En relación con la simulación a escala, el comportamiento de cada uno de los parámetros de salida es idéntico, y por lo tanto se presentan gráficas semejantes. La diferencia radica en la magnitud resultante para cada parámetro.

4.2.2.1. Velocidad angular ω_p

En la Figura 4.13 se observa el cambio de velocidad angular en el tiempo, con los parámetros del sistema a escala real:

Figura 4.13. Gráfica de velocidad angular ω_p vs tiempo a escala real.

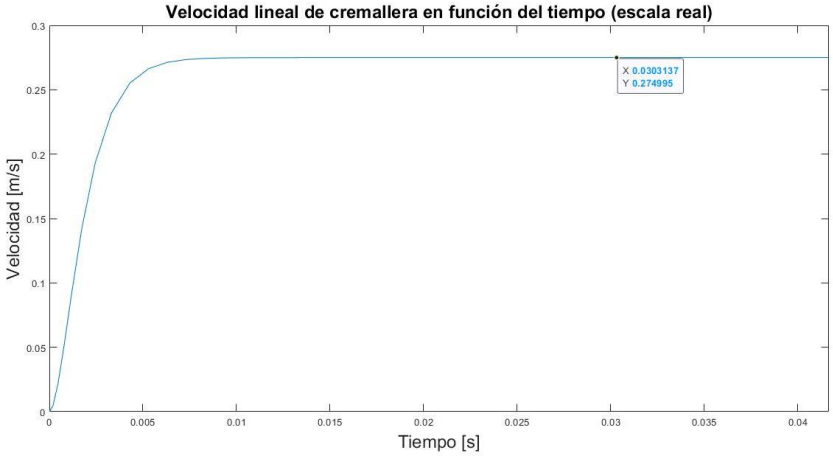


La Figura anterior muestra un comportamiento en ω_p a escala real idéntico a aquel realizado con el sistema a escala, experimentando con el tiempo una velocidad constante. La diferencia en este caso consiste en los valores obtenidos, como la velocidad angular de $52.36\ rad/s$ al hacerse constante.

4.2.2.2. Velocidad lineal de cremallera v_c

La Figura 4.14 muestra la variación de v_c con el tiempo:

Figura 4.14. Gráfica de la velocidad lineal v_c vs tiempo a escala real.

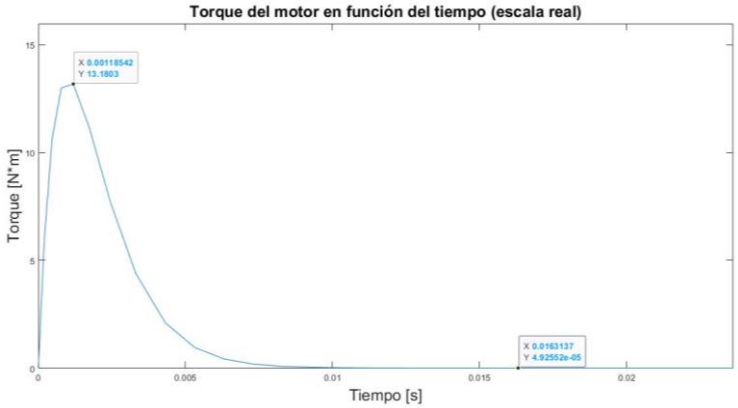


Se observa que el valor de v_c se hace constante a los $0.275\ m/s$, a diferencia de los $0.02\ m/s$ en la simulación anterior. Esta velocidad es considerablemente mayor a la de la simulación a escala, dado que el desplazamiento necesario es mayor de la misma forma. El comportamiento de la gráfica es el mismo por lo que no hay consideraciones adicionales para este parámetro.

4.2.2.3. Torque del motor τ_m

El torque resultante de la simulación a escala real tiene en cuenta los ajustes correspondientes al cambio del motor para generar un mayor torque. La Figura 4.15 muestra la variación de τ_m con el tiempo:

Figura 4.15. Torque del motor vs tiempo a escala real.

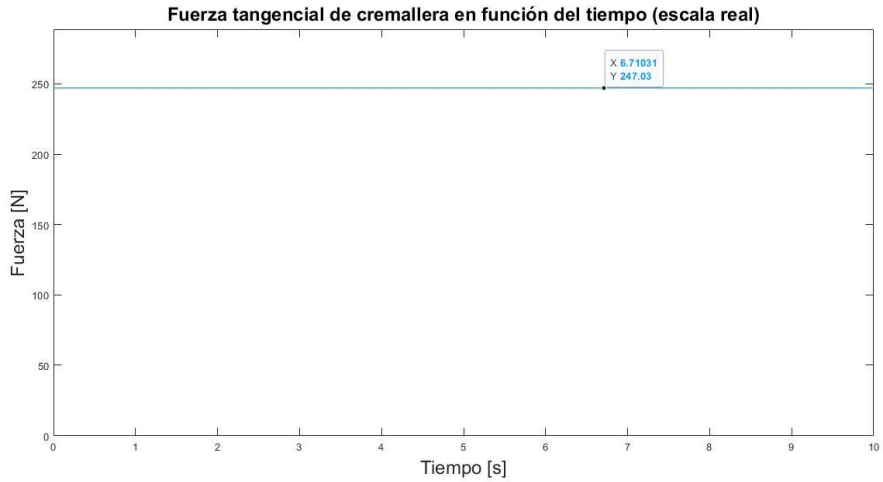


De la misma forma que en la simulación a escala el torque presenta un pico significativo en los primeros instantes de la prueba, en este caso con un valor de $13.18\,N\cdot m$. Este aumento en comparación a los $0.03\,N\cdot m$ de la prueba anterior muestran el impacto de la carga que representa la puerta, con un mayor tamaño.

4.2.2.4. Fuerza tangencial de cremallera F_c

El parámetro F_c que se obtiene a partir de la masa de la puerta m_p y la gravedad, presenta cambios con respecto a la simulación a escala, como se muestra en la Figura 4.16:

Figura 4.16. Fuerza tangencial vs tiempo a escala real.

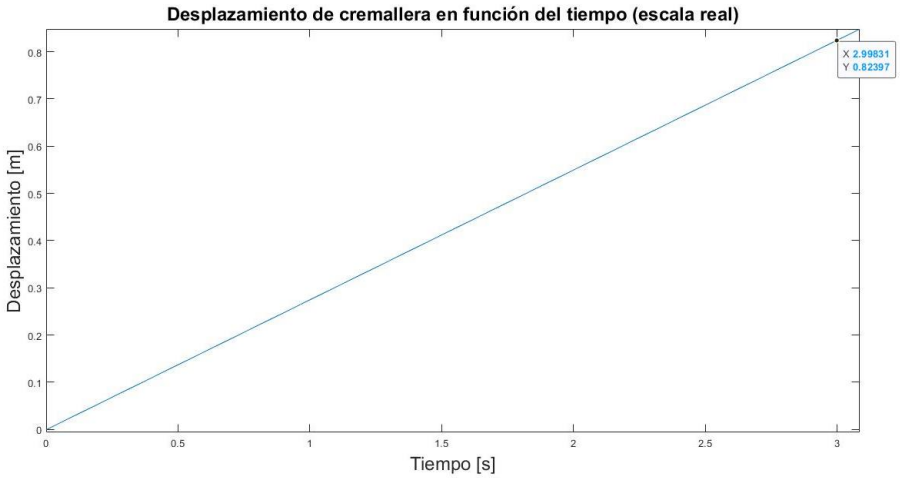


Donde se obtiene un valor constante en el tiempo de $247.03\,N$ a diferencia de los $7.904\,N$ para la simulación anterior. Este aumento en la carga requerida para mover el mecanismo tiene relación con el aumento de la masa de la puerta ($25.19\,kg$), resultante de implementar una puerta a escala real.

4.2.2.5. Desplazamiento de cremallera d_c

La cremallera para la simulación a escala real tiene una longitud mayor de $0.82\,m$ ajustándose a la necesidad de trasladar por completo una puerta de mayores dimensiones. Esta longitud se toma como el desplazamiento total requerido en el mecanismo, y los cambios en la posición de la cremallera con el paso en el tiempo se ven en la Figura 4.17:

Figura 4.17. Desplazamiento de cremallera vs tiempo a escala real.



Para esta gráfica, del tiempo de muestreo de $10\ s$ se extrajeron los primeros $3\ s$, dado que en este lapso de tiempo se presencia todo el desplazamiento posible de la cremallera. Esto se comprueba con el punto marcado donde a los $3\ s$ aproximadamente la cremallera ya ha realizado un desplazamiento de $0.82397\ m$, que es un valor cercano al ancho de la puerta a escala real.

5. Conclusiones y recomendaciones

5.1. Conclusiones

- Para las pruebas de conectividad de la Red Inalámbrica de Cámaras, basadas en la recolección de datos en el tiempo, se hizo necesario el empleo de software de gestión de datos como Excel (para el registro de las muestras tomadas) y Matlab (visualización de datos) para expresar la información obtenida en resultados que evidencien el comportamiento del sistema en función de los parámetros establecidos en un principio. En los parámetros de latencia y RSSI se generó una cantidad considerable de muestras que requirió un manejo de datos adicional con herramientas estadísticas, resultando así en las gráficas de las Figuras 4.1 y 4.2 que muestran distribuciones normales de las muestras obtenidas.

- Los parámetros evaluados la sección 4.1.1 permiten establecer las condiciones de operación de la RIC. Para los resultados de latencia, RSSI y cantidad de paquetes perdidos, las Figuras 4.1, 4.2 y 4.3 permiten comparar el rendimiento de las cámaras en la red en base a la distancia entre cada una de ellas: El primer parámetro presentó datos de 31.02ms para 2m y 109.06ms para 8m, lo que significa que menos distancia corresponde a mayor velocidad de transmisión de datos; el segundo parámetro generó un valor de -45.12dB para 2m y -72.36dB para 8m, indicando una mejor intensidad de señal recibida con una menor distancia; y finalmente, el tercer parámetro presenta valores de 36 y 69 paquetes perdidos para distancias de 2 y 8m respectivamente, lo que muestra una menor pérdida de información conforme la distancia sea más corta.

- Los resultados de consumo de energía en corriente y potencia, así como la obtención de la vida útil, permiten estimar la autonomía de operación de las cámaras y tomar decisiones sobre su ahorro energético y la selección de una fuente de alimentación adecuada: Los datos obtenidos del consumo de corriente en la Figura 4.5, de 2.5mA para el modo de operación de sueño profundo (mayor ahorro de energía) y 118.44mA para el modo de transmisión de imagen (mayor consumo), ponen en evidencia la relación directamente proporcional del uso de recursos con el gasto energético, sucediendo lo mismo con los resultados de consumo de potencia en la Figura 4.6, de 8.25mW con el modo de sueño profundo y 390.85mW con transmisión de imagen. Los valores de la vida útil de baterías de diferente capacidad mostrados en la Figura 4.7, de 400h y 2000h para el modo de sueño profundo con baterías de 1000 y 5000mAh respectivamente, presenta la relación vida útil vs. capacidad de batería, que se observa también para los modos de envío de ping y transmisión de imagen.

- El comportamiento de las gráficas resultantes del sistema mecatrónico a escala en comparación con aquellas a escala real es el mismo, debido a que en ambas condiciones de prueba se emplean las mismas bases teóricas y así también el cálculo de variables en base a las mismas ecuaciones de referencia. Por otra parte, la magnitud de las variables obtenidas sí puede tomarse como criterio para diferenciar la actuación de un modelo a escala de uno a escala real.

- La programación de los diferentes módulos de trabajo es el pilar para el desarrollo del proyecto, ya que les permite a los diferentes sistemas contar con las características planteadas y concebidas como la solución propuesta. Para la Red Inalámbrica de cámaras, configurar el módulo ESP32-CAM permitió a este sistema su disposición de nodos en topología de malla, la detección de rostros, la comunicación entre módulos y también hacia el MQTT broker. Para el sistema mecatrónico, programar el módulo ESP32-CAM concedió a este sistema características como el control de dirección y velocidad del motor, así como también recibir mensajes del MQTT broker.

5.2. Recomendaciones

En la obtención de los parámetros correspondientes al monitoreo del entorno residencial, la cantidad de muestras tomadas es fundamental en los resultados, siendo favorable recolectar una mayor cantidad de muestras para mejorar la precisión de datos numéricos y reducir el margen de error. Por lo tanto, se recomienda aumentar en la medida de lo posible la cantidad de muestras para cada variable, manteniendo un balance con el tiempo empleado para esta tarea, que aumenta de igual forma.

Se recomienda el empleo de diferentes tipos de sensores que soporten la detección de las cámaras, siendo más adecuados los sensores de detección de movimiento o los sensores infrarrojos, que pueden operar de forma complementaria a la Red Inalámbrica de Cámaras.

El diseño del sistema mecatrónico se basa principalmente en la operación de sus componentes y la aplicación final del mismo, por lo que un análisis enfocado a su optimización es recomendable. Por ejemplo, pueden ser temas de estudio aspectos como la reducción de dimensiones para lograr un sistema más compacto, la selección de componentes en torno a un mejor tiempo de respuesta y rendimiento energético, e incluso replantear el principio de operación del sistema con el fin disminuir su complejidad. También se puede considerar en la optimización el software empleado para gestionar el envío de la alarma al vigilante y su interacción con el sistema, pudiendo ser más intuitivo y garantizando una mejor respuesta del sistema en conjunto.

Anexos

A. Código en Arduino para enmallado de ESP32-CAM

```
#include "painlessMesh.h"

#define   MESH_PREFIX     "MallaNumero1"
#define   MESH_PASSWORD   "12345"
#define   MESH_PORT       5555

Scheduler userScheduler; //
painlessMesh  mesh;

//
void sendMessage() ; //

Task taskSendMessage( TASK_SECOND * 1 , TASK_FOREVER, &sendMessage );

void sendMessage() {
  String msg = "Nodo 2 (dos)";
  msg += mesh.getNodeId();
  mesh.sendBroadcast( msg );
  taskSendMessage.setInterval( random( TASK_SECOND * 1, TASK_SECOND * 5 ));
}
```

```
// Needed for painless library
void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}

void newConnectionCallback(uint32_t nodeId) {
    Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
}

void changedConnectionCallback() {
  Serial.printf("Changed connections\n");
}

void nodeTimeAdjustedCallback(int32_t offset) {
    Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),offset);
}



void setup() {
  Serial.begin(115200);


  mesh.setDebugMsgTypes( ERROR | STARTUP );  //
  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);

  userScheduler.addTask( taskSendMessage );
  taskSendMessage.enable();
}

void loop() {
  // it will run the user scheduler as well
  mesh.update();
}
```

B. Hoja de especificaciones motor FC-280PC

FC-280PC/SC

MABUCHI MOTOR

OUTPUT：0.1W～5.6W (APPROX)　　カーボンブラシ | Carbon-brush motors | 碳精电刷

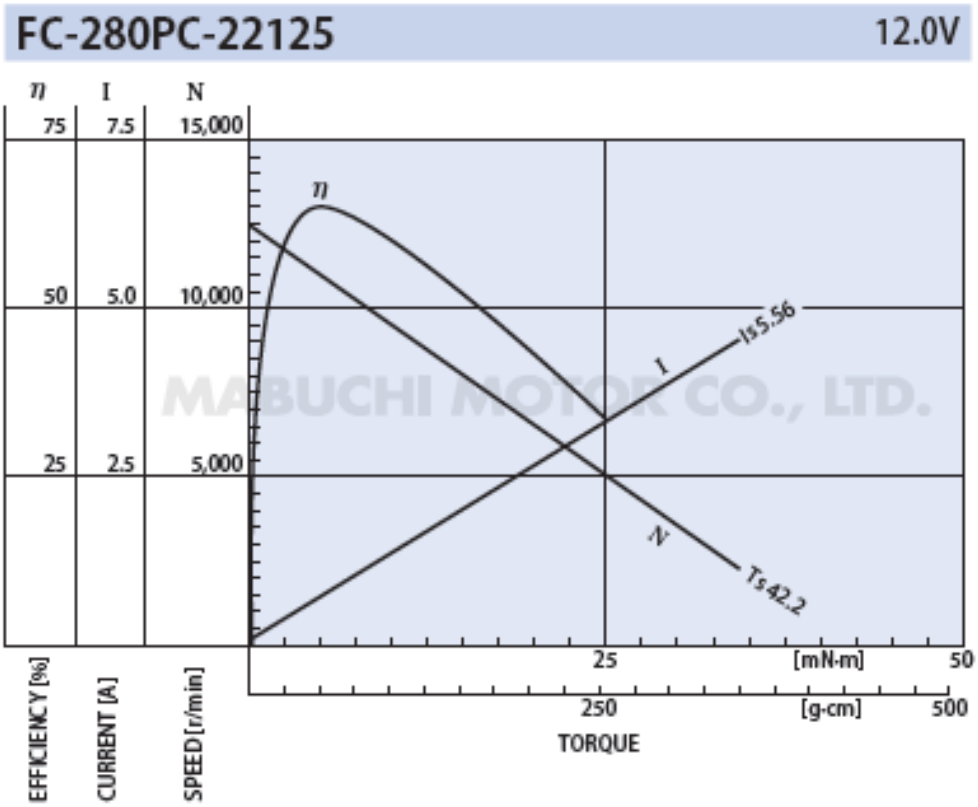
代表的用途　自動車電装機器：格納式ミラー／ドアロック／ステアリングロック

Typical Applications　Automotive Products：Retractable Rearview Mirror / Door Lock Actuator / Steering Lock

主要用途　汽车电装机器：电动折叠后视镜、车门锁、转向角限止器

WEIGHT：38g (APPROX)

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
FC-280PC-22125 (*1)	8～16	12V CONSTANT	12500	0.090	11090	0.71	4.76	48.6	5.52	42.2	430	5.56
FC-280SC-18180	8～16	12V CONSTANT	9600	0.070	8270	0.43	3.95	40.3	3.42	28.5	291	2.70
FC-280SC-20150	8～16	12V CONSTANT	11800	0.081	10310	0.56	4.14	42.2	4.47	32.9	335	3.90
FC-280SC-16220	8～16	12V CONSTANT	7900	0.055	6780	0.33	3.27	33.3	2.32	23.0	234	2.00



C. Hoja de especificaciones motor 8DCG12-25-30

DC Motor 25W(□80mm)

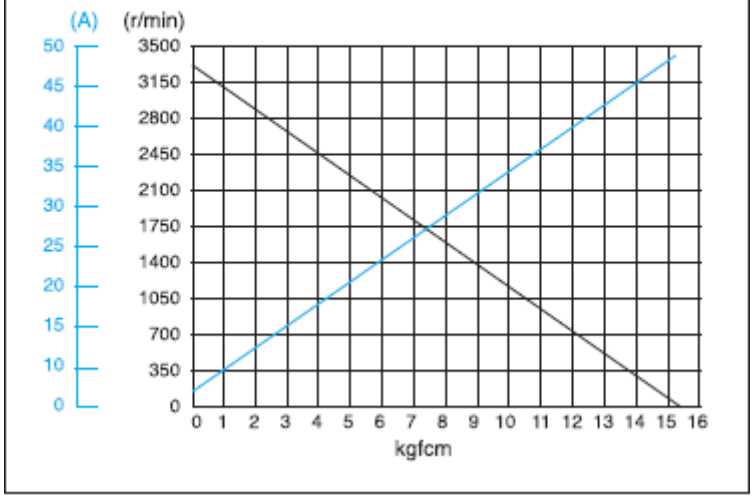
25W DC Motor 25W(□80mm)

Motor Specification

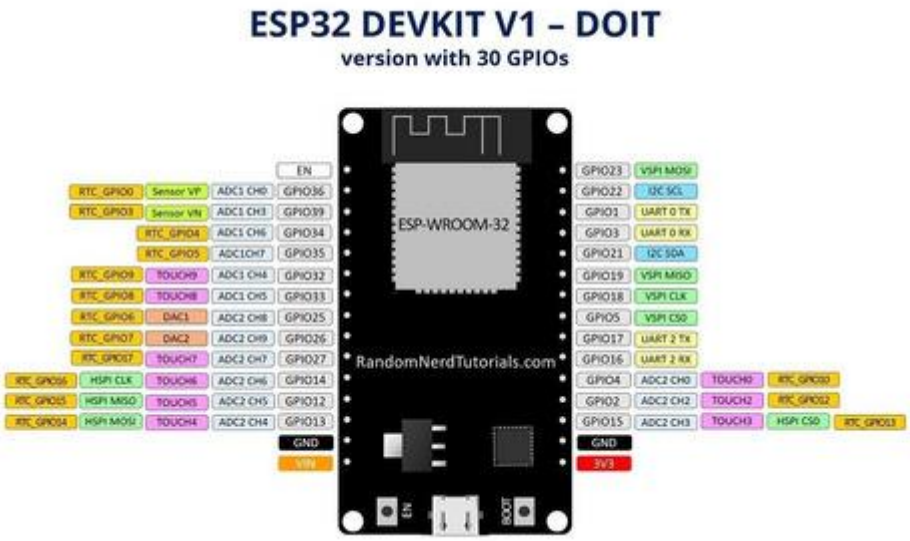
Model 8DCG(W)□-25-30: Gear Type Shaft 8DCD□-25-30: D-Cut Type Shaft	Output W	Voltage V	Starting Current A	Starting Torque		No Load		Rated Load			
				kgfcm	N.m	Current A	Speed r/min	Current A	Speed r/min	Torque kgfcm	Torque N.m
8DCG(W)12-25-30	25	12	48.00	15.50	1.500	1.80	3300	3.30	3100	0.811	0.081
8DCG(W)24-25-30	25	24	29.00	18.00	1.800	0.80	3050	1.90	2900	0.811	0.081
8DCG(W)90-25-30	25	90	10.00	21.50	2.150	0.04	3200	0.35	3000	0.811	0.081

1) Enter the phase & voltage code in the in the box (□) within the motor model name.
2) Gear Type Shaft are for attaching gearhead and D-Cut Type Shaft are for using motor only.

8DCG12-25-30



D. Diagrama de pines del módulo ESP32



Bibliografía

[1] M. U. H. A. Rasyid, F. A. Saputra y A. Kurniawan, «Surveillance Monitoring System based on Internet of Things,» *2020 International Electronics Symposium (IES),* pp. 588-593, doi: 10.1109/IES50839.2020.9231634, 2020.

[2] B. N. Rao y R. Sudheer, «Surveillance Camera using IoT and Raspberry Pi,» *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA),* pp. 1172-1176, doi: 10.1109/ICIRCA48905.2020.9182983, 2020.

[3] T. N. Nguyen, C. V. Ho y T. T. T. Le, «A Topology Control Algorithm in Wireless Sensor Networks for IoT-based Applications,» *2019 International Symposium on Electrical and Electronics Engineering (ISEE),* pp. 141-145, doi: 10.1109/ISEE2.2019.8921357, 2019.

[4] M. U. H. A. Rasyid, F. A. Saputra, Z. S. Hadi y A. Fahmi, «Beacon-enabled IEEE 802.15.4 wireless sensor network performance,» *2013 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT),* pp. 46-49, doi: 10.1109/COMNETSAT.2013.6870858, 2013.

[5] N. Vikram, K. S. Harish, M. S. Nihaal, R. Umesh y S. A. A. Kumar, «A Low Cost Home Automation System Using Wi-Fi Based Wireless Sensor Network Incorporating Internet of Things (IoT),» *2017 IEEE 7th International Advance Computing Conference (IACC),* pp. 174-178, doi: 10.1109/IACC.2017.0048, 2017.

[6] S. A. I. Quadri y P. Sathish, «IoT based home automation and surveillance system,» *2017 International Conference on Intelligent Computing and Control Systems (ICICCS),* pp. 861-866, doi: 10.1109/ICCONS.2017.8250586, 2017.

[7] Sarb y R. Bogdan, «Wireless motor control in automotive industry,» *2016 24th Telecommunications Forum (TELFOR),* pp. 1-4, doi: 10.1109/TELFOR.2016.7818790, 2016.

[8] S. M. S. R. Faisal, I. U. Ahmed, H. Rashid, R. Das, M. M. Karim y S. M. T. Reza, «Design and development of an autonomous floodgate using arduino uno and motor driver controller,» *2017 4th International Conference on Advances in Electrical Engineering (ICAEE),* pp. 276-280, doi: 10.1109/ICAEE.2017.8255366., 2017.

[9] S. Akter, R. A. Sima, M. S. Ullah y S. A. Hossain, «Smart Security Surveillance using IoT,» *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO),* pp. 659-663, doi: 10.1109/ICRITO.2018.8748703, 2018.

[10] J. Kim, S. Yu y J. Lee, «Short paper: Wireless sensor network management for sustainable Internet of Things,» *2014 IEEE World Forum on Internet of Things (WF-IoT),* pp. 177-178, doi: 10.1109/WF-IoT.2014.6803147, 2014.

[11] A. V. Oppenheim, Señales y Sistemas, Pearson Educación, 1997, pp. 38-39.

[12] D. G. Alciatore, Introducción a la mecatrónica y los sistemas de medición, McGraw-Hill, 2008, pp. 1-3.

[13] H. Torres, «Mecatrónica,» [En línea]. Available: https://hectortorresgallery.blogspot.com/2018/07/mecatronica.html.

[14] W. Bolton, Mecatrónica - Sistemas de control electrónico en la ingeniería mecánica y eléctrica, Alfaomega, 2006.

[15] «Topic – Changing Direction: Rack and Pinion,» [En línea]. Available: https://chalkboardpublishing.com/quizzes/topic-changing-direction-rack-and-pinion/.

[16] D. H. Myszka, Máquinas y mecanismos, Cuarta ed., Pearson Educación, 2012, pp. 260-282.

[17] «DB series - Motor DC by Chiaphua Components | DirectIndustry,» [En línea]. Available: https://www.directindustry.es/prod/chiaphua-components/product-61070-575546.html.

[18] «Driver Para Motor Puente H,» Arca Electrónica, [En línea]. Available: https://www.arcaelectronica.com/products/driver-para-motor-puente-h-mx1508-mini-l298n-arduino?_pos=1&_sid=8b08b321d&_ss=r.

[19] S. Electrónica, «ESP32-CAM,» [En línea]. Available: https://www.sigmaelectronica.net/producto/esp32-cam/.

[20] Y. Ye, S. Ci, A. K. Katsaggelos, Y. Liu y Y. Qian, «Wireless Video Surveillance: A Survey,» *IEEE Access,* vol. 1, pp. 646-660. doi: 10.1109/ACCESS.2013.2282613, 2013.

[21] S. Srivastava, M. Singh y S. Gupta, «Wireless Sensor Network: A Survey,» *2018 International Conference on Automation and Computational Engineering (ICACE),* pp. 159-163, doi: 10.1109/ICACE.2018.8687059, 2018.

[22] W. Dargie y C. Poellabauer, Fundamentals of Wireless Sensor Networks: Theory and Practice, John Wiley & Sons, 2010.

[23] O. Elharrouss, N. Almaadeed y S. Al-Maadeed, «A review of video surveillance systems,» *Journal of Visual Communication and Image Representation,* vol. 77, nº 103116, doi: 10.1016/j.jvcir.2021.103116, 2021.

[24] HiveMQ, MQTT & MQTT 5 Essentials - A comprehensive overview of MQTT facts and features for begginers and experts alike.

[25] R. Sithara y R. Rajasree, «A survey on Face Recognition Technique,» *IEEE International Conference on Innovations in Communication, Computing and Instrumentation (ICCI),* pp. 189-192, doi: 10.1109/ICCI46240.2019.9404387, 2019.

[26] M. Taskirana, N. Kahramana y C. E. Erdemb, «Face recognition: Past, present and future (a review),» *Digital Signal Processing,* vol. 106, 2020, doi: 10.1016/j.dsp.2020.102809.

[27] Random Nerds Tutorial, «ESP32-CAM Video Streaming Web Server (works with Home Assistant),» [En línea]. Available: https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/. [Último acceso: 29 Abril 2022].

[28] Random Nerds Tutorial, « ESP-MESH with ESP32 and ESP8266: Getting Started,» [En línea]. Available: https://randomnerdtutorials.com/esp-mesh-esp32-esp8266-painlessmesh/. [Último acceso: 1 Mayo 2022].

[29] Gitlab, «painlessMesh,» Gitlab, [En línea]. Available: https://gitlab.com/painlessMesh/painlessMesh. [Último acceso: 1 Mayo 2022].

[30] Isaac, «ESP32-CAM: lo que debes saber sobre este módulo,» [En línea]. Available: https://www.hwlibre.com/esp32-cam/. [Último acceso: 14 Mayo 2022].

[31] S. Diaz y D. Mendez, «CITT-construction of an interference-tolerant tree topology using cross-layer information,» *International Journal of Sensor Networks,* vol. 3, pp. 179-188, 2019.