Matías Gabriel Busum Fradera

Estudio de técnicas de smart IoT aplicadas a la detección de gestos para el accionamiento de dispositivos

2023

Instituto: Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática

PRÁCTICA PROFESIONAL SUPERVISADA

Informe final

Estudio de técnicas de smart IoT aplicadas a la detección de gestos para el accionamiento de dispositivos

Matías Gabriel Busum Fradera

Florencio Varela, agosto, 2023

ESTUDIANTE

Apellido y Nombres: *Busum Fradera, Matías Gabriel*

Correo electrónico: *matibf99@gmail.com*

ORGANIZACIÓN DONDE SE REALIZA LA PRÁCTICA PROFESIONAL SUPERVISADA

Nombre de la institución: *Universidad Nacional Arturo Jauretche*

Dirección: *Av. Calchaquí 6200, Florencio Varela, (1888) Buenos Aires, Argentina*

Teléfono: *+54 11 4275-6100*

Sector: *Programa Tecnologías de la Información y la Comunicación en aplicaciones de interés social (TICAPPPS), Instituto de Ingeniería y Agronomía*

TUTOR DE LA ORGANIZACIONAL

Apellido y Nombres: *Mg. Osio, Jorge Rafael*

Correo electrónico: *josio@unaj.edu.ar*

DOCENTE SUPERVISOR

Apellido y Nombres: *Ing. Salvatore, Juan Eduardo*

Correo electrónico: *jsalvatore@unaj.edu.ar*

COORDINADOR DE LA CARRERA DE INGENIERÍA INFORMÁTICA

Apellido y Nombres: *Dr. Ing. Morales, Martin*

Correo electrónico: *martin.morales@unaj.edu.ar*

Resumen

Este trabajo presenta el desarrollo y la implementación de un sistema basado en Internet de las Cosas (IoT) y visión por computadora para detectar y responder a gestos, destinado a mejorar la autonomía y calidad de vida de personas con discapacidades. El enfoque inclusivo busca empoderar a estos individuos en su interacción con su entorno. El sistema emplea una cámara para capturar imágenes que posteriormente se someten a procesamiento con el fin de identificar gestos como el alzamiento de cejas o el guiño de un ojo. El proceso de desarrollo involucró técnicas de IoT, procesamiento de imágenes y aprendizaje automático para lograr detección precisa. La arquitectura consta de un servidor, una aplicación de procesamiento de imágenes, varios microcontroladores y distintos tipos de dispositivos de visualización. La identificación de gestos activa los microcontroladores para ejecutar acciones en el entorno físico como respuesta. Los dispositivos de visualización muestran el proceso en tiempo real. Este informe detalla los componentes técnicos, herramientas, la implementación y las pruebas, demostrando la eficacia y el potencial del sistema para mejorar la calidad de vida de las personas con discapacidades.

Palabras clave: Internet de las Cosas, IoT, visión por computadora, detección de gestos, procesamiento de imágenes.

Abstract

This work presents the development and implementation of an Internet of Things (IoT) and computer vision-based system for detecting and responding to gestures, aimed at enhancing the autonomy and quality of life of people with disabilities. The inclusive approach seeks to empower these individuals in their interaction with their environment. The system utilizes a camera to capture images which are subsequently subjected to processing in order to identify gestures such as eyebrow raises or winks. The development process involved IoT, image processing, and machine learning techniques to achieve accurate detection. The architecture consists of a server, an image processing application, multiple microcontrollers, and various types of display devices. Gesture identification triggers the microcontrollers to execute actions in the physical environment as a response. The display devices showcase the process in real-time. This report details the technical components, tools, implementation, and testing, showcasing the effectiveness and potential of the system in improving the quality of life for people with disabilities.

Keywords: Internet of Things, IoT, computer vision, gesture detection, image processing.

Dedicatorias y agradecimientos

Quiero expresar mi más profundo agradecimiento a la Universidad Nacional Arturo Jauretche por brindarme la invaluable oportunidad de formar parte de su comunidad académica. Mi recorrido en la universidad ha sido de vital importancia para mi desarrollo tanto como estudiante, como en mi búsqueda de un camino hacia el crecimiento profesional.

Es imprescindible extender mi gratitud a mis tutores, Juan Salvatore y Jorge Osio, quienes me han guiado con su experiencia y conocimientos en el marco de la práctica profesional supervisada (PPS). Asimismo, deseo expresar mi agradecimiento hacia Martín Morales por su dedicado esfuerzo en la organización de la PPS.

No puedo dejar de reconocer y agradecer a todos mis compañeros, amigos y profesores que han sido una parte fundamental de mi travesía educativa. Juntos hemos compartido conocimientos, desafíos y momentos invaluables que han contribuido significativamente a mi formación integral.

Finalmente, quiero dedicar un espacio especial a mi familia y a todas aquellas personas cercanas que me han brindado un apoyo inquebrantable a lo largo de este camino. Su constante compañía y aliento han sido el motor que impulsó esta etapa de transformación y crecimiento dentro de mi vida académica.

Índice

Índice de figuras

1. Introducción

El presente trabajo se centra en el desarrollo e implementación de un sistema empleando tecnologías de Internet de las Cosas (IoT) y visión por computadora, el cual utiliza una cámara para capturar imágenes y detectar gestos. El propósito fundamental de este sistema es permitir que personas que padecen de discapacidades como parálisis, lesiones medulares, cuadriplejía o alguna condición que les impida hablar, puedan interactuar y ejecutar acciones mediante gestos detectados por la cámara.

La importancia de este proyecto radica en su enfoque inclusivo y en cómo puede mejorar la vida de las personas con discapacidades, proporcionándoles una herramienta tecnológica que les ayude a controlar su entorno de una manera más autónoma.

El proceso de desarrollo ha involucrado la evaluación de diversas técnicas y herramientas de IoT, procesamiento de imágenes e inteligencia artificial. El objetivo ha sido desarrollar una implementación lo más eficiente y precisa posible, reduciendo al mínimo los errores, como los falsos positivos y falsos negativos en la detección de gestos.

La arquitectura de la aplicación se enmarca dentro de un sistema IoT, el cual se encuentra dividido en cuatro partes principales: el servidor, la aplicación de procesamiento de imágenes, los microcontroladores y los dispositivos de visualización.

La aplicación de procesamiento de imágenes es el núcleo del sistema y se encarga de analizar las imágenes capturadas por una cámara en busca de gestos realizados por el usuario. Para efectuar esta tarea, se emplean técnicas y algoritmos de visión artificial y aprendizaje automático, los cuales han sido evaluados cuidadosamente para asegurar una detección precisa y eficiente de los gestos. Entre los gestos que puede detectar se incluyen: levantar las cejas, guiñar el ojo izquierdo, guiñar el ojo derecho, hacer una mueca con la boca hacia la derecha, hacer una mueca con la boca hacia la izquierda, formar un beso con los labios y abrir la boca.

Una vez que la aplicación de procesamiento de imágenes detecta un gesto en una imagen, envía un mensaje de control al servidor. El servidor actúa como intermediario (bróker) y se encarga de distribuir los mensajes recibidos a los microcontroladores que gestionan los dispositivos del entorno real. Estos microcontroladores son los encargados de ejecutar acciones concretas en respuesta a los gestos detectados. Por ejemplo, si el gesto indica encender una luz o abrir una

puerta, el servidor transmitirá el comando al microcontrolador correspondiente para llevar a cabo la acción requerida.

El sistema proporciona una interacción fluida entre el usuario y su entorno a través de dispositivos de visualización. Estos dispositivos, que pueden ser smartphones, tablets, computadoras u otros dispositivos con pantalla, muestran en tiempo real el proceso de procesamiento de imágenes y los datos generados. Así, el usuario puede tener un seguimiento directo de cómo sus gestos son interpretados y cómo se traducen en acciones que afectan al mundo real.

A lo largo del informe, se describirán en detalle las características técnicas de cada componente del sistema IoT y cómo se ha llevado a cabo su implementación. Se detallará la selección de las herramientas y tecnologías utilizadas, justificando las decisiones tomadas para lograr la eficiencia, precisión y facilidad de uso del sistema. Asimismo, se incluirán resultados de pruebas y evaluaciones que demuestren la funcionalidad y rendimiento del sistema.

1.1. Planteo del problema

El sistema desarrollado con tecnologías de Internet de las Cosas (IoT) y visión por computadora, aborda una problemática crucial en la sociedad: la falta de autonomía y comunicación para las personas que padecen discapacidades, como parálisis, lesiones medulares o cuadriplejía, así como aquellas que ven sus capacidades reducidas. Estas limitaciones físicas dificultan su interacción con el entorno y la ejecución de acciones cotidianas, lo que puede llevar a una sensación de dependencia y frustración en su vida diaria.

El propósito del sistema es proporcionar una solución inclusiva y tecnológica que empodere a las personas con discapacidades, permitiéndoles interactuar y realizar acciones mediante gestos detectados por una cámara. Esta innovadora herramienta tecnológica busca mejorar su calidad de vida, dándoles la posibilidad de controlar su entorno de manera más autónoma y efectiva.

El enfoque basado en IoT y visión por computadora es clave para abordar esta problemática. La integración de dispositivos y aplicaciones brinda una forma de comunicación alternativa y accesible. Además, el uso de algoritmos avanzados de visión artificial y aprendizaje automático garantiza una detección precisa y eficiente de los gestos efectuados por el usuario, lo que resulta fundamental para asegurar una interacción fluida y efectiva.

La arquitectura del sistema, dividida en el servidor, la aplicación de procesamiento de imágenes, los microcontroladores y los dispositivos de visualización, permite una coordinación efectiva entre los componentes para lograr la ejecución de acciones en el mundo real en respuesta a los gestos detectados. La comunicación entre estos elementos facilita la transmisión de comandos y asegura que las acciones solicitadas por el usuario se lleven a cabo de manera oportuna.

1.2. Objetivos

La aplicación descrita en este informe tiene como objetivo mejorar la calidad de vida de las personas con movilidad reducida, aumentando su independencia y seguridad.

En esta sección, se describen los objetivos generales y específicos requeridos para el desarrollo de este sistema.

1.2.1. Objetivos generales

A continuación, se describen los objetivos generales que son necesarios, desde el punto de vista técnico, para el desarrollo del sistema:

- Desarrollar un sistema IoT que permita enviar y recibir acciones a través del protocolo MQTT.
- Desarrollar una aplicación que, mediante técnicas de visión artificial, detecte los diferentes gestos efectuados por el usuario y los traduzca a acciones que serán enviadas al sistema IoT.
- Desarrollar una aplicación web o móvil que le permita al cliente recibir avisos y solicitudes o un comando para ejecutar una acción.

1.2.2. Objetivos específicos

Seguidamente, se detallan los objetivos específicos que se definieron con el objeto de lograr la implementación del sistema.

- Investigar y evaluar las diferentes alternativas que se pueden utilizar para la implementación del sistema IoT.

- Analizar las medidas que se deben tomar para una comunicación segura entre los dispositivos a través del sistema IoT.
- Investigar, analizar y evaluar las distintas librerías que existen para realizar el procesamiento de imágenes o de inteligencia artificial.
- Recibir los datos proporcionados por la cámara en tiempo real para que sean procesados por la aplicación que corresponda.
- Implementar una aplicación que reciba los datos proporcionados por la cámara y determine la acción que deba efectuarse, empleando técnicas de procesamiento de imágenes o inteligencia artificial.
- Desarrollar la aplicación web o móvil de monitoreo y control que permita al usuario recibir avisos, notificaciones, solicitudes o acciones a ejecutar.
- Hacer las pruebas necesarias para comprobar el correcto funcionamiento de la plataforma.

1.3. Justificación del trabajo

El propósito de esta aplicación es ofrecer una solución tecnológica que facilite la vida diaria de las personas con discapacidad motora y de las personas que requieren de cuidados especiales. Según la Organización Mundial de la Salud (OMS), aproximadamente 1300 millones de personas, es decir, 1 de cada 6 individuos en todo el mundo, sufren de una discapacidad significativa. Las personas con discapacidad motora enfrentan diversas barreras físicas, sociales y digitales que limitan su autonomía, inclusión y calidad de vida.

Nuestro proyecto se basa en la integración de dispositivos conectados a Internet de las Cosas (IoT) y algoritmos avanzados de reconocimiento e interpretación de imágenes, también conocida como visión por computadora.

El objetivo es proporcionar asistencia, seguridad y autonomía a las personas con discapacidad motora, permitiéndoles realizar tareas cotidianas como ajustar la temperatura del aire acondicionado, encender o apagar las luces, abrir o cerrar puertas y ventanas, y comunicarse con otras personas. La solución propuesta busca eliminar estas barreras y brindarles una mayor independencia y bienestar.

La relevancia tecnológica de este proyecto radica en la adopción de dos tecnologías emergentes: el Internet de las cosas (IoT) y el reconocimiento e interpretación de imágenes mediante visión por computadora. Estas tecnologías ofrecen un gran potencial para generar un impacto positivo en diversos ámbitos sociales.

Mediante el uso de dispositivos conectados a Internet, la solución tecnológica busca brindar a los usuarios un control más fácil y accesible sobre su entorno. Con la capacidad de ajustar la temperatura del aire acondicionado, encender o apagar luces, abrir o cerrar puertas y ventanas, y establecer comunicación con otras personas, se empodera a las personas con discapacidad motora y a los adultos mayores para que puedan llevar a cabo estas tareas de manera autónoma y sin dificultades.

En consecuencia, el desarrollo de esta solución tecnológica representa una iniciativa con un profundo impacto social, al abordar las dificultades y limitaciones que enfrentan las personas con discapacidad motora y los adultos mayores. La aplicación de tecnologías emergentes como IoT y visión por computadora demuestra una fuerte apuesta por la innovación tecnológica en beneficio de los sectores más vulnerables de la sociedad.

2. Marco de estudio

El propósito de este trabajo consiste en desarrollar un sistema enfocado en mejorar la calidad de vida, seguridad e independencia de personas con discapacidad motora. Para lograrlo, en esta sección se abordarán dos aspectos fundamentales que permitirán comprender las tecnologías involucradas y el estado actual de la técnica: marco teórico y antecedentes.

El marco teórico se basa en una revisión bibliográfica de las tecnologías empleadas en este trabajo. En este sentido, se definirán y analizarán los siguientes conceptos clave: tecnologías de la información y comunicación (TIC), internet de las cosas (IoT), interfaz cerebro-computadora (BCI), inteligencia artificial (AI) y visión por computadora.

En la sección de antecedentes, se realiza una revisión concisa de la información preexistente y reciente en relación con los temas tratados en esta investigación.

2.1. Marco teórico

2.1.1. Tecnologías de la Información y la Comunicación (TIC)

Las tecnologías de la información y comunicación (TIC) engloban un conjunto de herramientas, dispositivos, programas y servicios que permiten la comunicación, el procesamiento y la transmisión de información a través de medios digitales. Estas tecnologías han revolucionado la manera en que las personas interactúan entre sí y con el mundo que los rodea.

El origen de las TIC se encuentra en la evolución de las tecnologías de información a lo largo de la historia. La aparición de la escritura, la invención de la imprenta y la creación del telégrafo fueron hitos importantes en el desarrollo de las TIC. Sin embargo, el punto de inflexión se dio en la segunda mitad del siglo XX con la invención de la computadora y la creación de Internet. Desde aquel entonces, las TIC han evolucionado de manera cada vez más acelerada, incorporando nuevos dispositivos, herramientas y servicios que transformaron la manera en que trabajamos, nos comunicamos y vivimos.

Hoy en día, las TIC son utilizadas en una enorme variedad de campos, que van desde la educación y la investigación científica, hasta los negocios y la administración pública. Estas tecnologías permiten el acceso a información en tiempo real, una comunicación más eficiente

y un trabajo colaborativo, lo que ha mejorado significativamente la productividad y la calidad de vida.

Entre las características más importantes de las TIC se encuentran:

- **Conectividad:** Facilitan la conexión de dispositivos y personas en tiempo real sin importar su ubicación geográfica.
- **Velocidad:** Permiten el procesamiento, almacenamiento y transmisión rápida de grandes cantidades de datos.
- **Flexibilidad:** Son altamente adaptables y personalizables, ofreciendo una amplia variedad de usos y aplicaciones.
- **Interactividad:** Son altamente adaptables y personalizables, ofreciendo una amplia variedad de usos y aplicaciones.
- **Accesibilidad:** Están diseñadas para ser cada vez más accesibles y fáciles de utilizar, lo que amplía su alcance a un mayor número de personas.

Las TIC pueden mejorar la vida de las personas con discapacidad al proporcionarles acceso a una amplia variedad de recursos, herramientas y servicios que antes les resultaban inaccesibles. Estas tecnologías pueden ayudar a superar barreras físicas y sociales que limitan su participación plena y activa en la sociedad.

Por ejemplo, existen tecnologías de apoyo, como las sillas de ruedas motorizadas y dispositivos de asistencia para la movilidad, que ayudan a las personas con discapacidades físicas a moverse con mayor facilidad y autonomía. También existen sistemas de control del ambiente que les permiten, por ejemplo, controlar la iluminación del hogar con su voz o mediante dispositivos electrónicos.

En el caso de personas con discapacidad visual, existen herramientas o software de lectura de pantalla que son capaces de leer el contenido que se muestra en la pantalla en voz alta. Las personas con discapacidad auditiva también se pueden beneficiar de estas tecnologías, las tecnologías de subtitulado automático pueden ayudar a las personas a entender mejor el contenido de videos y la televisión en tiempo real.

En conclusión, las TIC son un conjunto de tecnologías que ofrecen una gran cantidad de ventajas, como una mayor eficiencia, productividad, accesibilidad y mayor capacidad de comunicación. Además, ofrecen una gran cantidad de herramientas que pueden ser de gran ayuda para mejorar la calidad de vida de todas las personas. Las TIC pueden resultar muy beneficiosas para ayudar a las personas con discapacidad en su vida cotidiana, mejorando su calidad de vida y su capacidad de vivir de manera autónoma e independiente.

2.1.2. Internet de las Cosas (IoT)

El Internet de las Cosas (IoT) se define como una red de objetos cotidianos interconectados que pueden enviar y recibir datos a través de internet sin necesidad de intervención humana. En otras palabras, se trata de la conexión de dispositivos y objetos físicos, como sensores, electrodomésticos, vehículos, edificios y otros, a Internet para permitir la recopilación y el intercambio de datos. La tecnología de IoT se basa en la capacidad de los dispositivos para comunicarse entre sí y con servidores en la nube, lo que permite su gestión y control remoto.

En términos más simples, IoT es una tecnología que permite conectar cualquier dispositivo con capacidad de conexión a Internet y recolectar información de ellos, como sensores, cámaras, electrodomésticos y vehículos, entre otros. Estos dispositivos se comunican entre sí y con otros sistemas, lo que permite una mayor eficiencia en la recopilación y análisis de datos para mejorar los procesos y servicios en diferentes sectores, como la industria, la salud, la agricultura, la domótica, la seguridad, el transporte, entre otros.

La estructura típica de un sistema IoT está compuesta por diversas capas, que interactúan para permitir la interconexión y el funcionamiento del sistema. Estas capas son las siguientes:

- **Capa de objetos:** En esta capa se encuentran los dispositivos inteligentes, que son objetos físicos, equipos con sensores, actuadores y software que les permiten capturar datos del entorno y realizar acciones en función de la información recibida. Estos dispositivos pueden ser muy variados, abarcando desde sensores industriales hasta electrodomésticos inteligentes, cámaras de vigilancia, vehículos conectados y dispositivos médicos, entre otros.

- **Capa de red:** La capa de red es esencial para el funcionamiento de un sistema IoT, ya que se encarga de establecer la comunicación entre los dispositivos y llevar los datos hacia su destino. Aquí se encuentran los diferentes protocolos y tecnologías de red

empleados, como LoRa, Sigfox y Bluetooth, dependiendo de las necesidades y el alcance de la solución IoT.

- **Capa de servicios:** En esta capa se ubican los brókers, servidores y otras plataformas de procesamiento en la nube. Los datos generados por los dispositivos se envían a estos servicios para ser recibidos, almacenados, procesados y analizados. Los brókers actúan como intermediarios que aseguran la comunicación fluida entre los dispositivos y los sistemas que utilizan los datos. Aquí se llevan a cabo tareas de agregación, filtrado y análisis avanzado, lo que permite obtener información valiosa y tomar decisiones basadas en los datos recopilados.

- **Capa de aplicaciones:** En esta última etapa se encuentran las aplicaciones con los que los usuarios interactúan para obtener información y controlar el sistema IoT. Estas aplicaciones pueden ser interfaces web, aplicaciones móviles u otros medios de acceso que permiten monitorear en tiempo real, realizar ajustes, generar informes y analizar tendencias, entre otras funcionalidades.

Figura 2.1. Capas de una aplicación IoT.

La idea de conectar objetos a Internet no es reciente, pero el término “Internet de las Cosas” (IoT) se acuñó en 1999 por Kevin Ashton, un empresario británico y cofundador del Auto-ID Center en el MIT. El empresario estaba buscando la forma de mejorar la eficiencia de la cadena de suministro para la empresa en la que trabajaba, Procter & Gamble. Su idea era colocar etiquetas RFID (Identificación por Radiofrecuencia) en los productos y emplear sensores para rastrearlos a lo largo de la cadena de suministro, desde su fabricación hasta el transporte y almacenamiento.

Sin embargo, la idea de conectar objetos a internet tiene una historia mucho más larga. Los primeros dispositivos conectados a internet aparecieron en la década de 1980, pero no fue hasta la proliferación de la conectividad inalámbrica y el auge de los sensores en la década de 2000 que IoT empezó a ser viable a gran escala. A medida que las tecnologías inalámbricas se desarrollaron y se hicieron más asequibles, se crearon nuevas formas de conectividad y se incrementó el acceso a internet, el interés en IoT comenzó a crecer. En la década de 2010, el auge de los teléfonos inteligentes y la rápida expansión de la nube permitieron la creación de soluciones de IoT cada vez más sofisticadas y accesibles.

En la actualidad, IoT es una de las tecnologías más importantes y en rápida evolución, con un gran potencial para transformar muchos aspectos de la vida y el trabajo. Se espera que el número de dispositivos conectados a IoT siga creciendo exponencialmente en los próximos años, lo que llevará a una mayor adopción y uso de esta tecnología.

La tecnología IoT posee la capacidad de transformar múltiples aspectos de la vida humana, mejorando la calidad de vida en distintos campos, desde el hogar y la salud hasta la industria y el transporte. Un ejemplo de ello es la implementación de hogares inteligentes, donde IoT permite controlar de manera remota la iluminación, temperatura y electrodomésticos, brindando una mayor comodidad y eficiencia en el uso de recursos. En el ámbito industrial, la combinación de sensores y IoT posibilita el monitoreo, control y optimización de los procesos de producción, mejorando la calidad de los productos y servicios ofrecidos.

Además de estas aplicaciones, IoT ofrece numerosas oportunidades para mejorar la vida de las personas con discapacidad, ayudándolas a superar las barreras que enfrentan en su día a día. Por ejemplo, para aquellos con discapacidad motora, IoT puede proporcionar dispositivos de asistencia que pueden ser controlados mediante gestos, movimientos corporales o aplicaciones móviles. Esto les permite controlar dispositivos domésticos, como luces, puertas, ventanas y electrodomésticos, a través de comandos de voz y gestos, facilitando su independencia y comodidad.

De igual manera, IoT puede ser de gran ayuda para las personas con discapacidad visual, brindándoles acceso a información que de otro modo sería inaccesible. En un futuro, los dispositivos IoT podrían conectarse a cámaras de seguridad en calles y proporcionar información en tiempo real sobre el estado del tránsito y las condiciones del entorno, lo que contribuiría a mejorar la movilidad y seguridad de las personas con discapacidad visual o

auditiva. Además, dispositivos como gafas inteligentes podrían ser utilizados para proporcionar información mediante el reconocimiento de objetos, beneficiando a quienes padecen de discapacidad visual.

2.1.3. Interfaz cerebro-computadora (BCI)

Una interfaz cerebro-computadora (BCI, por sus siglas en inglés) es un sistema que permite la comunicación directa entre el cerebro humano y un dispositivo electrónico, como puede ser una computadora, un robot o una prótesis. Su objetivo principal es permitir que las personas controlen estos dispositivos usando su actividad cerebral, sin necesidad de recurrir al movimiento de los músculos o al sistema nervioso periférico.

Las BCI registran y detectan las señales producidas por la actividad cerebral para traducirlas en comandos o acciones que pueden controlar diversos dispositivos eléctricos. A lo largo del tiempo, estas interfaces han experimentado una evolución significativa en términos de técnicas y aplicaciones.

Su origen se remonta a la década del ‘70, cuando se descubrió que las señales eléctricas producidas por la actividad cerebral podían registrarse y emplearse para controlar dispositivos electrónicos.

En sus primeros años, las BCI se basaban en técnicas altamente invasivas, que implicaban la colocación de electrodos directamente en el cerebro de quienes lo utilizaban. Sin embargo, con el avance de la tecnología, se desarrollaron técnicas no invasivas que permitían registrar las señales cerebrales mediante electrodos que se colocaban sobre la superficie del cuero cabelludo.

Con el paso de las décadas, se han desarrollado aún más técnicas no invasivas para la BCI, como la magnetoencefalografía (MEG), la tomografía por emisión de positrones (PET) y la resonancia magnética funcional (fMRI).

Sin embargo, no todas las BCI se basan en la detección de las señales producidas por la actividad cerebral. Existen técnicas que involucran el uso de cámaras para monitorear los movimientos del usuario y traducirlos a acciones o comandos para la generación de los comandos. Una de estas técnicas consta en usar cámaras infrarrojas para detectar la posición de los ojos del usuario y determinar qué punto de la pantalla está mirando. Así, el usuario puede

controlar el movimiento del cursor en la pantalla y realizar acciones mediante, por ejemplo, el parpadeo.

Usualmente, las BCI que utilizan cámaras para monitorear los movimientos del usuario suelen ser menos precisas que las que utilizan técnicas de monitoreo cerebral, pero son útiles en situaciones donde la velocidad de respuesta es crítica o no se cuenta con el equipamiento adecuado. Algunas ventajas de este tipo de técnicas son:

- **Accesibilidad:** Las cámaras son más fáciles de obtener que los electrodos o escáneres cerebrales utilizadas en otras técnicas de BCI.
- **No invasivas:** A diferencia de las técnicas que requieren, por ejemplo, de electrodos en el cuero cabelludo, las BCI basadas en cámaras no son invasivas y no requieren de la colocación de ningún elemento invasivo en el cuerpo humano.
- **Costo:** En general, las BCI basadas en cámaras son más baratas que las que utilizan electrodos o escáneres cerebrales.
- **Adaptabilidad:** Pueden ser más adaptables a las diferentes situaciones que se quieren tratar, ya que las cámaras pueden detectar una gran variedad de patrones de movimiento y no se limitan exclusivamente a la actividad cerebral.

En este trabajo, se utiliza una técnica basada en el reconocimiento facial que emplea una cámara para capturar los movimientos faciales del usuario, incluyendo los movimientos de los labios y los ojos, con el propósito de traducirlos en comandos. Esta técnica resulta especialmente útil para el control de dispositivos de asistencia o del hogar, como por ejemplo, luces, cerraduras, calefacción, entre otros.

2.1.4. Inteligencia Artificial (AI)

La inteligencia artificial (AI) es un campo de la informática que se ocupa del desarrollo de algoritmos y sistemas que pueden realizar tareas que normalmente requieren inteligencia humana, como la comprensión del lenguaje natural, el razonamiento, la toma de decisiones y el aprendizaje.

La AI se fundamenta en diversas técnicas y algoritmos diseñados para imitar la inteligencia humana en la resolución de problemas. Entre ellos, se encuentran los modelos de aprendizaje

automático, redes neuronales, algoritmos de procesamiento del lenguaje natural y sistemas de lógica difusa. Generalmente, la IA se basa en el análisis de datos y la identificación de patrones para tomar decisiones y realizar tareas. Los sistemas de IA pueden aprender y mejorar con el tiempo a medida que se les proporciona más datos y retroalimentación.

A lo largo de la historia, la AI ha experimentado importantes avances y aplicaciones en diferentes áreas. Su origen se remonta al siglo XIX, cuando algunos matemáticos y filósofos empezaron a explorar la posibilidad de crear máquinas lógicas y racionales. Sin embargo, el término "inteligencia artificial" fue utilizado por primera vez en 1956 por el informático John McCarthy, durante la Conferencia de Dartmouth, considerada el nacimiento oficial de la AI como campo de estudio.

Desde entonces, la AI ha evolucionado y se ha diversificado en diferentes ramas y aplicaciones, desde el reconocimiento de patrones y el procesamiento del lenguaje natural hasta la robótica y los sistemas expertos.

La historia de la AI se puede dividir en varias etapas, marcadas por los avances tecnológicos, los desafíos científicos y las expectativas sociales. Algunos de los hitos más importantes de la AI son:

- **La máquina de Turing (1936):** el matemático Alan Turing propuso un modelo teórico de computación que sentó las bases de la informática moderna y la posibilidad de simular el razonamiento lógico mediante algoritmos.
- **El test de Turing (1950):** Turing planteó un criterio para evaluar la capacidad de una máquina de imitar el comportamiento inteligente humano mediante una prueba de comunicación verbal.
- **El coche Stanford (1979):** uno de los primeros vehículos autónomos capaces de navegar por un entorno con obstáculos sin intervención humana.
- **Deep Blue (1997):** una supercomputadora creada por IBM que derrotó al campeón mundial de ajedrez Garry Kasparov, demostrando la superioridad de la IA en el ámbito del juego.
- **Eugene (2014):** un programa informático que superó el test de Turing haciéndose pasar por un niño de 13 años en una conversación con personas.

En la actualidad, la AI se puede utilizar para ayudar a personas con discapacidad a través de una amplia variedad de aplicaciones. Entre ellas, destaca el reconocimiento de gestos, especialmente útil para aquellos con dificultades para controlar dispositivos a través de gestos convencionales, como el movimiento de manos o dedos.

En el contexto de este trabajo, la librería MediaPipe se emplea para el reconocimiento de gestos faciales. Esta herramienta de procesamiento de datos en tiempo real usa técnicas de aprendizaje automático para identificar y analizar la postura y los movimientos del cuerpo humano captados por la cámara de un dispositivo. La implementación se enfoca en la detección y seguimiento del movimiento de los movimientos realizados con el rostro, permitiendo así un control más accesible y efectivo de dispositivos electrónicos a través de gestos.

Es importante resaltar que esta misma librería puede ser empleada para detectar y seguir los movimientos de otras partes del cuerpo, como la cabeza, las manos y los dedos. Así, se abriría la posibilidad de convertir más gestos en comandos que permitan el control de dispositivos electrónicos de manera más versátil y adaptada a las necesidades de las personas con discapacidad motora.

2.1.5. Visión por computadora

La visión por computadora es una rama de la inteligencia artificial y la informática que se enfoca en enseñar a las computadoras a interpretar y analizar imágenes y videos. En otras palabras, se trata de desarrollar algoritmos y técnicas para que las computadoras puedan “ver” y entender el mundo visual de la misma manera que los humanos.

Esta disciplina se basa en una combinación de técnicas de procesamiento de imágenes, aprendizaje automático, redes neuronales y otros enfoques de inteligencia artificial, y se aplica en una amplia variedad de áreas, como la robótica, la medicina, la seguridad y la agricultura, la astronomía, entre otros. Algunos ejemplos de aplicaciones de la visión por computadora incluyen la detección de objetos y personas en tiempo real, la clasificación automática de imágenes, el seguimiento de movimientos, la identificación facial y el reconocimiento de caracteres escritos a mano.

La historia de la visión por computadora se remonta a la década de 1960, cuando investigadores comenzaron a experimentar con la idea de enseñar a las computadoras a procesar imágenes.

Sin embargo, en ese momento, la capacidad de procesamiento de las computadoras era muy limitada y la visión por computadora aún estaba en sus etapas iniciales.

Durante la década de 1970, la visión por computadora comenzó a avanzar a medida que se desarrollaron nuevos algoritmos y técnicas de procesamiento de imágenes. En 1973, David Marr, un investigador en visión por computadora, publicó un influyente libro llamado "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information", que sentó las bases teóricas de la visión por computadora.

En la década de 1980, la visión por computadora experimentó un gran avance gracias a la llegada de las técnicas de aprendizaje automático y el uso de redes neuronales artificiales. En 1986, un equipo de investigadores de la Universidad Carnegie Mellon desarrolló un sistema de visión por computadora que podía reconocer objetos en tiempo real.

Actualmente, la visión por computadora continúa avanzando gracias a la mejora constante de la capacidad de procesamiento de las computadoras y el desarrollo de nuevas técnicas y algoritmos de aprendizaje profundo.

Entre sus aplicaciones destacadas se encuentra la asistencia a personas con discapacidad a través de la detección de gestos para el control de dispositivos electrónicos. Mediante el empleo de cámaras y algoritmos de procesamiento de imágenes, es posible detectar gestos de la mano o de la cabeza, convirtiéndolos en comandos para controlar dispositivos electrónicos como computadoras, teléfonos inteligentes, etc. Esta tecnología beneficia especialmente a personas con discapacidades físicas que enfrentan dificultades para utilizar los dispositivos electrónicos de forma convencional, como aquellos que padecen parálisis cerebral, amputaciones, lesiones de la médula espinal u otras condiciones que afectan su movilidad.

2.2. Antecedentes

En los últimos años, ha habido un creciente interés en el desarrollo de sistemas basados en reconocimiento de imágenes o gestos para brindar asistencia a personas con discapacidad. Estos sistemas aprovechan tecnologías como la visión por computadora, el Internet de las cosas (IoT) y el aprendizaje automático para mejorar la calidad de vida y la autonomía de las personas con discapacidades visuales, auditivas y motoras.

En la actualidad, los conocidos asistentes virtuales como Google Home o Amazon Alexa han sido adaptados para brindar apoyo a personas con discapacidad. Estos asistentes utilizan el reconocimiento de voz para permitir que las personas con discapacidad accedan a información, controlen dispositivos inteligentes y realicen diversas tareas cotidianas mediante comandos de voz.

Dentro del ámbito de las discapacidades visuales, también se han desarrollado sistemas cognitivos basados en IoT. Estos sistemas usan cámaras y algoritmos de aprendizaje automático para reconocer objetos, personas y escenas en tiempo real, y proporcionar información auditiva a través de auriculares o superficies táctiles. De esta manera, las personas con discapacidad visual pueden obtener detalles sobre su entorno y mejorar su autonomía.

También se indagó en el desarrollo de un dispositivo IoT que conecta a personas con discapacidad auditiva con el mundo real. Este sistema emplea un guante equipado con sensores que capturan los movimientos de la mano y los traducen al lenguaje de signos. La información se transmite a un dispositivo inteligente que convierte los gestos en texto o voz, permitiendo una comunicación más fluida entre las personas con discapacidad auditiva y el resto de la sociedad.

Por otro lado, los sistemas de control por gestos basados en visión por computadora y IoT han permitido que las personas con discapacidad motora interactúen con dispositivos inteligentes mediante movimientos simples de la mano. Estos sistemas reconocen gestos como apuntar con un dedo o deslizar con varios dedos, y los interpretan como comandos para controlar luces, ventiladores, televisores y otros dispositivos.

De igual manera, se han explorado soluciones como teclados virtuales basados en procesamiento de imágenes y visión por computadora. Estos sistemas capturan los gestos del usuario utilizando cámaras 3D ópticas o CCTV y crean una imagen holográfica del teclado, brindando una interfaz interactiva que permite a las personas con discapacidad acceder a funciones de escritura y comunicación.

Se ha investigado el desarrollo de sistemas de interacción humano-computadora basados en visión por computadora y IoT. Estos sistemas permiten realizar interacciones de teclado QWERTY y ratón sobre una superficie plana utilizando un láser y una cámara. Los gestos

efectuados por el usuario se reconocen y se traducen en comandos para controlar la pantalla del dispositivo.

En conclusión, los sistemas basados en reconocimiento de imágenes o gestos han demostrado ser herramientas prometedoras para ayudar a personas con discapacidad a superar barreras, mejorar su independencia y calidad de vida. Sin embargo, es importante destacar que aún existen desafíos técnicos y de usabilidad que deben abordarse para garantizar la efectividad y la accesibilidad de estos sistemas. En el futuro, se espera que el avance en tecnologías como el reconocimiento de imágenes y gestos siga impulsando el desarrollo de soluciones innovadoras para personas con discapacidad.

En este trabajo, se indaga en el desarrollo de un sistema que usa principalmente IoT e inteligencia artificial para permitir que las personas con discapacidad motora interactúen a través de gestos faciales. Este enfoque busca aprovechar las capacidades de reconocimiento de imágenes y gestos para detectar y comprender los gestos faciales y convertirlos en comandos que permitan a las personas con dicha discapacidad controlar dispositivos y acceder a diversas funciones. Este sistema tiene como objetivo mejorar la independencia y la calidad de vida de las personas, brindándoles una forma intuitiva y natural de interactuar con el entorno tecnológico que les rodea.

3. Solución propuesta

En esta sección, se presenta el sistema que está compuesto por diversas partes y herramientas. Para garantizar un funcionamiento eficiente y coordinado, el sistema se ha dividido en subsistemas, donde cada uno desempeña una función específica y complementaria, contribuyendo al funcionamiento de la aplicación en su conjunto.

En las siguientes subsecciones, describiremos detalladamente la arquitectura de la aplicación y los distintos subsistemas que la componen.

3.1. Descripción de la arquitectura

La aplicación se basa en una arquitectura compuesta por varias partes y herramientas integradas para alcanzar exitosamente los objetivos planteados.

Para tener una visión más clara y visual de cómo se interconectan estos componentes y cómo fluyen los mensajes entre ellos, presentamos a continuación un esquema que ilustra la arquitectura del sistema y la comunicación entre las diferentes partes.

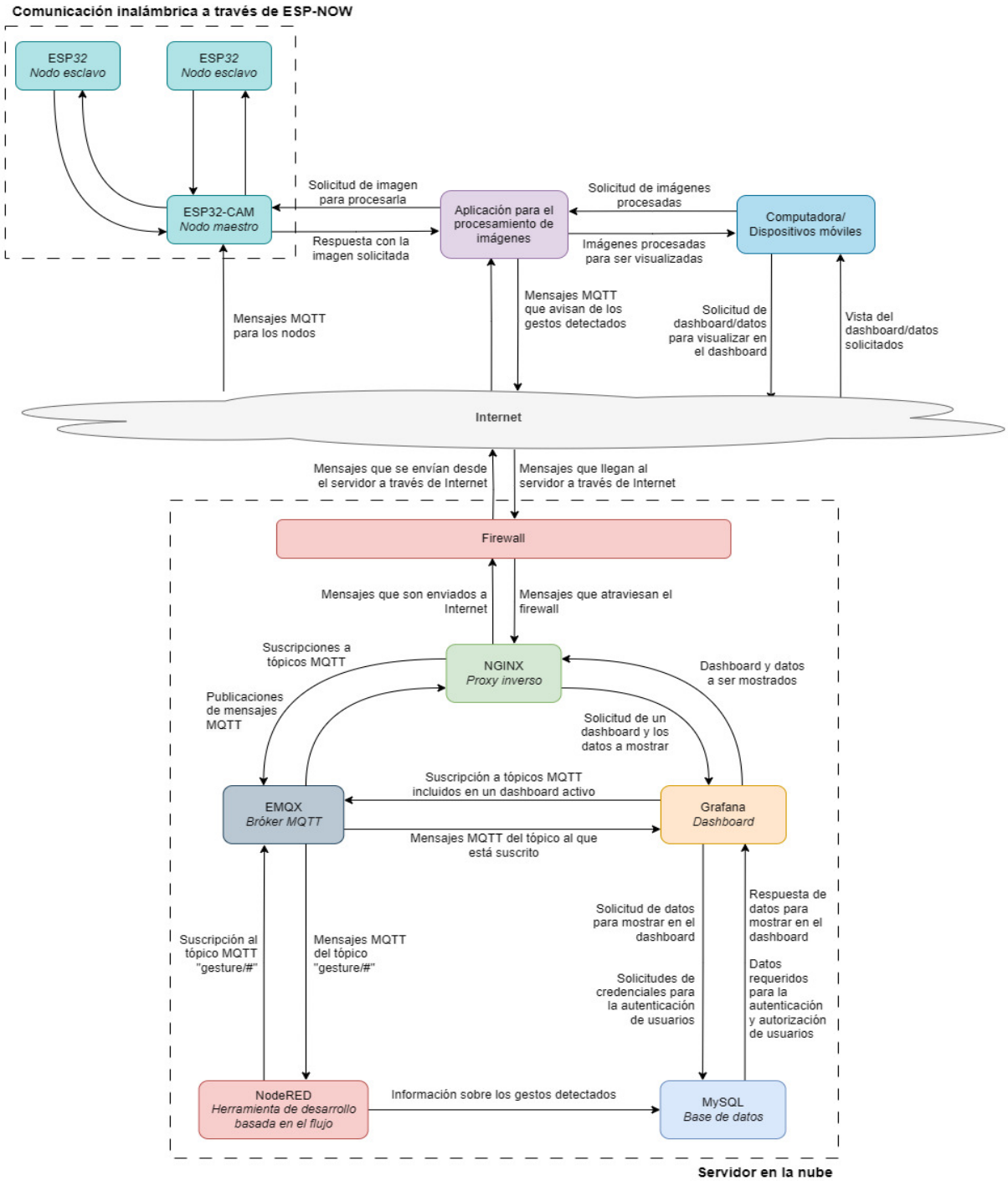

Figura 3.1. Esquema de la solución propuesta.

3.2. Subsistemas

Como se puede ver en el esquema anterior, la arquitectura de la aplicación se divide en cuatro partes principales: el servidor, la aplicación de procesamiento de imágenes, los microcontroladores y los dispositivos de visualización.

El servidor es un componente esencial en la arquitectura, aloja todas las herramientas necesarias para asegurar una comunicación fluida entre los distintos servicios y dispositivos. Además, se encarga del procesamiento, almacenamiento y visualización de toda la información relevante. Gracias al rol que cumple, se logra una interacción efectiva de todos los elementos del sistema, garantizando un rendimiento óptimo y una interacción coherente entre las diferentes partes de la aplicación.

La aplicación de detección de gestos, por su parte, juega un papel fundamental al encargarse de detectar los gestos faciales realizados por los usuarios. Cuando se identifica un gesto, este subsistema envía los mensajes de control correspondientes tanto al servidor como a los microcontroladores a través de un bróker MQTT, lo que permite que se realicen las acciones necesarias de manera coordinada. Esta aplicación actúa como la puerta de entrada para activar respuestas y desencadenar eventos que se traducen en acciones ejecutadas por otros subsistemas.

Los microcontroladores desempeñan la importante tarea de capturar los mensajes enviados por la aplicación de procesamiento de imágenes al servidor y ejecutar las acciones asociadas. Son la interfaz entre el mundo físico y la lógica del sistema. En particular, uno de los microcontroladores, el ESP32-CAM, cumple un rol adicional al proporcionar las imágenes necesarias para que la aplicación de procesamiento de imágenes pueda llevar a cabo la detección de los gestos realizados por los usuarios.

Por último, los dispositivos de visualización actúan como interfaces de usuario, permitiendo que las personas vean la información generada por el sistema y brindándoles una experiencia intuitiva y accesible. Estos permiten que los usuarios se conecten a un dashboard donde pueden visualizar un video del procesamiento de imágenes en tiempo real, junto con el estado de los gestos detectados.

3.2.1. Servidor

El servidor es un subsistema clave dentro de la arquitectura del sistema, ya que centraliza los servicios necesarios para asegurar una comunicación efectiva, la visualización de la información y el almacenamiento de datos relevantes.

Las herramientas que están funcionando dentro del servidor, como se ve en el esquema, son:

- **FirewallD como firewall.** FirewallD es una herramienta de firewall que desempeña un rol clave en la seguridad del sistema. Permite gestionar de manera sencilla las reglas de filtrado y protección de la red, lo que garantiza una protección efectiva contra posibles amenazas y accesos no autorizados.

- **NGINX como proxy inverso.** NGINX actúa como intermediario entre Internet y las aplicaciones que se encuentran instaladas en el sistema. Esta herramienta juega un papel crucial en la seguridad y optimización del servidor al gestionar la carga de solicitudes entrantes y distribuirlas de manera eficiente entre las aplicaciones que están corriendo en el servidor.

- **EMQX como bróker MQTT.** EMQX desempeña un rol fundamental al actuar como bróker MQTT en el sistema. Al utilizar el protocolo MQTT, esta herramienta permite una comunicación bidireccional entre los dispositivos y las aplicaciones, lo que facilita el intercambio de mensajes de control y acciones generadas a partir del procesamiento de imágenes. EMQX recibe los mensajes enviados por la aplicación de procesamiento de imágenes y las reenvía de manera eficiente a los microcontroladores que están a la espera de estos mensajes.

- **Node-RED como herramienta de desarrollo.** Node-RED es una poderosa herramienta de desarrollo que cumple con una función fundamental en el procesamiento y almacenamiento de los datos que son enviados a través del protocolo MQTT para su posterior análisis y visualización.

- **MySQL como motor de base de datos.** MySQL es el motor de base de datos utilizado para almacenar y gestionar los registros y datos generados por el sistema. Los datos capturados por Node-RED se almacenan en esta base de datos, lo que permite mantener un histórico de eventos y facilita el análisis posterior de la información.

- **Grafana para la visualización de datos.** Grafana actúa como interfaz de usuario o dashboard del sistema, brindando una visualización gráfica y amigable de los datos almacenados en la base de datos y aquellos que se transmiten a través del protocolo MQTT. Esta herramienta permite que los usuarios, a través de los dispositivos de visualización, puedan acceder a información acerca del funcionamiento del sistema en tiempo real.

3.2.2. Aplicación de procesamiento de imágenes

La aplicación de procesamiento de imágenes aprovecha la capacidad del ESP32-CAM para capturar imágenes en tiempo real. Emplea herramientas de visión por computadora para detectar gestos faciales en tiempo real realizados por el usuario. Gracias a la comunicación a través del bróker MQTT, otros dispositivos y aplicaciones conectados al bróker pueden reaccionar a los gestos detectados en tiempo real, lo que abre un mundo de posibilidades para la automatización e interacción con el usuario en tiempo real.

Adicionalmente, la aplicación actúa como un servidor HTTP, lo que permite a los usuarios acceder a un video en tiempo real que muestra las imágenes procesadas por la aplicación. Esta característica resulta muy útil, ya que permite a los usuarios visualizar de manera instantánea el resultado del procesamiento en tiempo real. Así, la experiencia de interacción con el sistema mejora significativamente, puesto que los usuarios pueden tener un feedback inmediato sobre los gestos detectados y cómo están siendo interpretados por la aplicación.

3.2.3. Microcontroladores

El sistema usa dos tipos de microcontroladores, el ESP32-CAM y el ESP32. Estos se conectan y forman una red interna utilizando el protocolo ESP-NOW.

El ESP32-CAM desempeña un papel fundamental al proporcionar las imágenes necesarias para el procesamiento de imágenes y la detección de gestos en la aplicación de procesamiento de imágenes. También actúa como un gateway entre Internet y la red interna.

Por otro lado, el ESP32 posee los actuadores necesarios para la ejecución de acciones al recibir algún comando. Reciben los mensajes de control enviados desde el servidor a través del ESP32-CAM. Estos mensajes contienen comandos que los ESP32 ejecutan para llevar a cabo las acciones correspondientes luego de que se haya detectado un gesto en el servidor.

3.2.4. Dispositivos de visualización

Los dispositivos de visualización están constituidos por las computadoras y dispositivos móviles (como celulares y tabletas, entre otros) que se conectan al tablero de control de Grafana a través de un navegador web. Estos dispositivos brindan a los usuarios la capacidad de monitorear y visualizar en tiempo real el funcionamiento del sistema.

A través de Grafana, los usuarios pueden acceder a distintas visualizaciones gráficas y tableros de control que muestran datos claves del funcionamiento del sistema. Permite ver información referente al procesamiento de imágenes, la detección de gestos y los mensajes de control enviados a través de MQTT.

Los usuarios pueden acceder a un video en tiempo real que muestra las imágenes procesadas por la aplicación. Esta función brinda una visión actualizada y dinámica del procesamiento de imágenes, lo que facilita el seguimiento de los gestos detectados.

4. Subsistema: Servidor

4.1. Descripción

Este subsistema es una pieza fundamental para el funcionamiento del sistema en su totalidad, ya que desempeña un papel clave en diversas tareas específicas. Sus principales funciones son la gestión de mensajes entre los dispositivos mediante el protocolo MQTT, la visualización de la información a través de un dashboard y el almacenamiento de datos.

Para garantizar un rendimiento eficiente y adaptado a nuestras necesidades específicas, se realizó una cuidadosa selección de herramientas de código abierto. La elección de estas herramientas se basa en su configurabilidad, ya que su código fuente es accesible y puede ser modificado y ajustado según las particularidades de nuestro sistema. Esta flexibilidad nos permite ajustar las herramientas de acuerdo a nuestros requisitos precisos. Además, la adopción de herramientas de código abierto también presenta ventajas económicas significativas, al reducir costos y reducir la dependencia con proveedores específicos.

En el resto de esta sección, se detallarán las distintas herramientas empleadas en este subsistema, justificando la razón de su elección y explicando su utilidad dentro del sistema. También proporcionaremos una definición clara de cada herramienta para realzar su utilidad en el sistema.

4.2. Sistema operativo: CentOS

4.2.1. Definición de sistema operativo

Un sistema operativo es una pieza imprescindible en cualquier dispositivo electrónico, como computadoras personales, servidores, teléfonos inteligentes, tabletas o sistemas embebidos. Se compone de un conjunto de programas y servicios esenciales encargados de administrar tanto los recursos de software como los de hardware, lo que le confiere un papel crucial en el funcionamiento armonioso y eficiente del sistema.

Un sistema operativo cumple con una enorme cantidad de funciones. Entre las funciones más importantes que desempeña se encuentran:

- **Administración de procesos.** Coordina y supervisa la ejecución de programas y procesos, asignando recursos y tiempos de procesador para maximizar el rendimiento del sistema.
- **Administración de memoria.** Controla y asigna la memoria disponible a los programas en ejecución, asegurando que haya suficiente espacio para la carga y ejecución de los programas.
- **Administración del sistema de archivos.** Maneja las operaciones relacionadas con la gestión de archivos, como la creación, modificación, eliminación y acceso a los archivos en el sistema de almacenamiento. Esto le permite a los usuarios organizar y acceder a sus datos de manera organizada.
- **Gestión de dispositivos.** Interactúa con los controladores de los dispositivos periféricos, como teclados e impresoras, para permitir su correcto funcionamiento y proporcionar una interfaz común para su uso.
- **Interfaz de usuario.** Proporciona una forma intuitiva de interactuar con el sistema operativo. Esto puede ser a través de interfaces gráficas de usuario (GUI) o interfaces de línea de comandos (CLI) que permiten a los usuarios ejecutar programas, acceder a archivos, configurar las opciones del sistema, etc.

Existen diferentes sistemas operativos, cada uno diseñado para funcionar en dispositivos específicos con necesidades particulares. Entre los más populares, se encuentran Windows, macOS, Linux, Android y iOS. En el contexto del desarrollo de este servidor, nos centraremos en sistemas operativos basados en Linux, diseñados específicamente para servidores.

Linux es una familia de sistemas operativos de código abierto tipo Unix, basados en el kernel de Linux, el cual fue iniciado por Linux Torvalds en 1991. A pesar de que en la jerga cotidiana la mayoría de las personas utilizan el término Linux para referirse a este sistema operativo, en realidad ese es el nombre del kernel o núcleo, que representa menos del 50% de todo el código del sistema.

Una característica destacada de Linux es su naturaleza de código abierto, lo que significa que cualquier persona puede examinar, modificar y distribuirlo de acuerdo a las licencias de

software libre. Esta filosofía ha llevado a la creación de una enorme cantidad de distribuciones Linux, que son variantes del sistema operativo original.

Cada distribución es una versión específica que incluye el kernel de Linux junto con una selección de software adicionales, agrupados y distribuidos como una unidad coherente. En la actualidad, Linux es uno de los sistemas operativos más populares, por lo que es empleado en una amplia variedad de dispositivos, como computadoras, dispositivos móviles, sistemas embebidos y servidores. Su versatilidad y naturaleza de código abierto ha contribuido significativamente al éxito y adopción en diversos campos tecnológicos.

4.2.2. Justificación de la elección

Para este servidor, se utilizó CentOS como sistema operativo debido a que el servidor proporcionado por la universidad contaba con este sistema operativo preinstalado. CentOS es una elección sólida para sistemas que buscan estabilidad, seguridad y capacidad de ejecutar aplicaciones y servicios críticos.

Una de las principales ventajas de CentOS es su estabilidad y consistencia a largo plazo. Esta distribución se basa en Red Hat Enterprise Linux (RHEL) y se ha ganado desde hace tiempo una buena reputación por ser fiable y seguro. Hasta la versión 7, CentOS era una réplica gratuita y de código abierto de RHEL, lo que la hacía una opción popular para aquellos que deseaban las características de RHEL sin tener que pagar por la licencia. Sin embargo, a partir de la versión 8, se introdujo CentOS Stream, una versión de desarrollo que se encuentra un paso adelante de RHEL.

Cabe mencionar que el servidor actual cuenta con la versión CentOS 7, pero su soporte finalizará el 30 de junio de 2024. Por lo tanto, para mantener el sistema operativo actualizado y garantizar su seguridad y funcionalidad a largo plazo, será necesario realizar una actualización a una versión más reciente.

Esta distribución usa el manejador de paquetes RPM, lo que facilita la instalación y actualización de software en el sistema. RPM es una herramienta ampliamente empleada y bien establecida dentro del ecosistema de Linux.

A pesar de que se está utilizando CentOS en este servidor, es relevante destacar que existen muchas otras distribuciones de Linux adecuadas para servidores, como Ubuntu Server, Debian

y Fedora Server, cada una con sus propias ventajas y características únicas. La elección del sistema operativo debe basarse en los requisitos específicos del servidor y la familiaridad con la distribución. Todas estas opciones pueden ser buenas alternativas según las necesidades del usuario.

En este caso puntual, se optó por CentOS debido a que cumple con las necesidades específicas de este subsistema. Su estabilidad, seguridad y capacidad para ejecutar aplicaciones y servicios críticos lo hacen adecuado para el propósito del servidor en cuestión.

4.2.3. Utilidad en el sistema

Como se mencionó anteriormente, el sistema operativo desempeña una función fundamental al gestionar y controlar los recursos del sistema, así como la ejecución y coordinación de los procesos. Su papel es administrar eficientemente el hardware y los recursos del sistema, permitiendo el funcionamiento adecuado de todas las aplicaciones y servicios que serán mencionados en las siguientes secciones.

Una de sus tareas es la administración de la memoria y el procesador durante el procesamiento de imágenes, asegurando un uso óptimo de los recursos disponibles. Además, facilita la comunicación y el intercambio de mensajes entre los diversos servicios del sistema, promoviendo una interacción fluida y eficaz entre ellos.

Asimismo, el sistema operativo se encarga de administrar el almacenamiento y el acceso a los datos, garantizando la integridad y seguridad de la información almacenada en el servidor.

Un elemento destacado es el manejador de paquetes de CentOS, que juega un papel crucial al facilitar la instalación y la actualización de las aplicaciones en este servidor. Esto agiliza el proceso de incorporar nuevas aplicaciones y servicios al sistema, lo que contribuye a mantener un entorno dinámico, seguro y actualizado.

4.3. Firewall: FirewallD

4.3.1. Definición de firewall

Un firewall, también conocido como cortafuegos, es una herramienta de seguridad informática diseñada para proteger redes de computadoras y sistemas informáticos contra amenazas y

accesos no autorizados desde redes externas, como Internet. Actúa como una barrera entre la red protegida y las redes externas, controlando el tráfico que entra y sale de la red.

Figura 4.1. Ejemplo de funcionamiento de un firewall.

Un firewall se puede imaginar cómo un guardia de seguridad que controla quién puede entrar o salir de un edificio. De la misma manera, un firewall monitorea y filtra el tráfico de red basándose en reglas predefinidas o personalizadas. Estas reglas determinan qué tipo de tráfico se permite y cuál se bloquea, según criterios como direcciones IP, puertos y protocolos, entre otros.

Al establecer estas reglas, el firewall es capaz de bloquear el tráfico malintencionado o no deseado, mientras que permite el paso del tráfico legítimo y autorizado.

Existen diferentes tipos de firewall. Sin embargo, nos enfocaremos en dos de ellos, clasificados según su ubicación de instalación:

- **Los firewalls de red.** Se ubican entre la red interna y la externa y filtran el tráfico a nivel de red.
- **Los firewalls de host.** Se ejecutan en sistemas individuales y controlan el tráfico a nivel del sistema operativo.

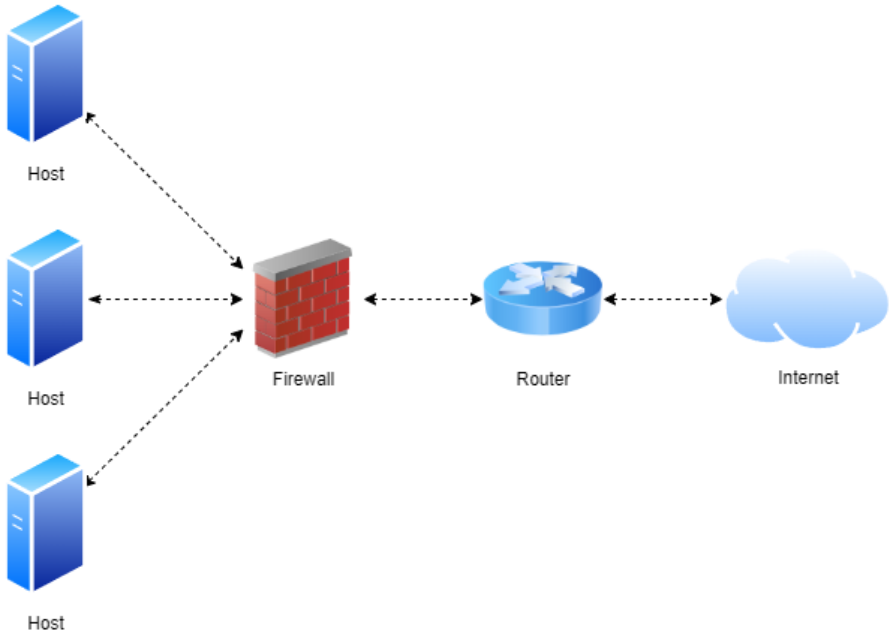

Figura 4.2. Firewall de red.

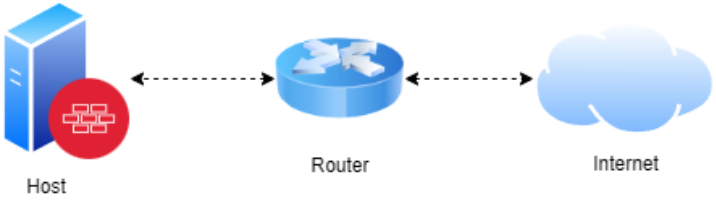

Figura 4.3. Firewall de host.

Para este trabajo solo nos centraremos en el firewall de host, ya que el firewall que utilizaremos estará corriendo en el servidor y filtrando el tráfico a nivel del kernel del sistema operativo del servidor.

4.3.2. Justificación de la elección

Existen varias alternativas de firewalls en Linux, como iptables, UFW (Uncomplicated Firewall) o FirewallD. Sin embargo, todas las alternativas funcionan como interfaces que permiten configurar y administrar las reglas de filtrado de paquetes en el kernel a través de netfilter.

Netfilter es una parte integral del núcleo de Linux y proporciona un conjunto de hooks dentro del flujo de paquetes de red. Un hook, en el contexto de Netfilter y el núcleo de Linux, es un punto de entrada o salida en el flujo de paquetes de red donde se pueden insertar módulos o extensiones para interceptar y manipular los paquetes. Todos los firewalls mencionados

anteriormente usan esta interfaz para interceptar y manipular los paquetes de red a medida que atraviesan el sistema.

Por ende, independientemente de la elección del firewall, estaremos aprovechando el mismo mecanismo subyacente en el kernel. Los firewalls mencionados anteriormente solo proporcionan interfaces de usuario y herramientas de configuración que facilitan la administración y aplicación de reglas de filtrado de paquetes en netfilter.

Después de analizar varias alternativas de firewall, hemos llegado a la conclusión de que FirewallD es la opción más recomendada para el sistema operativo que estamos utilizando en el servidor, CentOS. Hay varias razones que respaldan esta elección:

- **Incluido en CentOS.** FirewallD ya viene instalado junto con la distribución de CentOS, lo que facilita su configuración y uso sin la necesidad de instalar software adicional.
- **Simplicidad.** FirewallD está diseñado para ser fácil de usar y configurar. Permite crear y gestionar reglas de firewall de manera rápida y sencilla.
- **Documentación.** Existe documentación detallada y extensa sobre FirewallD. Es fácil encontrar recursos confiables para resolver dudas y obtener orientación en la configuración y administración de firewall.
- **Interfaz de línea de comandos (CLI).** FirewallD proporciona una interfaz de línea de comandos muy fácil de utilizar, lo que permite gestionar las reglas de firewall de forma rápida y sencilla a través de una terminal.
- **Experiencia previa.** Ya contábamos con la experiencia de haber utilizado FirewallD en servidores. Esto nos brinda una confiabilidad adicional con el software y nos permite aprovechar al máximo sus características y funcionalidades.

4.3.3. Utilidad en el sistema

Un servidor conectado a Internet está constantemente expuesto a amenazas externas que buscan acceder a sus servicios y comprometer la integridad y confidencialidad de la información que tiene almacenada. En este contexto, el firewall actúa como una barrera defensiva, estableciendo reglas y filtros para controlar el flujo de datos que ingresan y salen del servidor.

En el caso de este sistema, se ha configurado FirewallD para permitir el acceso a cada una de las aplicaciones a través del puerto 443, que es el puerto estándar empleado para HTTP y HTTPS, respectivamente.

No obstante, se ha hecho una excepción para EMQX, nuestro bróker MQTT. Además del puerto 443 para ingresar al dashboard, se ha configurado FirewallD para permitir el acceso a EMQX mediante los puertos MQTT/TCP, MQTT/WS y MQTT/WSS. Estos puertos son utilizados por el protocolo MQTT para la comunicación con los dispositivos IoT. Permitir estos puertos garantiza que el bróker MQTT pueda recibir mensajes MQTT de manera segura.

Con esta configuración cuidadosa de FirewallD, se garantiza que el acceso a los recursos del sistema esté restringido y solo se permita el tráfico legítimo a las aplicaciones que se ejecutan en el servidor. Al bloquear el acceso a otros puertos y recursos, se fortalece la seguridad en el servidor y se minimiza el riesgo frente a ataques o intrusiones no autorizadas.

Puerto	Protocolo	Aplicación
22	TCP	SSH
80	TCP	HTTP
443	TCP	EMQX Dashboard, Node-RED Dashboard, Grafana Dashboard (HTTPS)
1883	TCP	MQTT
8883	TCP	MQTT/SSL
8083	TCP	MQTT/WS
8084	TCP	MQTT/WSS

Figura 4.4. Puertos abiertos en el servidor.

4.4. Proxy inverso: NGINX

4.4.1. Definición de proxy inverso

Un proxy inverso, conocido como reverse proxy en inglés, es una herramienta que actúa como servidor intermediario en nombre de otros servicios de red.

La función principal de un proxy inverso es recibir las solicitudes de los clientes que llegan desde Internet y redirigirlas al recurso o servicio de red correspondiente, se encuentre en el mismo servidor o no. En otras palabras, actúa como un intermediario entre las peticiones de

los clientes y las aplicaciones que corren en los servidores, proporcionando una capa adicional de seguridad y privacidad.

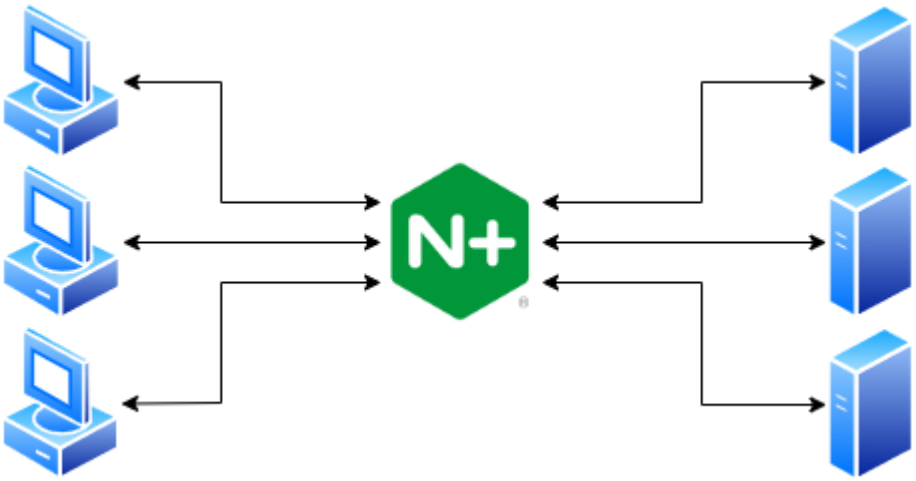

Figura 4.5. Ejemplo de funcionamiento de un proxy inverso (NGINX).

La aplicación de un proxy inverso permite simplificar la configuración de red al proporcionar un solo punto de entrada a todos los clientes que quieran acceder a varios recursos o servicios. Sin su utilización, los usuarios deberían recordar los diferentes puertos para acceder a cada servicio, lo que puede resultar muy complicado. El proxy inverso resuelve esta problemática al permitir que un subdominio se asigne a cada servicio específico.

También hay otras maneras de aplicar un proxy inverso que están relacionadas con esta, tales como:

- **Centralización del tráfico:** Al dirigir todo el tráfico de los clientes a través del proxy inverso, se elimina la necesidad de configurar y mantener múltiples puntos de entrada en la red. En lugar de tener que configurar cada servidor para recibir y procesar las solicitudes de los clientes, solo se necesita configurar el proxy inverso para hacerlo.

- **Administración de la seguridad:** Un proxy inverso puede actuar como un punto de control de seguridad centralizado. Puede realizar tareas como el filtrado de solicitudes, la inspección de paquetes, la detección de intrusos y la protección contra ataques de denegación de servicio (DoS) o de denegación de servicio distribuidos (DDoS). Al concentrar estas funciones en un solo punto, es más fácil configurar y mantener las medidas de seguridad de la red.

- **Balanceo de carga:** Un proxy inverso puede distribuir el tráfico entrante entre varios servidores. Esto permite lograr un equilibrio de carga, asegurándose de que ningún servidor esté sobrecargado mientras se utilizan de manera eficiente los recursos disponibles. Los clientes sólo necesitan conocer la dirección del proxy inverso, que se encarga de enviar la solicitud al servidor menos ocupado.

- **Caché de contenido:** Un proxy inverso puede almacenar en caché el contenido de los servidores. Cuando un cliente solicita un recurso, el proxy inverso verifica si ya lo tiene en caché y, en caso afirmativo, lo devuelve directamente al cliente sin tener que pasar por el servidor. Esto reduce la carga en los servidores y mejora el tiempo de respuesta para los clientes.

- **Escalabilidad y flexibilidad:** Al utilizar un proxy inverso, se puede agregar o eliminar servidores detrás de él sin que los clientes perciban los cambios. Esto facilita la escalabilidad de la infraestructura, ya que se pueden agregar más servidores según sea necesario sin afectar a los clientes. Además, el proxy inverso puede adaptarse a diferentes tipos de servidores y aplicaciones, brindando flexibilidad en la configuración de la red.

4.4.2. Justificación de la elección

Se evaluaron varias alternativas al seleccionar un proxy inverso de código abierto que satisfaga las necesidades de nuestro sistema: NGINX, Apache HTTP Server, HAProxy y Envoy.

Seguidamente, se expone un cuadro comparativo que analiza las ventajas y desventajas de cada una de las alternativas mencionadas:

Proxy inverso	Descripción	Ventajas	Desventajas
NGINX	Es un servidor web y proxy inverso de alto rendimiento. Reconocido por su eficiencia, escalabilidad y capacidad para manejar grandes volúmenes de solicitudes concurrentes.	— Alto rendimiento y eficiencia. — Configuración sencilla y legible. — Soporte para protocolos modernos como HTTP2. — Manejo eficiente de conexiones concurrentes. — Amplia variedad de módulos adicionales y extensibilidad.	— Limitaciones en configuraciones muy complejas. — Funcionalidades más orientadas a servir contenido estático.
Apache HTTP Server	Uno de los servidores web más populares. Puede actuar como proxy inverso utilizando el módulo *mod_proxy*. Apache es altamente configurable y ampliamente empleado en entornos en los que ya actúa como servidor web.	— Amplia comunidad y soporte. — Altamente flexible y configurable. — Capacidad para funcionar como proxy inverso y servidor web. — Amplia variedad de módulos adicionales y características disponibles. — Compatibilidad con muchos sistemas operativos.	— Mayor consumo de recursos en comparación con otros proxies inversos. — Rendimiento inferior en condiciones de alto tráfico y conexiones concurrentes.
HAProxy	Reconocido por su alto rendimiento y capacidad de manejar grandes volúmenes de datos. Es especialmente adecuado para aplicaciones que requieren de balanceo de carga y tolerancia a fallos.	— Enfocado en el balanceo de carga. — Buen rendimiento y escalabilidad. — Configuración sencilla y legible. — Capacidad para manejar grandes volúmenes de tráfico. — Tolerancia a fallos y alta disponibilidad.	— Orientado principalmente al balanceo de carga.
Envoy	Está diseñado específicamente para entornos de microservicios y contenedores.	— Amplias características para la configuración del enrutamiento y administración de tráfico. — Gran flexibilidad y extensibilidad.	— Diseñado para entornos de microservicios. — Mayor complejidad de configuración en comparación con otros proxies inversos.

Figura 4.6. Tabla comparativa entre distintas alternativas de proxy inverso disponibles.

Luego de un análisis exhaustivo de todas las alternativas disponibles, se tomó la decisión de utilizar NGINX como proxy inverso en nuestro sistema.

En primer lugar, se descartó Envoy como una alternativa viable. Aunque Envoy es muy empleado en entornos de microservicios y contenedores, nuestro sistema no está pensado para ser ejecutado en este tipo de entorno. Por ende, se optó por buscar una alternativa más viable para la configuración de infraestructura que se quería usar.

Posteriormente, se evaluó HAProxy como una posible solución. Sin embargo, debido a la nula experiencia con esta herramienta a comparación con NGINX y Apache HTTP Server, se decidió descartarla. Puesto que la configuración del proxy inverso es crítica para nuestro sistema, se prefirió utilizar una herramienta con la que ya tenemos experiencia previa.

Finalmente, se llegó a la conclusión de que NGINX es la opción más adecuada para los requerimientos del sistema. NGINX ha demostrado un alto rendimiento y eficiencia en entornos con mucho tráfico. Además, contamos con experiencia previa con la herramienta, lo que facilita la implementación, configuración y resolución de problemas.

Otra de las razones por la que se eligió NGINX es la facilidad de escribir configuraciones que sean sencillas y legibles. Como es uno de los proxies inversos más populares de código abierto, también hay disponible mucha documentación.

Si bien Apache también fue considerado, se descartó por su mayor consumo de recursos y rendimiento inferior en condiciones de alto tráfico y conexiones recurrentes en comparación con NGINX. La familiaridad y comodidad con NGINX también influyó en la decisión final.

4.4.3. Utilidad dentro del sistema

En nuestro sistema, el proxy inverso se ubica entre el Firewall y el resto de las aplicaciones o servicios que están corriendo dentro del servidor. La función principal de NGINX como proxy inverso es la de simplificar la configuración de red y el acceso de los usuarios a las prestaciones del sistema.

El proxy inverso actúa como intermediario entre los usuarios y los servicios que residen dentro del servidor. En lugar de acceder directamente a los servicios a través de sus puertos individuales, puede configurarse mediante NGINX un subdominio para conectarse a cada una de las aplicaciones.

Supongamos que tenemos un dominio llamado *ejemplo.com*. La configuración de NGINX permite, por ejemplo, redirigir las solicitudes entrantes desde *grafana.*dominio*.com* (*grafana* representa el subdominio, *dominio.com* el dominio) al puerto 3000 de nuestro servidor, que es el puerto de entrada por defecto de Grafana. Así, se facilita enormemente la tarea de acceder y compartir el contenido, ya que no hay que recordar direcciones complejas con números que no se relacionan de ninguna manera con el servicio al que se está accediendo: *dominio.com:3000* se transforma en *grafana.dominio.com*.

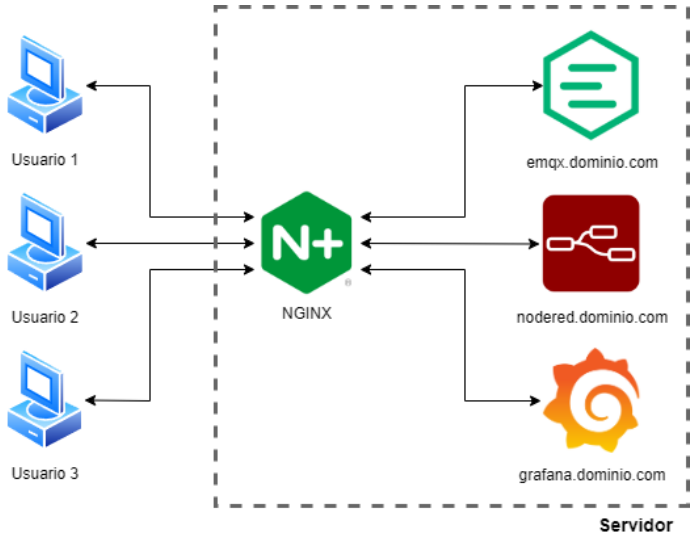

Figura 4.7. Utilidad del proxy inverso en el servidor, redireccionando cada subdominio a la aplicación correspondiente.

De este modo, se configuraron los siguientes subdominios en el sistema con ayuda de NGINX:

Solicitud entrante	Destino	Aplicación
emqx.dominio.com:443	127.0.0.1:3000	EMQX Dashboard
emqx.dominio.com:1883	127.0.0.1:1883	EMQX MQTT/TCP
emqx.dominio.com:8883	127.0.0.1:8883	EMQX MQTT/SSL
emqx.dominio.com:8083	127.0.0.1:8083	EMQX MQTT/WS
emqx.dominio.com:8084	127.0.0.1:8084	EMQX MQTT/WSS
grafana.dominio.com:443	127.0.0.1:3000	Grafana
nodered.dominio.com:443	127.0.0.1:1880	Node-RED

Figura 4.8. Configuración de los subdominios en NGINX.

En la columna de *destino*, el término "127.0.0.1" se utiliza como una dirección IP especial para referenciar el servidor local en el que se encuentra el servicio o aplicación que recibe la solicitud (en este caso, NGINX). También se lo conoce como *localhost*. Todas las redirecciones se realizan a la IP 127.0.0.1 debido a que todas las aplicaciones están corriendo dentro de este mismo servidor.

Por otra parte, también se ha configurado NGINX para que todas las peticiones realizadas al puerto estándar de HTTP, es decir, el puerto 80, sean redirigidas automáticamente al puerto 443. Esto garantiza que solo el tráfico cifrado y seguro a través de HTTPS sea permitido para comunicarse con las aplicaciones.

4.5. Bróker MQTT: EMQX

4.5.1. Definición del protocolo MQTT

El protocolo MQTT (por sus siglas en inglés, *Message Queuing Telemetry Transport*) es un protocolo de mensajería ligero y de bajo consumo de ancho de banda diseñado para la comunicación eficiente entre dispositivos que se encuentran en redes con recursos limitados, como aplicaciones de Internet de las Cosas (IoT) y comunicación máquina a máquina (M2M).

Este protocolo utiliza el modelo de publicación/suscripción, en el que los dispositivos o clientes MQTT pueden actuar de dos maneras distintas: como publicadores (*publishers*) o suscriptores (*suscribers*). Los publicadores emiten mensajes en tópicos (*topics*) específicos, mientras que los suscriptores se suscriben a los tópicos para recibir los mensajes publicados.

Un tópico MQTT es una cadena de texto que actúa como una etiqueta o identificador que agrupa mensajes relacionados. Los mensajes enviados por los publicadores se envían a un tópico determinado, y los suscriptores pueden expresar su interés en recibir mensajes de un tópico específico. Cada mensaje que se envía a un tópico es distribuido por el broker MQTT a todos los suscriptores que estén suscritos a ese tópico en particular.

La estructura de los tópicos es jerárquica y se organiza mediante niveles separados por barras ("/"). Por ejemplo, un tópico puede tener la forma "casa/comedor/luz" para representar el estado de una luz en el comedor dentro de una casa. Los suscriptores pueden elegir suscribirse a un tópico específico o a un patrón que coincida con varios tópicos utilizando comodines ("#"), lo que les permite recibir mensajes relevantes para sus necesidades.

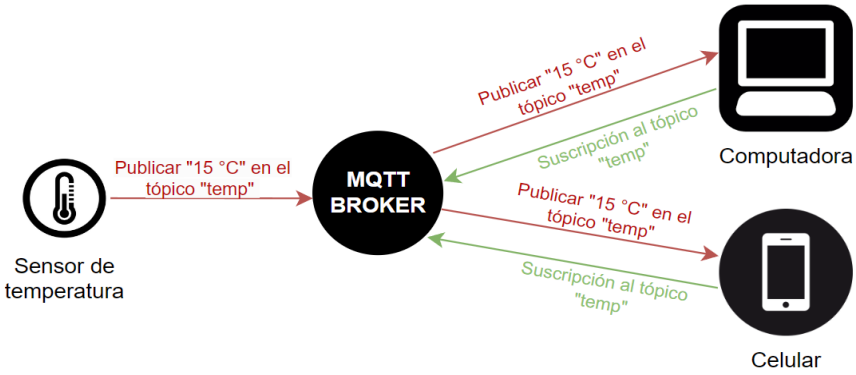

Figura 4.9. Ejemplo ilustrativo del funcionamiento del protocolo MQTT.

Dentro del protocolo, el bróker MQTT es un intermediario central que facilita la comunicación entre los dispositivos que emplean MQTT. Los publicadores envían sus mensajes al bróker, y este último se encarga de enrutarlos correctamente a los suscriptores interesados. El bróker es responsable de gestionar los tópicos y las suscripciones, asegurando que los mensajes sean entregados a los destinatarios adecuados.

Hay muchas características que hacen que este protocolo sea apropiado para sistemas como el nuestro, entre las que se encuentran:

- **Ligero.** Fue diseñado para minimizar la sobrecarga de datos y el consumo de ancho de banda. Los mensajes son pequeños y la sobrecarga del protocolo es mínima, lo que lo hace adecuado para aplicaciones que funcionan en redes con recursos limitados.
- **Eficiente.** Usa un modelo de comunicación asíncrona y está optimizado para minimizar el consumo de red y energía.
- **Escalable.** Puede adaptarse a diferentes aplicaciones y entornos de red, desde dispositivos individuales hasta sistemas distribuidos. Es capaz de manejar miles de conexiones simultáneas y ofrece flexibilidad en la configuración y gestión de tópicos.
- **Confiabilidad.** Ofrece distintos niveles de calidad de servicio (QoS) para garantizar la entrega de mensajes de acuerdo a las características y necesidades de la aplicación.
- **Soporte para desconexiones.** Está diseñado para funcionar en redes inestables, por lo que permite que los dispositivos se conecten y desconecten sin perder mensajes, ya que el bróker MQTT puede almacenar mensajes entrantes para su posterior entrega.

4.5.2. Justificación de la elección

Existe una variedad muy amplia de implementaciones de brókers MQTT de código abierto. Entre los más conocidos, se encuentran Mosquitto, EQMX, HiveMQ y VerneMQ, entre otros.

Luego de un análisis de cada una de las implementaciones, el bróker MQTT que se eligió para este sistema es EMQX. Para justificarlo, en primer lugar se evaluará el soporte a cada una de las versiones del protocolo MQTT en la siguiente tabla comparativa.

Nombre	MQTT 1.2	MQTT 3.1	MQTT 5.0	SSL/TLS	TCP	WS/WSS
Mosquitto	No	Sí	Sí	Sí	Sí	Sí
EMQX (Community Edition)	Sí	Sí	Sí	Sí	Sí	Sí
HiveMQ (Community Edition)	No	Sí (solo en el bróker)	Sí	Sí	Sí	Sí
VerneMQ	No	Sí	Sí	Sí	Sí	Sí

Figura 4.10. Comparativa entre alternativas de brókers MQTT de código abierto.

La tabla anterior muestra que EMQX es la implementación con el soporte a la mayor cantidad de versiones del protocolo. Cumple con el estándar MQTT 1.2, MQTT 3.1, MQTT 3.1.1 y MQTT 5.0, lo que garantiza la interoperabilidad con otros dispositivos y clientes MQTT. Soporta cuatro protocolos de transporte: TCP, TLS, WebSocket y QUIC, y permite la publicación de mensajes a través de HTTP, lo que lo convierte en el bróker que soporta la mayor cantidad de posibilidades para la publicación de mensajes.

Una de las ventajas significativas de EMQX es que está escrito en el lenguaje de programación Erlang. Erlang es un lenguaje de programación funcional y concurrente diseñado específicamente para construir sistemas altamente escalables, tolerantes a fallos y distribuidos. Ha sido utilizado durante décadas en aplicaciones de telecomunicaciones y sistemas críticos en los que la fiabilidad y la disponibilidad son fundamentales. Este lenguaje ha demostrado ser altamente robusto y resistente a fallas, lo convierte en un excelente lenguaje de programación para escribir un bróker MQTT para aplicaciones en tiempo real que requieren de alta disponibilidad.

En términos de seguridad, EMQX ofrece la posibilidad de configurar varios métodos para proteger las comunicaciones que se realizan a través del protocolo MQTT. Permite la autenticación y autorización basadas en usuarios y roles, cifrado de extremo a extremo mediante TLS/SSL y control de acceso a los tópicos basado en lista de control de acceso (ACL). EMQX soporta autenticación basada en usuario y contraseña, ClientID, JWT (JSON Web Tokens) y LDAP (Lightweight Directory Access Protocol), lo que brinda mayor flexibilidad y le permite adaptarse a diferentes escenarios de seguridad según los requisitos del sistema.

Además, EMQX es un bróker altamente flexible y personalizable. Admite extensiones y complementos para extender o añadir nuevas funcionalidades según los requisitos específicos del proyecto. Puede integrarse con todo tipo de bases de datos como MySQL, Redis, PostgreSQL, MongoDB, etc.

Por último, la elección de este bróker también se basó en su enorme comunidad activa de usuarios y desarrolladores, lo que proporciona una documentación sólida que facilita su configuración según los requisitos del sistema.

4.5.3. Utilidad dentro del sistema

EMQX es un bróker MQTT que actúa como intermediario en el sistema, interconectando dispositivos y aplicaciones que utilizan el protocolo MQTT para enviar y recibir mensajes.

Su función principal es la de recibir las señales de control y acciones generadas a partir del procesamiento de imágenes y luego reenviar estas señales a los microcontroladores que están esperando estos mensajes para efectuar acciones. La utilidad de EMQX radica en su capacidad para gestionar y administrar eficientemente la comunicación entre los dispositivos y aplicaciones de un sistema IoT.

4.6. Base de datos: MySQL

4.6.1. Definición de base de datos

Una base de datos es un conjunto estructurado de información que se emplea para almacenar, gestionar y recuperar grandes cantidades de información de manera eficiente. Usualmente, se utiliza para recopilar, almacenar, estructurar y recuperar datos de manera eficiente.

En su forma más básica, una base de datos consiste en una colección de datos interrelacionados que se organizan y almacenan en un sistema informático. Los datos que se almacenan pueden ser de distintos tipos, como texto, números y fechas. La estructura de una base de datos se basa en un conjunto de reglas y formatos predefinidos, llamado esquema, que define la forma en que se almacenan y se acceden a los datos.

La principal ventaja de las bases de datos es su capacidad para gestionar grandes volúmenes de información de manera eficiente. Para ello, se usan técnicas de indexación y estructuras de datos organizadas que permiten realizar operaciones como agregar, actualizar y eliminar datos de forma controlada y segura, garantizando la integridad y consistencia de los datos.

Existen dos tipos principales de bases de datos: las bases de datos relacionales y las bases de datos no relacionales (también llamadas bases de datos NoSQL). Estas difieren en la forma en que se estructuran y relacionan los datos.

Las bases de datos relacionales están compuestas por tablas para la representación y organización de los datos. Cada tabla consta de filas y columnas, donde las filas representan registros individuales y las columnas simbolizan los atributos o campos de estos registros. Este tipo de base de datos utiliza claves primarias y claves foráneas para establecer relaciones entre las tablas y asegurar la integralidad de datos.

Por otra parte, las bases de datos no relacionales se emplean para manejar grandes volúmenes de datos no estructurados o semiestructurados. En lugar de utilizar tablas, usan diferentes modelos de datos, como grafos, documentos, columnas o pares clave-valor para almacenar y organizar la información. Se caracterizan por su flexibilidad y escalabilidad, lo que las hace ideales para el análisis de datos masivos y sistemas distribuidos. Además, son altamente tolerantes a fallos y pueden manejar grandes volúmenes de trabajo.

Para este sistema, se evaluarán tanto bases de datos relacionales como no relacionales.

4.6.2. Justificación de la elección

En el proceso de elección de la base de datos para nuestro sistema, decidimos enfocarnos únicamente en aquellas herramientas con la que ya tenemos cierta familiaridad. Esta elección se basa en el objetivo de agilizar y optimizar el desarrollo del sistema, aprovechando el conocimiento y la experiencia que se obtuvo en el transcurso de la carrera.

Se compararon MySQL, MongoDB y Cassandra como se muestra a continuación:

Característica	MySQL	MongoDB	Cassandra
Tipo de base de datos	Relacional	No relacional (orientado a documentos)	No relacional (orientado a columnas)
Lenguaje de consulta	SQL (Structured Query Language)	Query Language (similar a JSON)	CQL (Cassandra Query Language)
Escalabilidad	Vertical (a través de mejoras de hardware)	Horizontal (clustering)	Horizontal (clustering)
Esquema	Fijo y definido	Flexible	Flexible
Transacciones	Soporte completo para transacciones ACID	Soporte limitado para transacciones	No soporta transacciones ACID
Flexibilidad	Estructura de datos poco flexible	Estructura flexible	Estructura flexible
Rendimiento	Rápido en consultas estructuradas	Rápido en lecturas	Rápido en escrituras y lecturas en grandes volúmenes de datos
Indexación	Soporta índices tradicionales	Soporte para índices	Soporte para índices
Uso principal	Aplicaciones empresariales	Aplicaciones web y móviles	Grandes volúmenes de datos, tiempo real y aplicaciones distribuidas

Figura 4.11. Comparativa entre sistemas de gestión de bases de datos.

Luego de efectuar el análisis y la comparación entre las alternativas, decidimos utilizar MySQL como base de datos para el sistema debido a la familiaridad con la herramienta y el lenguaje de consulta SQL que utiliza.

Ya hemos trabajado con MySQL en proyectos anteriores y estamos familiarizados con su configuración, administración y optimización. Esto permite que comencemos rápidamente con el desarrollo sin tener que invertir tiempo en aprender una nueva herramienta.

Por otra parte, MySQL utiliza el lenguaje de consulta estructurado SQL, que es ampliamente conocido y usado en la industria.

Por último, MySQL es altamente compatible con una variedad de herramientas, bibliotecas y frameworks de desarrollo. Esto brinda una mayor flexibilidad a la hora de integrar nuestra base

de datos en otros servicios como Grafana y Node-RED, que ya disponen de extensiones que permiten interactuar con bases de datos MySQL.

4.6.3. Utilidad dentro del sistema

MySQL se emplea como base de datos para almacenar el momento en que se realiza la detección de los gestos y los comandos enviados a los actuadores. Esto permitirá tener un historial de las acciones detectadas.

Además, otros servicios como Grafana pueden acceder a los datos almacenados a la base de datos y presentarle al usuario gráficas que representen de forma visual la detección de gestos a lo largo del tiempo.

4.7. Herramienta de programación visual: Node-RED

4.7.1. Definición

Una herramienta de programación visual es un entorno de desarrollo o una aplicación que permite a los programadores crear programas o lógica de manera gráfica, en lugar de usar la tradicional codificación de texto.

En contraste con el empleo de lenguajes de programación tradicionales, como Python, Java o C#, las herramientas de programación visual otorgan una interfaz visual en la que los usuarios pueden arrastrar y soltar elementos predefinidos, conocidos como bloques, nodos o componentes, dependiendo de la herramienta en uso. Cada elemento visual representa una función o tarea específica, como la entrada de datos, procesamiento, salida, bucles, tomas de decisiones, etc. La lógica del programa se origina al conectar estos elementos entre sí.

Este tipo de herramienta es sumamente útil para producir prototipos rápidos o automatizar tareas sin la necesidad de desarrollar aplicaciones utilizando los lenguajes de programación tradicionales.

4.7.2. Justificación de la elección

Existen una multitud de herramientas de programación visual populares como Scratch, Blockly, LabVIEW y Node-RED. En el caso particular de este subsistema, buscamos una

solución que permita suscribirse a tópicos MQTT y conectarse a una base de datos MySQL para almacenar los mensajes enviados a través de dicho protocolo.

En este contexto, hemos identificado que Node-RED es una opción que satisface ampliamente todos los requisitos necesarios en una herramienta de esta naturaleza. Además, contamos con experiencia previa y familiaridad en su uso, lo que agrega un factor positivo a favor de su elección.

No obstante, como parte de un proceso de evaluación exhaustiva, se decidió realizar una comparativa entre Node-RED, Blockly y LabVIEW. El objetivo es analizar en profundidad las características, ventajas y desventajas de cada una de estas herramientas para tomar una decisión fundamentada que mejor se adapte a nuestras necesidades.

Característica	Node-RED	Blockly	LabVIEW
Comunidad y soporte	Amplia comunidad y soporte.	Comunidad en crecimiento.	Comunidad y soporte bien establecidos.
Flexibilidad	Amplia variedad de nodos y alternativas. Inmensa cantidad de extensiones para adaptarlo a todo tipo de requerimientos.	Limitado en cuanto a la cantidad de bloques y funciones predefinidas. Catálogo de extensiones.	Amplia gama de módulos y herramientas
IoT	Amplia compatibilidad con dispositivos y protocolos IoT.	Requiere de extensiones.	Amplia compatibilidad con dispositivos y protocolo IoT.
Integración con bases de datos	Amplia compatibilidad con varias bases de datos.	Requiere de extensiones.	Amplia compatibilidad con varias bases de datos.
Curva de aprendizaje	Relativamente baja.	Baja.	Media o alta.
Escalabilidad	Escalable para proyectos IoT de pequeña a mediana escala.	Limitado a proyectos simples.	Escalable para proyectos IoT de pequeña a gran escala.
Licencia	Código abierto y gratuito.	Código abierto y gratuito.	Propietario, cuenta con una versión *community*.

Figura 4.12. Comparativa entre herramientas de programación visual.

Después de analizar las opciones previas, se descartó en primer lugar la herramienta LabVIEW debido a su licencia propietaria. Aunque LabVIEW es una herramienta altamente empleada en la industria y ofrece la posibilidad de desarrollar soluciones potentes, el objetivo de este sistema era utilizar únicamente herramientas de código abierto.

Luego, se decidió no utilizar Blockly debido a que está principalmente diseñado para proyectos simples y educativos. Si bien cuenta con una interfaz amigable y una curva de aprendizaje baja, su compatibilidad con dispositivos y protocolos IoT es limitada y requiere de extensiones y personalización adicional para adaptarse a las necesidades de nuestro proyecto.

Finalmente, se optó por usar Node-RED como la herramienta utilizada en nuestro sistema. Node-RED es una plataforma de programación visual de código abierto basada en nodos que permite a los programadores conectar dispositivos, APIs y servicios en línea de manera sencilla. Es ampliamente usada en el ámbito de Internet de las Cosas (IoT) y la integración de sistemas, ya que facilita la creación de aplicaciones y la conexión intuitiva de dispositivos y servicios.

Una de las razones por las que se eligió esta herramienta es porque ofrece una curva de aprendizaje relativamente baja y la experiencia previa que ya teníamos con la herramienta facilita el desarrollo requerido para este componente en el sistema. Además, Node-RED es altamente escalable, por lo que se puede adaptar a proyectos IoT de pequeña a mediana escala sin problemas.

Node-RED también cuenta con una comunidad de código abierto muy activa, lo que brinda acceso a una amplia variedad de complementos desarrollados por la comunidad. Esto permite la integración con las bases de datos más populares y la conexión a nuestro bróker MQTT, que es crucial en cuanto a la necesidad con esta herramienta.

Además, Node-RED permite definir y crear nodos personalizados que efectúen tareas específicas usando el lenguaje de programación JavaScript, por lo que es posible adaptarlo para cumplir con nuestros requerimientos.

4.7.3. Utilidad dentro del sistema

En este caso específico, se tuvo la opción de desarrollar la aplicación utilizando un lenguaje de programación tradicional; sin embargo, se optó por emplear una herramienta de programación visual. La elección de esta herramienta se basó en la necesidad de agilizar la etapa de desarrollo, permitiendo producir la lógica de manera más rápida y eficiente a través de su interfaz gráfica.

El uso de la programación visual proporcionó una ventaja significativa en términos de rapidez de implementación y reducción de la complejidad asociada con enfoques más tradicionales.

Esta decisión estratégica en el desarrollo permitió alcanzar los objetivos del proyecto de manera más efectiva y eficiente.

Node-RED desempeña un papel relevante al capturar los mensajes que se envían a través del protocolo MQTT cuando se produce la detección de un gesto. Cómo mencionamos previamente, Node-RED cuenta con una gran cantidad de complementos que facilitan la conexión con nuestro bróker MQTT y el posterior almacenamiento de los datos capturados en la base de datos.

La capacidad de añadir nodos personalizados y ejecutar funciones programadas en JavaScript es especialmente útil en nuestro caso, ya que permite adaptar los datos capturados al formato requerido antes de guardarlos en la base de datos.

Gracias a la versatilidad de Node-RED, podemos implementar de manera eficiente la captura de mensajes MQTT y asegurar que los datos están correctamente estructurados y almacenados en nuestra base de datos. Esto nos otorga la capacidad de en un futuro realizar análisis y obtener información valiosa a partir de los gestos detectados.

4.8. Herramienta de visualización de datos: Grafana

4.8.1. Definición de herramienta de visualización de datos

Una herramienta de visualización de datos es una aplicación diseñada para ayudar a representar visualmente conjuntos de datos de manera comprensible y significativa. Este tipo de herramientas permiten convertir datos en gráficos, tablas, diagramas u otras formas visuales que facilitan la comprensión y el análisis de la información.

Son utilizadas en diversos campos, como análisis de negocios, ciencias de datos y marketing, debido a que ayudan a explorar los datos, identificar patrones, tendencias, relaciones y comunicar hallazgos o información de manera efectiva.

Generalmente, las herramientas de visualización de datos permiten importar datos de diversas fuentes, seleccionar diferentes tipos de gráficos y visualizaciones, filtrar y resumir los datos, e interactuar con las visualizaciones para obtener detalles adicionales o realizar análisis más profundos.

4.8.2. Justificación de la elección

Se exploraron varias opciones al seleccionar una herramienta de visualización de datos. Las principales alternativas que se consideraron fueron Node-RED Dashboard y Grafana, por ser de código abierto y compatibles con MySQL y el protocolo MQTT.

En primer lugar, Grafana es una herramienta de código abierto diseñada específicamente para la visualización de datos y la creación de paneles. Ofrece una amplia variedad de paneles de visualización, lo que la convierte en una elección muy popular para el monitoreo y análisis de datos.

Grafana admite una amplia variedad de orígenes de datos, como bases de datos, servicios en la nube y sistemas de monitorización. Además, cuenta con una amplia cantidad de opciones de visualización, desde gráficos de líneas o barras hasta mapas y paneles de control.

Por otra parte, Node-RED Dashboard es una extensión de Node-RED, un entorno de programación visual utilizado para construir aplicaciones IoT y de automatización. Node-RED Dashboard se integra fácilmente en flujos de Node-RED, lo que permite una comunicación fluida con las fuentes de datos y otros dispositivos.

Node-RED Dashboard ofrece una amplia variedad de widgets predefinidos para mostrar y controlar datos en tiempo real. Permite crear fácilmente paneles que muestran valores, envíen comandos a dispositivos, realicen cálculos y ejecuten acciones basadas en eventos. No obstante, sus capacidades de visualización son limitadas en comparación con Grafana.

Por este motivo, se seleccionó Grafana para la visualización de los datos. Si bien Node-RED proporciona una solución más sencilla para crear paneles básicos de visualización y control, Grafana se centra en la visualización de datos y permite crear dashboards más flexibles, lo que la convierte en una opción más versátil y potente para el análisis y monitoreo de datos.

4.8.3. Utilidad dentro del sistema

Grafana es utilizada en este contexto para mostrar las imágenes procesadas en tiempo real, la detección de gestos y las acciones que se envían a los actuadores.

Primeramente, Grafana permite mostrar las imágenes procesadas en tiempo real. A medida que las imágenes son capturadas y procesadas por el sistema, Grafana proporciona una interfaz

intuitiva para visualizar estas imágenes en tiempo real. Esto permite que los usuarios monitoreen y controlen los resultados de manera instantánea.

Por otra parte, Grafana también se usa para mostrar el resultado de la detección de los gestos. A medida que el sistema procesa las imágenes y realiza el reconocimiento de gestos, se ofrecen en Grafana paneles que ofrecen una representación visual de los gestos detectados. De esta manera, los usuarios pueden saber cuándo se detectó cada uno de los posibles gestos.

Por último, Grafana permite ver las acciones que se envían a los actuadores. Una vez que los gestos son detectados y se determinan las acciones correspondientes, Grafana muestra las acciones enviadas a través de widgets. Así, se puede hacer una supervisión efectiva del sistema.

4.9. Implementación

4.9.1. Consideraciones generales

La implementación del servidor se llevó a cabo utilizando un servidor proporcionado por la universidad con las siguientes especificaciones:

- **CPU:** 1 core de Intel(R) Xeon(R) CPU E5-2699C v4 @ 2.20 GHz
- **Memoria RAM:** 1.5 GB
- **HDD:** 11 GB.

En este servidor, se han puesto en funcionamiento las aplicaciones descritas anteriormente, todas ellas accesibles a través de subdominios configurados con la ayuda de NGINX. Para garantizar la seguridad y privacidad de los datos transmitidos, se han configurado certificados SSL/TLS mediante el uso de certbot y Let's Encrypt.

Certbot es una herramienta de línea de comandos usada para obtener e instalar automáticamente certificados SSL/TLS de Let's Encrypt en el servidor. Let's Encrypt es una Autoridad de Certificación de dominio público que proporciona certificados SSL/TLS gratuitos para asegurar que las conexiones entre las aplicaciones que se encuentran fuera del servidor y el mismo servidor estén protegidas contra posibles ataques de intermediarios maliciosos.

Esta combinación de herramientas asegura que los usuarios puedan acceder a los servicios de manera segura y privada, protegiendo la integridad de los datos transmitidos ante posibles ataques.

Por otra parte, para la implementación, también se empleó un dominio proporcionado por la universidad y para fines de referencia, se le ha denominado "dominio.com" (aunque es diferente al dominio real) a lo largo del informe.

A través de un panel de control asociado al servicio adquirido, se han configurado los siguientes registros DNS:

Dominio	Dirección IP
dominio.com	Dirección IP pública del servidor

Figura 4.13. Registros DNS de tipo A configurados para el dominio.

Dominio	Destino
emqx.dominio.com	dominio.com
nodered.dominio.com	dominio.com
grafana.dominio.com	dominio.com

Figura 4.14. Registros DNS de tipo CNAME configurados para el dominio.

Los registros de tipo A asocian el dominio principal (dominio.com) con la dirección IP del servidor, permitiendo que los usuarios accedan a las aplicaciones mediante el nombre de dominio. Los registros de tipo CNAME, por otro lado, definen los subdominios (emqx.dominio.com, nodered.dominio.com y grafana.dominio.com) que apuntan al dominio principal (dominio.com). De esta forma, los usuarios pueden acceder a cada aplicación a través de sus respectivos subdominios.

Por otra parte, para ejecutar los comandos que se presentarán en las siguientes secciones, se estableció una conexión al servidor mediante SSH utilizando las credenciales proporcionadas por la universidad. De esta manera, se logró acceder al servidor de manera remota y realizar las configuraciones necesarias.

4.9.2. FirewallD

En primer lugar, se habilitaron todos los puertos necesarios para todas las aplicaciones que van a estar corriendo en este servidor. A continuación, se explica paso a paso el procedimiento llevado a cabo.

Antes de hacer cualquier cambio, es importante verificar si FirewallD está habilitado y en funcionamiento en el servidor. Esto se realizó con el comando:

```
sudo systemctl status firewalld
```

Figura 4.15. Comando para comprobar el estado de FirewallD.

En el caso de que FirewallD no esté activo, lo inicializamos con el comando:

```
sudo systemctl start firewalld
```

Figura 4.16. Comando para iniciar FirewallD.

Sin embargo, en nuestro caso, no fue necesario activarlo, ya que ya se encontraba en funcionamiento.

```
[server@vxsct2810 ~]$ sudo systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-08-02 13:08:52 -03; 22min ago
     Docs: man:firewalld(1)
 Main PID: 751 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─751 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid

Aug 02 13:08:44 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon ...
Aug 02 13:08:52 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
Aug 02 13:08:53 localhost.localdomain firewalld[751]: WARNING: AllowZoneDrifting is enabled. Thi ... w.
Hint: Some lines were ellipsized, use -l to show in full.
```

Figura 4.17. Estado del servicio de FirewallD en el servidor.

Asimismo, para asegurarse de que FirewallD se inicie automáticamente junto con el sistema, ejecutamos el comando:

```
sudo systemctl enable firewalld
```

Figura 4.18. Comando para que FirewallD inicie automáticamente con el sistema.

Con FirewallD activo, procedimos a habilitar los puertos necesarios para las aplicaciones que se ejecutarán en el servidor. De acuerdo a lo visto en la sección correspondiente a FirewallD, estos son:

- Puerto 22 TCP (SSH)
- Puerto 80 TCP (HTTP)
- Puerto 443 TCP (HTTPS)
- Puerto 1883 TCP (MQTT)
- Puerto 8883 TCP (MQTT/SSL)
- Puerto 8083 TCP (MQTT/WS)
- Puerto 8084 TCP (MQTT/WSS)

Para habilitar cada uno de estos puertos, se utilizaron los siguientes comandos:

```
sudo firewall-cmd --permanent --add-port=22/tcp
sudo firewall-cmd --permanent --add-port=80/tcp
sudo firewall-cmd --permanent --add-port=443/tcp
sudo firewall-cmd --permanent --add-port=1883/tcp
sudo firewall-cmd --permanent --add-port=8883/tcp
sudo firewall-cmd --permanent --add-port=8083/tcp
sudo firewall-cmd --permanent --add-port=8084/tcp
```

Figura 4.19. Lista de comandos para habilitar todos los puertos necesarios en el servidor.

El parámetro "--permanent" asegura que las reglas se mantendrán incluso después de reiniciar el servidor.

Una vez que se han añadido las reglas para los puertos específicos, es necesario recargar FirewallD para que los cambios surtan efecto. Esto lo realizamos mediante el siguiente comando:

```
sudo firewall-cmd --reload
```

Figura 4.20. Comando para recargar las reglas del firewall en FirewallD.

Para confirmar que los puertos se han habilitado correctamente, utilizamos el siguiente comando para obtener una lista de los puertos permitidos en FirewallD:

```
sudo firewall-cmd --list-ports
```

Figura 4.21. Comando para recargar las reglas del firewall en FirewallD.

Este comando muestra la lista de los puertos permitidos, incluyendo los puertos que han sido habilitados previamente.

```
[server@vxsct2810 ~]$ sudo firewall-cmd --list-ports
22/tcp 443/tcp 1883/tcp 8883/tcp 8083/tcp 8084/tcp 80/tcp
```

Figura 4.22. Lista de puertos habilitados en el servidor.

Como todos los puertos se muestran correctamente habilitados, se ha realizado con éxito la configuración de FirewallD en el servidor. Esto permite el funcionamiento adecuado de las aplicaciones que requieren acceso a los puertos especificados.

4.9.3. NGINX

Luego, se procedió con la instalación del proxy inverso, NGINX, en el servidor proporcionado por la universidad.

Recordando la sección de NGINX, los subdominios que se debían configurar para las aplicaciones que se querían acceder a través de subdominios son:

- eqmx.dominio.com
- nodered.dominio.com
- grafana.dominio.com

Antes de instalar NGINX, tenemos que instalar el repositorio EPEL en el servidor. Esto lo hacemos con el siguiente comando:

```
sudo yum install epel-release
```

Figura 4.23. Comando para instalar el repositorio EPEL en el servidor.

Luego, es importante actualizar los repositorios y los paquetes instalados en el sistema operativo. Realizamos esta tarea utilizando la herramienta de administración de paquetes *yum*, incluida en CentOS:

```
sudo yum update
```

Figura 4.24. Comando para actualizar los paquetes en el servidor.

Una vez que hemos actualizado el sistema, procedimos a instalar NGINX mediante el siguiente comando:

```
sudo yum install nginx
```

Figura 4.25. Comando para instalar NGINX en el servidor.

Tras completar la instalación, iniciamos el servicio de NGINX y lo configuramos para que se inicie automáticamente al arrancar el sistema:

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

Figura 4.26. Comandos para iniciar y configurar el inicio automático de NGINX.

Luego, verificamos el estado del servicio de NGINX con el comando:

```
sudo systemctl status nginx
```

Figura 4.27. Comandos para iniciar y configurar el inicio automático de NGINX.

Como se puede observar, el servicio de NGINX se encuentra activo:

```
[server@vxsct2810 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 13:51:49 -03; 11s ago
 Main PID: 68459 (nginx)
   CGroup: /system.slice/nginx.service
           ├─68459 nginx: master process /usr/sbin/nginx
           ├─68460 nginx: worker process
           └─68461 nginx: worker process

Aug 02 13:51:49 vxsct2810 systemd[1]: Starting The nginx HTTP and reverse proxy server ...
Aug 02 13:51:49 vxsct2810 nginx[68454]: nginx: the configuration file /etc/nginx/nginx.conf sy ... s ok
Aug 02 13:51:49 vxsct2810 nginx[68454]: nginx: configuration file /etc/nginx/nginx.conf test i ... sful
Aug 02 13:51:49 vxsct2810 systemd[1]: Started The nginx HTTP and reverse proxy server.
Hint: Some lines were ellipsized, use -l to show in full.
```

Figura 4.28. Estado del servicio de NGINX en el servidor.

El siguiente paso fue configurar NGINX para manejar el tráfico de los subdominios y redirigirlo a las aplicaciones correspondientes. En CentOS 7, los archivos de configuración de NGINX se encuentran en el directorio "/etc/nginx/conf.d/". Utilizamos un editor de texto como nano o vi para añadir el contenido necesario a los archivos de configuración. Por ejemplo, para el subdominio "grafana.dominio.com", ejecutamos:

```
sudo nano /etc/nginx/conf.d/grafana.conf
```

Figura 4.29. Comando para editar con nano el archivo de configuración de NGINX para el subdominio de Grafana.

En este contexto, resultó fundamental proporcionar el contenido adecuado para cada subdominio en sus respectivos archivos de configuración. A continuación, se muestra el archivo de configuración que se utilizó para Grafana:

```
server {
    server_name grafana.dominio.com;

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_cache_bypass  $http_upgrade;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Port $server_port;
    }

    listen [::]:443 ssl;
    listen 443 ssl;
    ssl_certificate
/etc/letsencrypt/live/dominio.com/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/dominio.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    if ($host = grafana.dominio.com) {
        return 301 https://$host$request_uri;
    }

    listen 80;
    listen [::]:80;
    server_name grafana.dominio.com;
    return 404;
}
```

Figura 4.30. Contenido del archivo de configuración de NGINX para el subdominio de Grafana.

Este archivo de configuración se utiliza para habilitar la redirección y el acceso seguro (HTTPS) a la aplicación web de Grafana que está escuchando en el puerto 3000.

El primer bloque "server" está configurado para manejar las solicitudes HTTPS dirigidas a "grafana.dominio.com". Las principales directivas utilizadas son:

- **server_name:** Especifica el nombre del servidor para el cual esta configuración es válida, en este caso, "grafana.dominio.com".
- **location /:** Esta directiva define cómo NGINX debe manejar las solicitudes que llegan al servidor. La configuración **proxy_pass** indica que NGINX debe reenviar todas las solicitudes recibidas a la dirección "http://127.0.0.1:3000", que es donde Grafana está ejecutándose.
- **ssl_certificate, ssl_certificate_key, ssl_dhparam:** Estas directivas están configuradas para habilitar la seguridad mediante el uso de certificados SSL/TLS, lo que permite conexiones HTTPS seguras a Grafana. Se especifican la ubicación de los certificados SSL y los archivos de configuración necesarios para el manejo seguro de conexiones.

El segundo bloque "server" es una redirección HTTP para cualquier solicitud dirigida a "grafana.dominio.com". Se configura para redirigir todas las solicitudes HTTP a la versión HTTPS de "grafana.dominio.com" usando un redireccionamiento 301. Esto garantiza que todas las solicitudes se redirijan a través de HTTPS.

Este mismo enfoque se utilizó para los archivos de configuración de EMQX y Node-RED, asegurando así el acceso seguro y la redirección adecuada para estas aplicaciones también.

```
[server@vxsct2810 conf.d]$ ll
total 12
-rw-r--r--. 1 root root 1632 Aug  2 13:56 emqx.conf
-rw-r--r--. 1 root root 1625 Aug  2 13:55 grafana.conf
-rw-r--r--. 1 root root 1626 Aug  2 13:57 nodered.conf
```

Figura 4.31. Lista de archivos de configuración de NGINX para los subdominios disponibles.

Luego de añadir el contenido a los archivos de configuración, verificamos si hay errores en la configuración antes de aplicarla. Podemos realizar esta verificación con el siguiente comando:

```
sudo nginx -t
```

Figura 4.32. Comando para verificar si existen errores en los archivos de configuración de NGINX.

```
[server@vxsct2810 conf.d]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Figura 4.33. Mensaje que expresa que no hay errores en los archivos de configuración de NGINX creados.

Luego de verificar que la configuración no tiene errores, aplicamos los cambios reiniciando el servicio de NGINX:

```
sudo systemctl restart nginx
```

Figura 4.34. Comando para reiniciar el servicio de NGINX.

Con esto, todos los subdominios necesarios para las aplicaciones ya quedaron correctamente configurados.

4.9.4. MySQL

A continuación, se describe el proceso de instalación y configuración de MySQL en el servidor. A lo largo del proceso, se detallan las acciones realizadas y las acciones tomadas durante el proceso.

Para garantizar que se instale la versión más reciente de MySQL disponible en los repositorios de CentOS, ejecutamos el siguiente comando para actualizar el índice de paquetes del sistema:

```
sudo yum update
```

Figura 4.35. Comando para actualizar los paquetes en el servidor.

Con el sistema actualizado, procedimos a instalar el servidor MySQL mediante el siguiente comando:

```
sudo yum install mariadb-server
```

Figura 4.36. Comando para instalar MariaDB.

En este caso, optamos por instalar MariaDB, que es un sistema gestor de bases de datos de código abierto. MariaDB es una bifurcación (fork) de MySQL y es administrado por los desarrolladores originales de MySQL. Su diseño se plantea como un reemplazo directo de MySQL y, por consiguiente, utiliza algunos comandos que hacen referencia a “mysql”, como se detalla más adelante. La elección de MariaDB se basa en el hecho de que, en la distribución CentOS 7, se establece como el paquete predeterminado para la gestión de bases de datos en lugar de MySQL.

Una vez finalizada la instalación, iniciamos el servicio de MySQL con el comando:

```
sudo systemctl start mariadb
```

Figura 4.37. Comando para iniciar el servicio de MariaDB.

Para asegurarse de que MySQL se inicie automáticamente cada vez que se inicie el servidor, utilizamos el comando:

```
sudo systemctl enable mariadb
```

Figura 4.38. Comando para configurar el inicio automático de MariaDB.

```
[server@vxsct2810 ~]$ sudo systemctl status mariadb
● mariadb.service - MariaDB database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 14:12:15 -03; 18s ago
 Main PID: 79523 (mysqld_safe)
   CGroup: /system.slice/mariadb.service
           ├─79523 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
           └─79688 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/l ...

Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: MySQL manual for more instructions.
Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: Please report any problems at http://ma ... ra
Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: The latest information about MariaDB is ... /.
Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: You can find additional information abo ... t:
Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: http://dev.mysql.com
Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: Consider joining MariaDB's strong and v ... y:
Aug 02 14:12:13 vxsct2810 mariadb-prepare-db-dir[79440]: https://mariadb.org/get-involved/
Aug 02 14:12:13 vxsct2810 mysqld_safe[79523]: 230802 14:12:13 mysqld_safe Logging to '/var/log/ ... g'.
Aug 02 14:12:13 vxsct2810 mysqld_safe[79523]: 230802 14:12:13 mysqld_safe Starting mysqld daemo ... sql
Aug 02 14:12:15 vxsct2810 systemd[1]: Started MariaDB database server.
Hint: Some lines were ellipsized, use -l to show in full.
```

Figura 4.39. Comando para actualizar los paquetes en el servidor.

Luego, ejecutamos el script para configurar la seguridad de MySQL:

```
sudo mysql_secure_installation
```

Figura 4.40. Comando para ejecutar el script para segurizar la instalación de MySQL.

En este punto, se configura la seguridad de la instalación de MySQL, incluyendo la contraseña del usuario “root”, la eliminación de usuarios anónimos y la desactivación del inicio de sesión remoto del usuario “root”.

En el caso de la configuración de MySQL para este servidor, se proporcionó una contraseña segura para el usuario “root” de MySQL. Esto evita el acceso no autorizado a la base de datos y protege la privacidad e integridad de los datos almacenados.

Con el objetivo de mejorar la seguridad, se desactiva el inicio de sesión remoto para el usuario “root”. Esto significa que el usuario “root” solo puede acceder a MySQL desde el propio servidor, lo que reduce las posibles vulnerabilidades asociadas con el acceso remoto.

También se procedió a eliminar los usuarios anónimos de MySQL. Los usuarios anónimos son aquellos que pueden acceder al servidor sin proporcionar credenciales. Al eliminarlos, se minimiza el riesgo de posibles brechas de seguridad.

```
Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database ...
 ... Success!
 - Removing privileges on test database ...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up ...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Figura 4.41. Algunos pasos para segurizar la instalación de MySQL.

Con la finalización de estos pasos, se completó la instalación y configuración de MySQL en el servidor, garantizando una mayor protección de los datos almacenados y reduciendo las posibles vulnerabilidades asociadas a la base de datos. Luego, continuamos con la configuración de los usuarios y las tablas requeridas para cada una de las aplicaciones.

En primer lugar, se inició sesión con el usuario "root":

```
sudo mysql -u root -p
```

Figura 4.42. Comando para iniciar sesión en MySQL con el usuario "root".

La primera aplicación para la que configuramos las credenciales y las tablas fue EMQX. Para que EMQX pueda interactuar con la base de datos, fue necesario crear un usuario dedicado. Este usuario es utilizado para autenticar y autorizar a los usuarios en el bróker MQTT. El siguiente comando SQL se utilizó para crear el usuario "emqx":

```
CREATE USER 'emqx'@'localhost' IDENTIFIED BY 'contraseña';
```

Figura 4.43. Comando para crear el usuario 'emqx'@'localhost' en MySQL.

Donde "emqx" es el nombre del usuario que se está creando y "localhost" indica que este usuario solo puede conectarse desde el mismo equipo en el que se encuentra la base de datos.

Sin embargo, el usuario "emqx" necesita ciertos permisos para realizar operaciones en la base de datos del bróker MQTT. En este caso, se le concedió el permiso de SELECT en la base de datos "emqx" para que pueda ejecutar consultas sobre ella. El comando SQL para otorgar los permisos es el siguiente:

```
GRANT SELECT ON emqx.* TO 'emqx'@'localhost';
FLUSH PRIVILEGES;
```

Figura 4.44. Comandos para otorgar los permisos de selección a 'emqx'@'localhost' en MySQL.

Posteriormente, creamos la base de datos "emqx", donde almacenaremos las tablas que contienen la información de los usuarios y sus permisos.

```
CREATE DATABASE emqx;
```

Figura 4.45. Comando para la base de datos 'emqx' en MySQL.

Una vez creada la base de datos, añadimos la tabla "mqtt_user", que almacena la información de los usuarios que tienen acceso al bróker MQTT. A continuación, se presenta el comando de lenguaje de definición de datos (DDL) para crear dicha tabla con la estructura necesaria:

```
CREATE TABLE `mqtt_user` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) DEFAULT NULL,
  `password_hash` varchar(100) DEFAULT NULL,
  `salt` varchar(35) DEFAULT NULL,
  `is_superuser` tinyint(1) DEFAULT 0,
  `created` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `mqtt_username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figura 4.46. Comando para crear la tabla 'mqtt_user' en MySQL.

La tabla "mqtt_user" contiene los siguientes campos:

- **id:** Un identificador único para cada usuario (autoincremental).
- **username:** El nombre de usuario del cliente MQTT.
- **password_hash**: El hash de la contraseña del usuario para mantener la seguridad de las credenciales.
- **salt:** Un valor aleatorio utilizado para reforzar la seguridad de la contraseña almacenada.
- **is_superuser:** Un campo booleano que indica si el usuario tiene privilegios de superusuario.
- **created:** La fecha y hora de creación del registro.

```
MariaDB [emqx]> CREATE TABLE `mqtt_user` (
    ->   `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
    ->   `username` varchar(100) DEFAULT NULL,
    ->   `password_hash` varchar(100) DEFAULT NULL,
    ->   `salt` varchar(35) DEFAULT NULL,
    ->   `is_superuser` tinyint(1) DEFAULT 0,
    ->   `created` datetime DEFAULT NULL,
    ->   PRIMARY KEY (`id`),
    ->   UNIQUE KEY `mqtt_username` (`username`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
Query OK, 0 rows affected (0.00 sec)
```

Figura 4.47. Ejecución del comando para crear la tabla 'mqtt_user' en MySQL en el servidor.

Entonces, para crear un nuevo usuario para, por ejemplo, los microcontroladores o la aplicación de procesamiento de imágenes, podemos utilizar el siguiente comando SQL:

```
INSERT INTO mqtt_user(username, password_hash, salt, is_superuser)
VALUES ('emqx_u', SHA2(concat('public', 'foo123'), 256), 'foo123',
1);
```

Figura 4.48. Ejemplo de comando para insertar un registro a la tabla 'mqtt_user' en MySQL.

En este ejemplo, se está creando un usuario con nombre de usuario "emqx_u". La contraseña se ha hasheado utilizando la función SHA2 junto con un valor "foo123" para aumentar la seguridad de la contraseña almacenada. Además, este usuario se ha marcado como superusuario, lo que le otorga privilegios adicionales en el bróker MQTT.

También se creó la tabla "mqtt_acl", que se utiliza para definir listas de control de acceso (ACL) que determinan qué acciones pueden realizar los usuarios MQTT en diferentes tópicos (topics). La sentencia DDL que se usó para crear esta tabla es el siguiente:

```
CREATE TABLE `mqtt_acl` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `ipaddress` VARCHAR(60) NOT NULL DEFAULT '',
  `username` VARCHAR(255) NOT NULL DEFAULT '',
  `clientid` VARCHAR(255) NOT NULL DEFAULT '',
  `action` ENUM('publish', 'subscribe', 'all') NOT NULL,
  `permission` ENUM('allow', 'deny') NOT NULL,
  `topic` VARCHAR(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figura 4.49. Comando para crear la tabla 'mqtt_acl' en MySQL.

La tabla "mqtt_acl" consta de los siguientes campos:

- **id:** Un identificador único para cada entrada de la lista de control de acceso (autoincremental).
- **ipaddress:** La dirección IP desde la cual se puede realizar la acción definida en la entrada.
- **username:** El nombre de usuario del cliente MQTT al que se aplicará la entrada ACL.
- **clientid:** El identificador del cliente MQTT al que se aplicará la entrada ACL.
- **action:** El tipo de acción permitida o denegada en el tópico (publish, subscribe o todas "all").
- **permission:** Indica si la acción especificada está permitida o denegada para el usuario (permitida "allow" o denegada "deny").
- **topic:** El tópico (topic) al que se aplica la entrada ACL.

```
MariaDB [emqx]> CREATE TABLE `mqtt_acl` (
    ->   `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
    ->   `ipaddress` VARCHAR(60) NOT NULL DEFAULT '',
    ->   `username` VARCHAR(255) NOT NULL DEFAULT '',
    ->   `clientid` VARCHAR(255) NOT NULL DEFAULT '',
    ->   `action` ENUM('publish', 'subscribe', 'all') NOT NULL,
    ->   `permission` ENUM('allow', 'deny') NOT NULL,
    ->   `topic` VARCHAR(255) NOT NULL DEFAULT '',
    ->   PRIMARY KEY (`id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
Query OK, 0 rows affected (0.00 sec)
```

Figura 4.50. Ejecución del comando para crear la tabla ‘mqtt_user’ en MySQL en el servidor.

Para permitir que un usuario específico realice, por ejemplo, la acción “publicar” (publish) en un tópico (topic) determinado, se puede insertar una nueva entrada en la tabla "mqtt_acl". A continuación, se muestra un ejemplo del comando SQL para hacer esto:

```
INSERT INTO mqtt_acl(username, permission, action, topic,
ipaddress) VALUES ('user123', 'allow', 'publish',
'data/user123/#', '127.0.0.1');
```

Figura 4.51. Ejemplo de comando para insertar un registro a la tabla ‘mqtt_acl’ en MySQL.

En este ejemplo, se permite que el usuario “user123” efectúe la acción “publicar” en cualquier subtópico (topic) bajo “data/user123/#” y se restringe el acceso a esta operación desde la dirección IP “127.0.0.1”.

Por otra parte, se efectuó la configuración necesaria para utilizar la base de datos MySQL en Node-RED con el propósito de almacenar los mensajes enviados a través de MQTT. La base de datos en la que se almacenan estos datos se nombró "mqtt" y se creó con el siguiente comando DDL:

```
CREATE DATABASE mqtt;
```

Figura 4.52. Comando para crear la base de datos 'mqtt' en MySQL.

El primer paso que se efectuó fue crear un usuario en MySQL que permita que el servicio de Node-RED se conecte a la base de datos y pueda insertar nuevos registros. El siguiente comando SQL es el que se encargó de esta tarea:

```
CREATE USER 'nodered'@'localhost' IDENTIFIED BY 'contraseña';
```

Figura 4.53. Comando para crear el usuario 'nodered'@'localhost' en MySQL.

Este comando crea un usuario con el nombre de "nodered" y restringe su acceso solo al equipo local (localhost), es decir, al propio servidor. Además, se le asigna una contraseña segura para garantizar la seguridad de la conexión.

No obstante, el usuario "nodered" necesita permisos específicos para interactuar con la base de datos "MQTT". En este caso, se le otorga el permiso de inserción (INSERT) en la base de datos "mqtt". Los siguientes comandos SQL cumplieron con esta tarea:

```
GRANT INSERT ON mqtt.* TO 'nodered'@'localhost';
FLUSH PRIVILEGES;
```

Figura 4.54. Comando para otorgar permisos de inserción a 'nodered'@'localhost' en MySQL.

Con esta configuración, el usuario "nodered" puede insertar nuevos registros en cualquier tabla de la base de datos "mqtt".

Luego, se creó la tabla "mqtt_data", la cual es utilizada para almacenar los mensajes MQTT recibidos y enviados. A continuación, se presenta el comando SQL que utilizamos para crear la tabla con la estructura adecuada:

```
CREATE TABLE mqtt_data (
    id int(15) unsigned NOT NULL AUTO_INCREMENT,
    topic varchar(100) NOT NULL COMMENT 'Topic',
    value double NOT NULL COMMENT 'Value',
    context varchar(255) DEFAULT NULL COMMENT 'Context',
    created datetime DEFAULT NULL COMMENT 'Created',
    PRIMARY KEY (id),
    INDEX (topic),
    INDEX (created)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figura 4.55. Ejemplo de comando para insertar un registro a la tabla 'mqtt_user' en MySQL.

La tabla "mqtt_data" contiene los siguientes campos:

- **id:** Un identificador único para cada registro (autoincremental).
- **topic:** El tópico (topic) asociado al mensaje MQTT.
- **value:** El valor del mensaje (se asume como valor numérico, de ahí que se utilice el tipo "double").
- **context:** Información adicional o contexto del mensaje (opcional).
- **created:** La fecha y hora en que se recibió o envió el mensaje.

```
MariaDB [emqx]> CREATE TABLE mqtt_data (
    ->     id int(15) unsigned NOT NULL AUTO_INCREMENT,
    ->     topic varchar(100) NOT NULL COMMENT 'Topic',
    ->     value double NOT NULL COMMENT 'Value',
    ->     context varchar(255) DEFAULT NULL COMMENT 'Context',
    ->     created datetime DEFAULT NULL COMMENT 'Created',
    ->     PRIMARY KEY (id),
    ->     INDEX (topic),
    ->     INDEX (created)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
Query OK, 0 rows affected (0.00 sec)
```

Figura 4.56. Ejecución del comando para crear la tabla 'mqtt_data' en el servidor.

Para insertar un nuevo dato o mensaje MQTT en la tabla "mqtt_data", se utiliza un comando SQL similar al siguiente ejemplo:

```
INSERT INTO mqtt_data (topic, value, context, created) VALUES
('gesture/mouth_open', 1, 'Información adicional', '2023-07-29
12:34:56');
```

Figura 4.57. Ejemplo de comando para insertar un registro a la tabla 'mqtt_data' en MySQL.

En este ejemplo, se inserta un nuevo registro con los siguientes valores:

- **topic:** "gesture/mouth_open"
- **value:** 1
- **context:** "Información adicional"
- **created:** "2023-07-29 12:34:56"

Esto representa un mensaje MQTT relacionado con la detección de un gesto por un usuario en el momento y fecha indicados.

Finalmente, se ha creado un usuario específico para la aplicación de Grafana. Con este usuario, Grafana puede acceder a los datos almacenados en la base de datos y mostrar información histórica sobre los gestos detectados. Se creó con el siguiente comando:

```
CREATE USER 'grafana'@'localhost' IDENTIFIED BY 'contraseña';
```

Figura 4.58. Comando para crear el usuario 'grafana'@'localhost' en MySQL.

Para permitir que Grafana acceda a los registros correspondientes a los gestos detectados, se le han otorgado los permisos de SELECT en la base de datos "mqtt" mediante las siguientes instrucciones:

```
GRANT SELECT ON mqtt.* TO 'grafana'@'localhost';
FLUSH PRIVILEGES;
```

Figura 4.59. Comando para otorgar permisos de selección a 'grafana'@'localhost' en MySQL.

4.9.5. EMQX

A continuación, se detalla el proceso de instalación y configuración de EMQX, un bróker MQTT, en el servidor. Además, se explica cómo se configuró la autenticación y autorización utilizando una base de datos MySQL.

Primeramente, para instalar EMQX en CentOS 7, descargamos e instalamos el repositorio donde están los binarios de EMQX para RPM con el comando:

```
curl -s https://assets.emqx.com/scripts/install-emqx-rpm.sh | sudo bash
```

Figura 4.60. Comando para añadir el repositorio de EMQX para RPM.

Luego, instalamos EMQX utilizando el gestor de paquetes yum:

```
sudo yum install emqx
```

Figura 4.61. Comando para instalar EMQX en el servidor.

Iniciamos el servicio de EMQX:

```
sudo systemctl start emqx
```

Figura 4.62. Comando para iniciar el servicio de EMQX en el servidor.

Para asegurarnos de que EMQX se inicie automáticamente con el arranque del sistema, ejecutamos el siguiente comando:

```
sudo systemctl enable emqx
```

Figura 4.63. Comando para habilitar el arranque automático del servicio de EMQX en el servidor.

Verificamos que EMQX esté en ejecución utilizando:

```
sudo systemctl status emqx
```

Figura 4.64. Comando para observar el estado del servicio de EMQX en el servidor.

```
[server@vxsct2810 ~]$ sudo systemctl status emqx
● emqx.service - emqx daemon
   Loaded: loaded (/usr/lib/systemd/system/emqx.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 14:50:36 -03; 11s ago
 Main PID: 80733 (beam.smp)
   CGroup: /system.slice/emqx.service
           ├─80733 emqx -Bd -spp true -A 4 -IOt 4 -SDio 8 -e 262144 -zdbbl 8192 -Q 1048576 -P 2097 ...
           ├─80976 erl_child_setup 1048576
           ├─81000 /usr/lib/emqx/lib/os_mon-2.8.2/priv/bin/memsup
           ├─81001 /usr/lib/emqx/lib/os_mon-2.8.2/priv/bin/cpu_sup
           ├─81021 /usr/lib/emqx/lib/jq-0.3.10/priv/erlang_jq_port
           └─81023 /usr/lib/emqx/lib/jq-0.3.10/priv/erlang_jq_port

Aug 02 14:50:36 vxsct2810 systemd[1]: Started emqx daemon.
Aug 02 14:50:37 vxsct2810 bash[80733]: WARNING: Default (insecure) Erlang cookie is in use.
Aug 02 14:50:37 vxsct2810 bash[80733]: WARNING: Configure node.cookie in /etc/emqx/emqx.conf o ... OKIE
Aug 02 14:50:37 vxsct2810 bash[80733]: WARNING: NOTE: Use the same cookie for all nodes in the ... ter.
Aug 02 14:50:39 vxsct2810 bash[80733]: Listener ssl:default on 0.0.0.0:8883 started.
Aug 02 14:50:39 vxsct2810 bash[80733]: Listener tcp:default on 0.0.0.0:1883 started.
Aug 02 14:50:39 vxsct2810 bash[80733]: Listener ws:default on 0.0.0.0:8083 started.
Aug 02 14:50:39 vxsct2810 bash[80733]: Listener wss:default on 0.0.0.0:8084 started.
Aug 02 14:50:39 vxsct2810 bash[80733]: Listener http:dashboard on :18083 started.
Aug 02 14:50:39 vxsct2810 bash[80733]: EMQX 5.1.4 is running now!
```

Figura 4.65. Estado del servicio de EMQX en el servidor.

Una vez que hemos verificado que EMQX está activo, procedemos a configurar la autenticación y autorización para el sistema. Para acceder al dashboard web de EMQX, es necesario configurar el proxy inverso (en nuestro caso, NGINX), asegurándonos de redirigir el subdominio específico a la dirección del dashboard. Sin embargo, esta configuración ya fue realizada en pasos anteriores, por lo que no se requiere realizar ninguna configuración adicional.

Cuando accedemos al dashboard de EMQX por primera vez, se nos solicita que cambiemos la contraseña de la cuenta de administrador, con el propósito de aumentar la seguridad. Efectuamos el cambio de contraseña y confirmamos.

Figura 4.66. Solicitud de cambio de contraseña al ingresar a EMQX por primera vez.

En las versiones posteriores a la 5, habilitar la autenticación basada en usuario y contraseña, al igual que la autorización, ya no requiere modificar archivos de configuración. Ahora, podemos realizar esta configuración directamente desde la interfaz web, simplificando el proceso. Anteriormente, esta configuración debía efectuarse modificando archivos de configuración ubicados en el directorio "/etc/emqx".

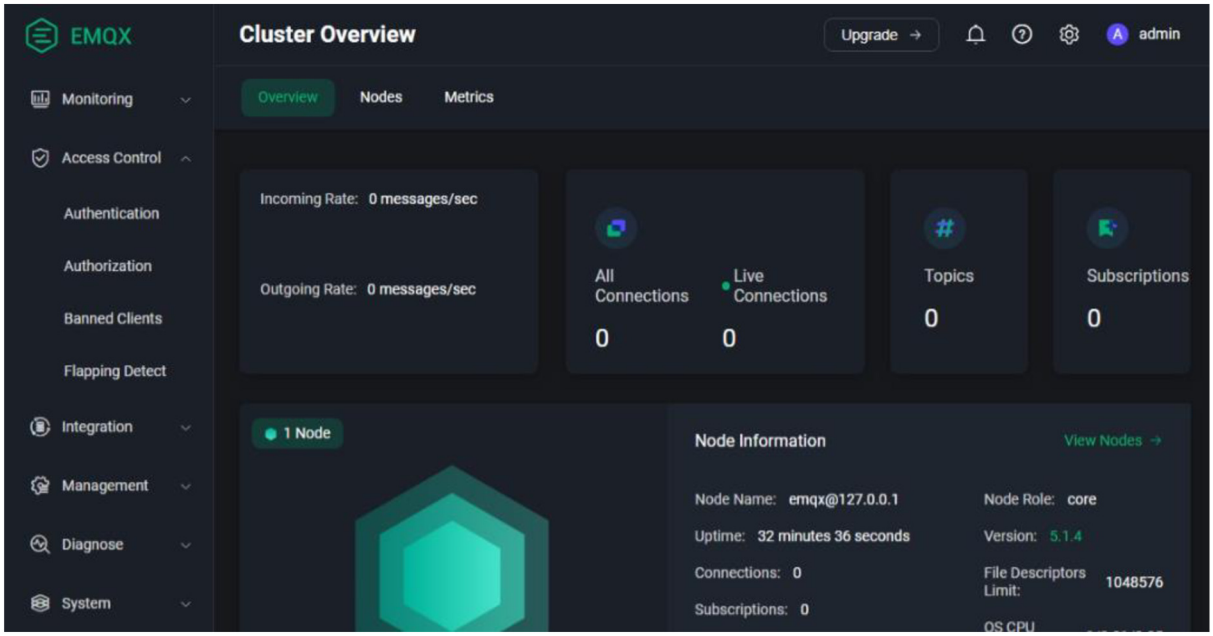

Figura 4.67. Vista del dashboard de EMQX.

Para habilitar el nuevo método de autenticación, simplemente hacemos clic en el botón "+ Create" en la sección de autenticación dentro del dashboard web. Este paso nos permite configurar las credenciales de acceso de manera sencilla y segura.

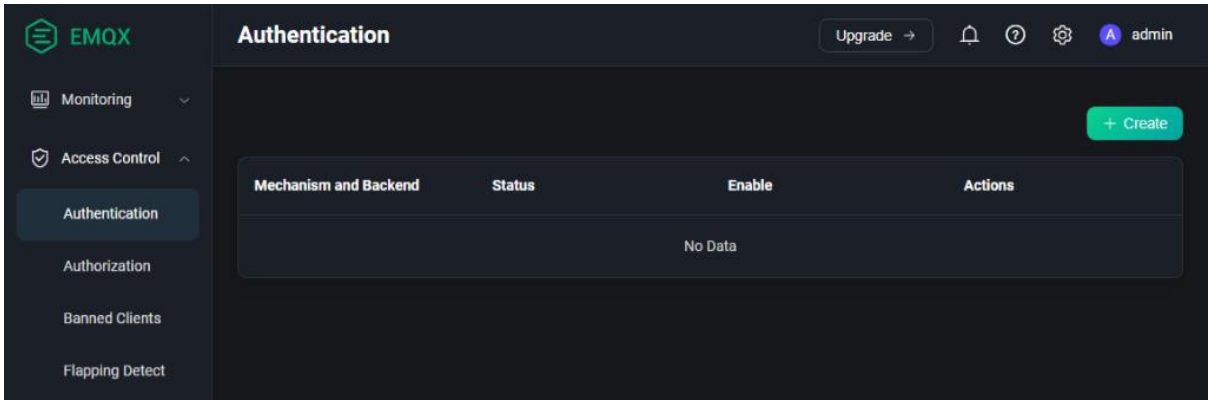

Figura 4.68. Vista para la gestión de la autenticación en EMQX.

A continuación, seleccionamos el tipo de autenticación que deseamos usar, que en este caso será por contraseña ("Password").

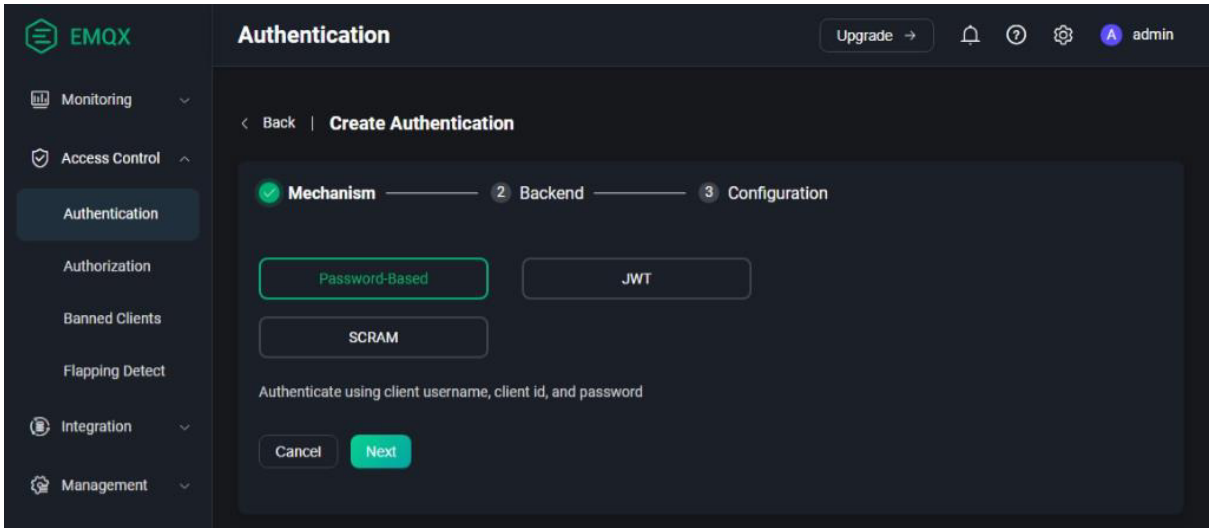

Figura 4.69. Primer paso en la creación de un nuevo mecanismo de autenticación en EMQX.

Luego, seleccionamos MySQL como la base de datos que contendrá nuestras contraseñas.

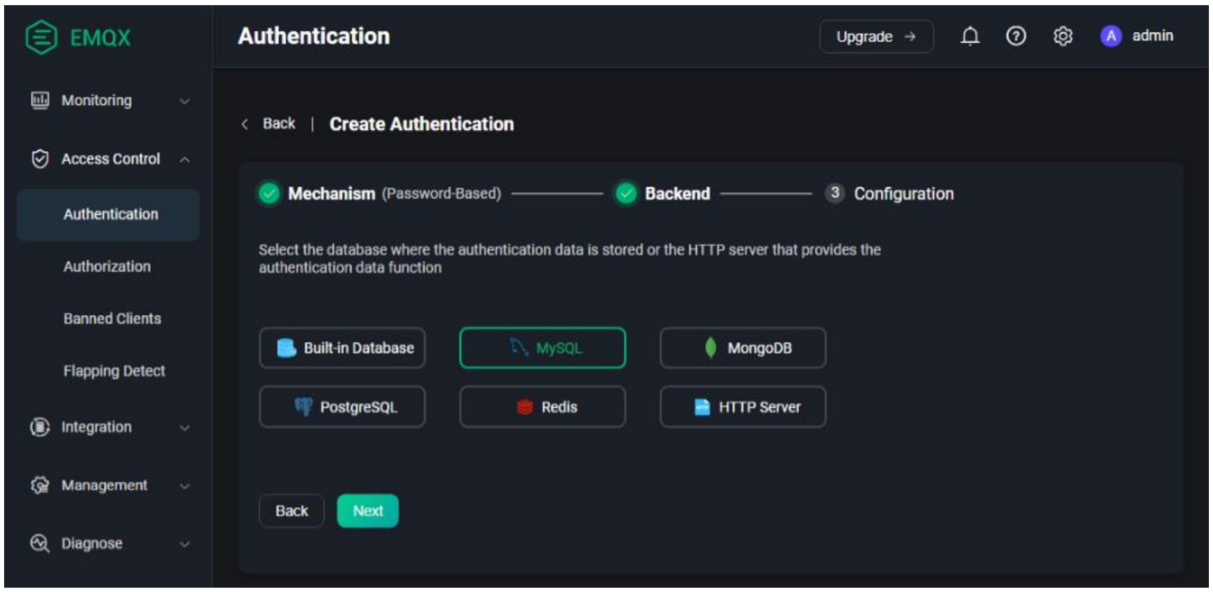

Figura 4.70. Segundo paso en la creación de un nuevo mecanismo de autenticación en EMQX.

Finalmente, completamos los datos correspondientes a la dirección IP y puerto, nombre de la base de datos, usuario y contraseña para que la aplicación pueda conectarse a la base de datos y obtener los registros necesarios.

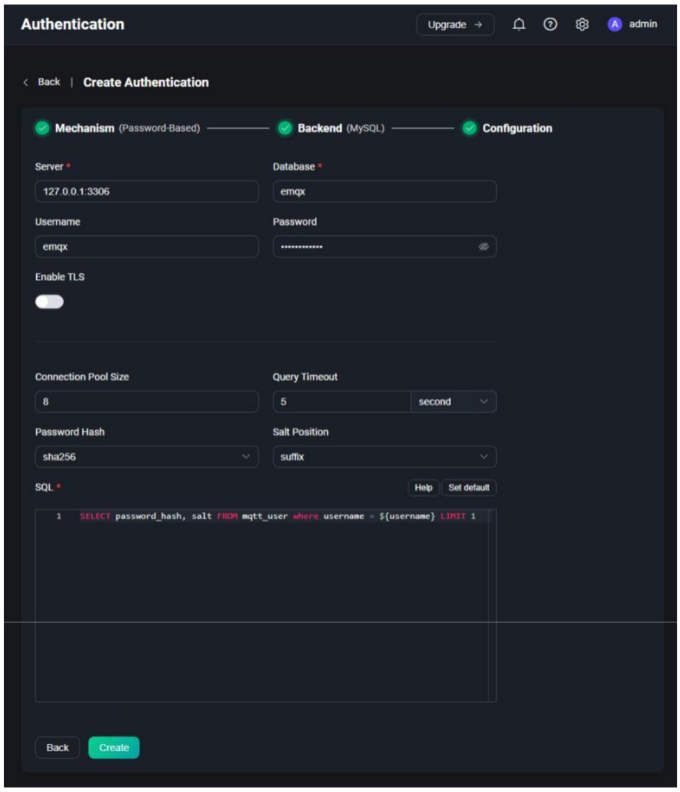

Figura 4.71. Tercer paso en la creación de un nuevo mecanismo de autenticación en EMQX.

Con esto, hemos añadido con éxito un nuevo método de autenticación basado en MySQL.

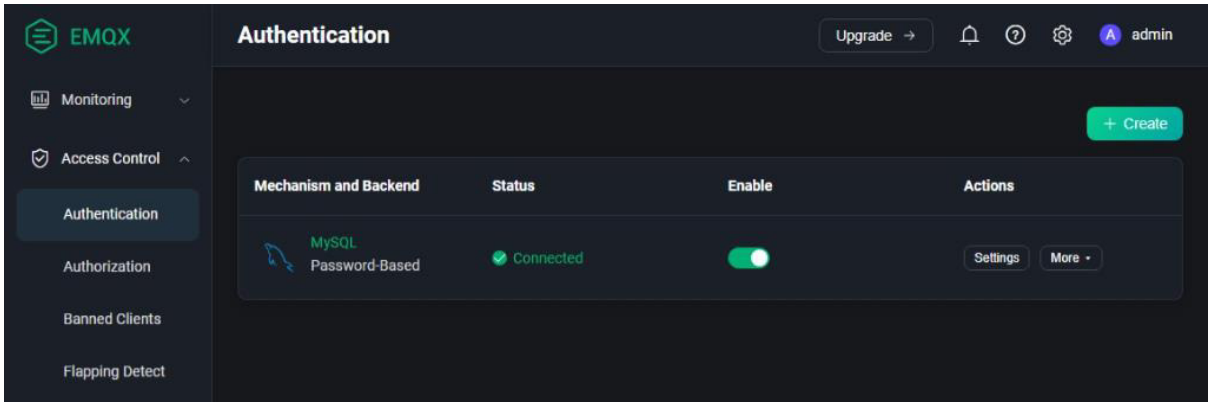

Figura 4.72. Vista para la gestión de la autenticación en EMQX, luego de añadir el mecanismo de MySQL.

El proceso para configurar la autorización mediante una base de datos MySQL es similar al de la autenticación. Dentro del dashboard web, nos dirigimos a la configuración de autorización y creamos un nuevo tipo de autorización.

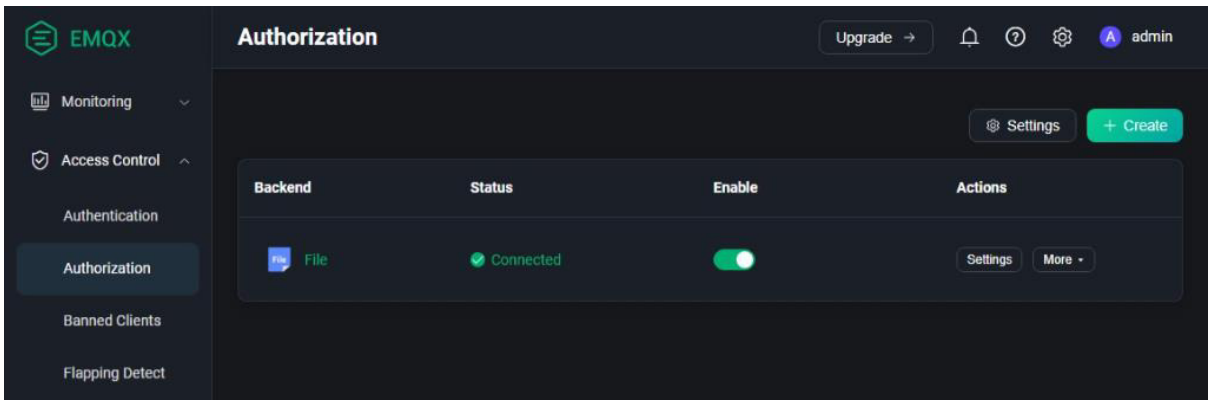

Figura 4.73. Vista para la gestión de la autorización en EMQX.

Seleccionamos “MySQL” como la base de datos que utilizará para la autorización.

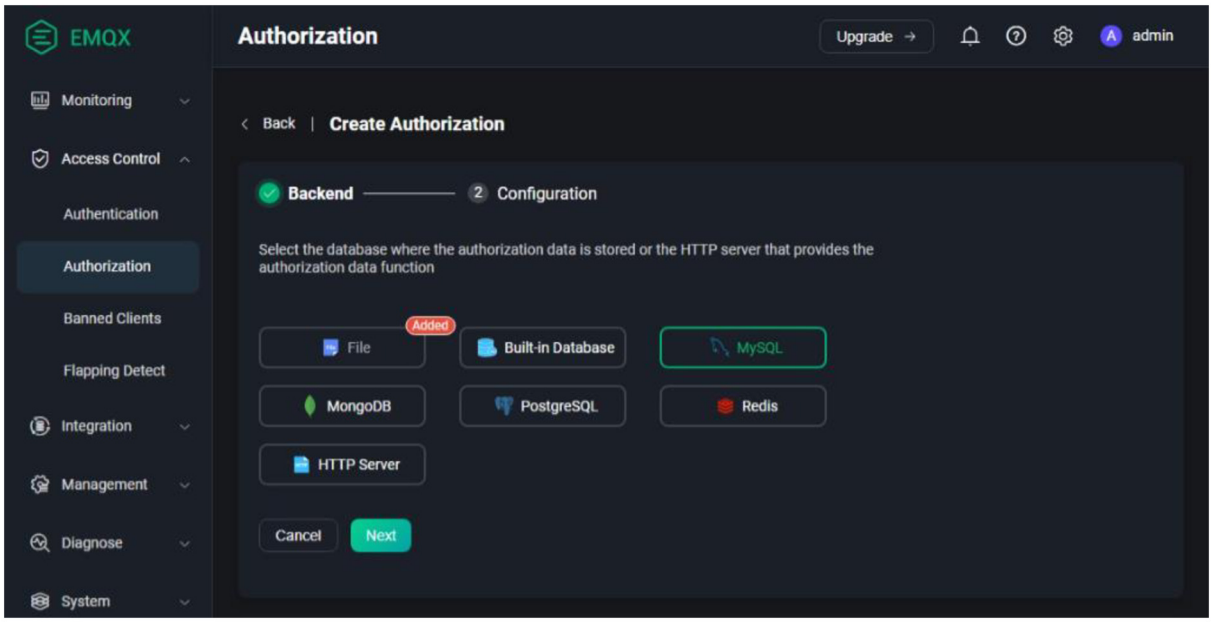

Figura 4.74. Primer paso para añadir un nuevo mecanismo de autorización en EMQX.

Luego, ingresamos la información correspondiente a la dirección IP y puerto, nombre de la base de datos, usuario y contraseña, y guardamos la configuración.

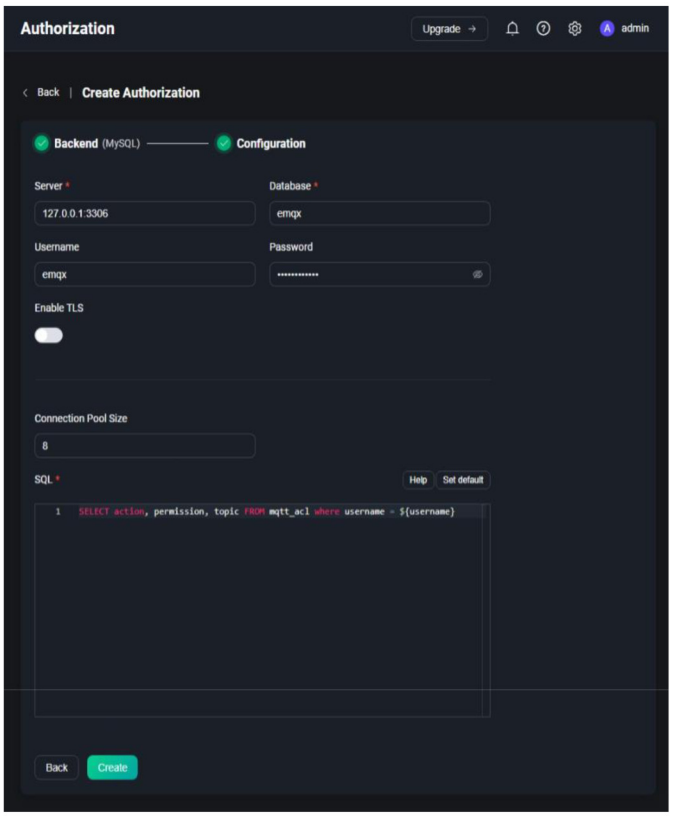

Figura 4.75. Primer paso para añadir un nuevo mecanismo de autorización en EMQX.

Así, hemos configurado con éxito la autorización mediante ACL (Control de Lista de Acceso) utilizando MySQL.

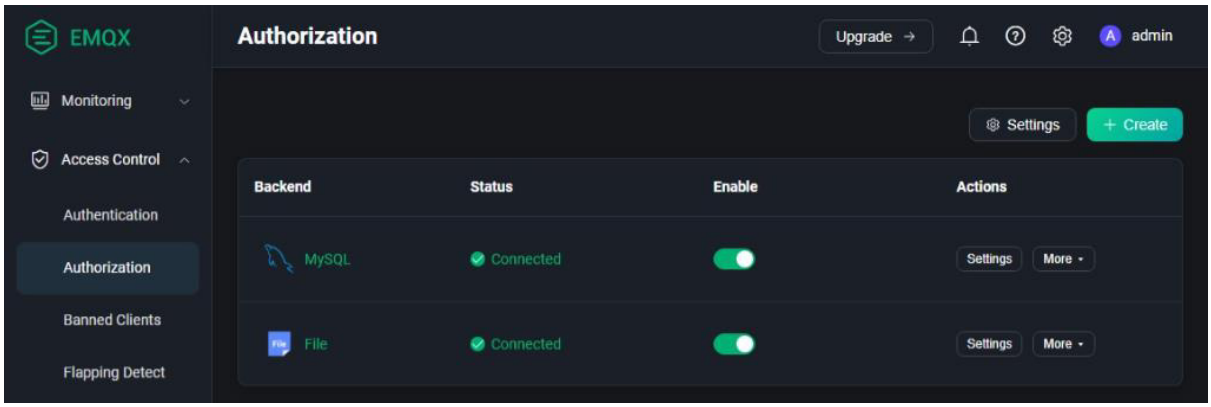

Figura 4.76. Vista de gestión de la autorización en EMQX, luego de añadir el mecanismo de autorización de MySQL.

En la imagen anterior, se observa como la configuración ha resultado exitosa.

4.9.6. Node-RED

Seguidamente, se detallan los pasos que se realizaron para la instalación y configuración de Node-RED en el servidor con sistema operativo CentOS 7.

Para la instalación y configuración de Node-RED, fue necesario contar con Node.js y npm instalados. Node.js es un entorno de ejecución de código abierto que permite ejecutar aplicaciones JavaScript en el lado del servidor. Utiliza el motor de JavaScript V8 de Google Chrome para ejecutar código fuera del navegador web.

Por otro lado, npm (Node Package Manager) es el gestor de paquetes predeterminado para Node.js. Permite a los desarrolladores instalar, compartir y administrar las dependencias y módulos que componen sus proyectos.

Con el propósito de instalar Node.js y npm en nuestro servidor, inicialmente se intentó realizar la instalación de Node.js v18.x. Para ello, ejecutamos el siguiente comando con el objetivo de añadir el repositorio de Node.js:

```
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
curl -sL https://dl.yarnpkg.com/rpm/yarn.repo | sudo tee
/etc/yum.repos.d/yarn.repo
```

Figura 4.77. Comandos para añadir el repositorio de Node.js para RPM.

Desafortunadamente, al intentar instalar esta versión de Node.js utilizando yum, se presentaron errores durante el proceso de instalación.

```
[server@vxsct2810 ~]$ sudo yum install nodejs
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.eze.sysarmy.com
 * epel: mirror.hnd.cl
 * extras: mirrors.eze.sysarmy.com
 * updates: mirrors.eze.sysarmy.com
yarn                                                      | 2.9 kB  00:00:00
yarn/primary_db                                           |  23 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package nodejs.x86_64 2:18.16.0-1nodesource will be installed
--> Processing Dependency: libstdc++.so.6(GLIBCXX_3.4.21)(64bit) for package: 2:nodejs-18.16.0-1nodes
ource.x86_64
--> Processing Dependency: libc.so.6(GLIBC_2.28)(64bit) for package: 2:nodejs-18.16.0-1nodesource.x86
_64
--> Processing Dependency: libm.so.6(GLIBC_2.27)(64bit) for package: 2:nodejs-18.16.0-1nodesource.x86
_64
--> Processing Dependency: libstdc++.so.6(CXXABI_1.3.9)(64bit) for package: 2:nodejs-18.16.0-1nodesou
rce.x86_64
--> Processing Dependency: libstdc++.so.6(GLIBCXX_3.4.20)(64bit) for package: 2:nodejs-18.16.0-1nodes
ource.x86_64
--> Finished Dependency Resolution
Error: Package: 2:nodejs-18.16.0-1nodesource.x86_64 (nodesource)
           Requires: libstdc++.so.6(GLIBCXX_3.4.20)(64bit)
Error: Package: 2:nodejs-18.16.0-1nodesource.x86_64 (nodesource)
           Requires: libstdc++.so.6(GLIBCXX_3.4.21)(64bit)
Error: Package: 2:nodejs-18.16.0-1nodesource.x86_64 (nodesource)
           Requires: libm.so.6(GLIBC_2.27)(64bit)
Error: Package: 2:nodejs-18.16.0-1nodesource.x86_64 (nodesource)
           Requires: libstdc++.so.6(CXXABI_1.3.9)(64bit)
Error: Package: 2:nodejs-18.16.0-1nodesource.x86_64 (nodesource)
           Requires: libc.so.6(GLIBC_2.28)(64bit)
 You could try using --skip-broken to work around the problem
 You could try running: rpm -Va --nofiles --nodigest
```

Figura 4.78. Errores ocurridos durante la instalación de Node.js.

Por tanto, para lograr una instalación exitosa de Node.js, fue necesario recurrir al manejador de paquetes dnf, aunque se obtuvo una versión anterior del software del repositorio EPEL.

```
sudo yum install dnf
```

Figura 4.79. Comando para instalar el manejador de paquetes dnf.

Entonces, instalamos Node.js y npm utilizando la herramienta dnf:

```
sudo dnf install nodejs
sudo dnf install npm
```

Figura 4.80. Comandos para instalar Node.js y npm utilizando dnf.

Para verificar que Node.js y npm se instalaron correctamente, utilizamos los siguientes comandos, los cuales muestran la versión instalada de ambas herramientas:

```
node -v
npm -v
```

Figura 4.81. Comandos para obtener la versión instalada de Node.js y npm.

```
[server@vxsct2810 ~]$ node -v
v16.18.1
[server@vxsct2810 ~]$ npm -v
8.19.2
```

Figura 4.82. Versión de Node.js y npm instaladas en el servidor.

Con Node.js y npm correctamente instalados, continuamos con la instalación de Node-RED y Node-RED Admin utilizando el gestor de paquetes npm:

```
sudo npm install -g --unsafe-perm node-red
sudo npm install -g --unsafe-perm node-red-admin
```

Figura 4.83. Comandos para instalar Node-RED y Node-RED Admin utilizando npm.

Una vez completada la instalación, generamos un hash para la contraseña del usuario administrador de Node-RED. Lo hicimos con el siguiente comando:

```
node-red admin hash-pw
```

Figura 4.84. Comando para generar el hash de la contraseña para un usuario de Node-RED.

Este comando solicita que se ingrese la contraseña que queremos utilizar. Luego, devuelve el hash generado, el cual utilizaremos más adelante para proteger el acceso al servicio:

```
[server@vxsct2810 ~]$ node-red admin hash-pw
Password:
$2b$08$a3UQyyv0Fpt5PSjxer3O6uTxJK0zsD.vJmE5bgr.qhV6yf0qsI9BG
```

Figura 4.85. Ejecución del comando para generar el hash de la contraseña para un usuario de Node-RED en el servidor.

A continuación, modificamos el archivo de configuración de Node-RED para incluir la autenticación con contraseña. Podemos ubicar el archivo de configuración con el siguiente comando:

```
node-red --settings
```

Figura 4.86. Comando para devolver la ubicación de los archivos de configuración de Node-RED.

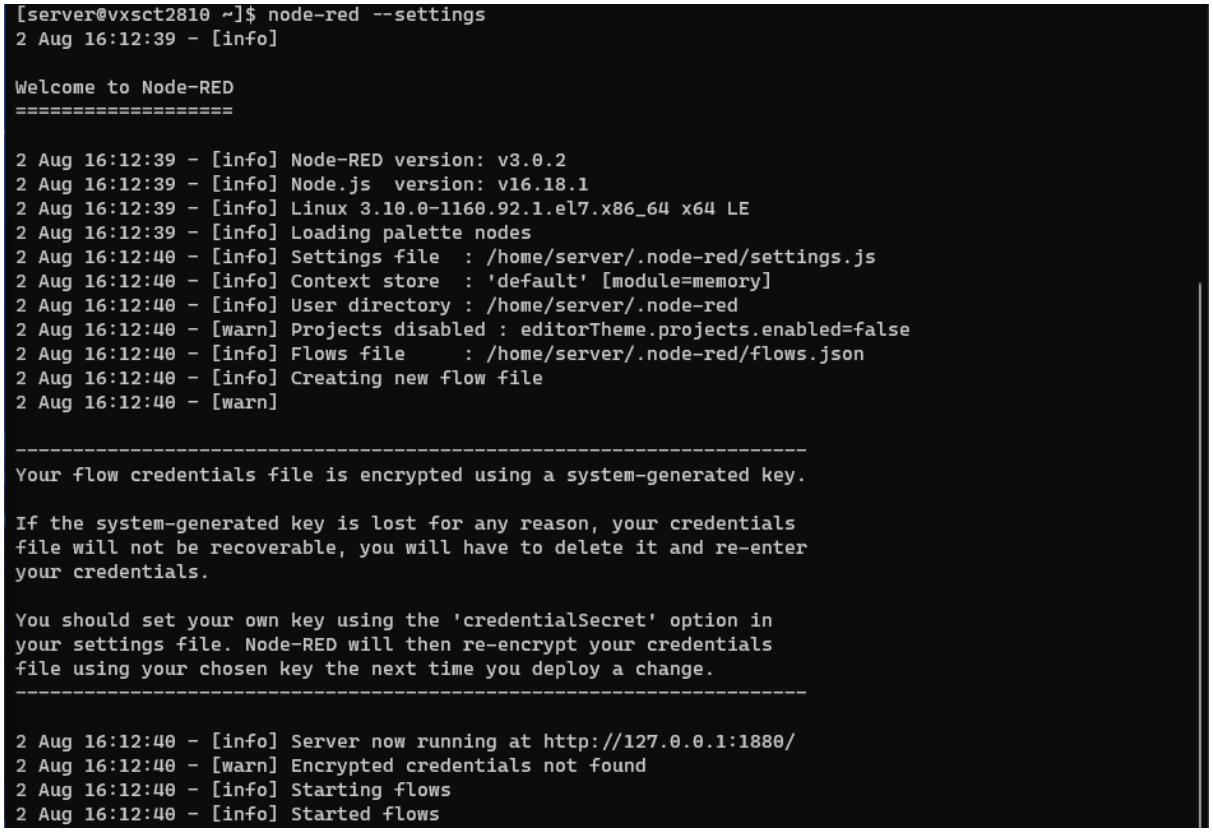

Figura 4.87. Ejecución del comando para devolver la ubicación de los archivos de configuración de Node-RED en el servidor.

Una vez que localizamos el archivo de configuración, procedemos a abrirlo utilizando un editor como nano y, a continuación, realizamos modificaciones en la sección correspondiente para habilitar la autenticación basada en credenciales.

```
adminAuth: {
   type: "credentials",
   users: [{
      username: "admin",
      password: "hash de la contraseña",
      permissions: "*"
  }]
},
```

Figura 4.88. Segmento del archivo de configuración de Node-RED en donde se configuran las credenciales de los usuarios.

Reemplazamos “hash de la contraseña” con el hash generado previamente. Con esto, la aplicación de Node-RED está protegida con la contraseña proporcionada.

```
    /** To password protect the Node-RED editor and admin API, the following
     * property can be used. See http://nodered.org/docs/security.html for details.
     */
    adminAuth: {
        type: "credentials",
        users: [{
            username: "admin",
            password: "$2b$08$a3UQyyv0Fpt5PSjxer3O6uTxJK0zsD.vJmE5bgr.qhV6yf0qsI9BG",
            permissions: "*"
        }]
    },
```

Figura 4.89. Edición del archivo de configuración de Node-RED para cambiar la contraseña por el hash generado anteriormente.

Finalmente, generamos un archivo de configuración de servicio de Node-RED, lo que nos permite gestionar el servicio utilizando la herramienta systemctl. Creamos un archivo llamado “node-red.service” en la ruta “/etc/systemd/system/”, que es donde se encuentran el resto de los archivos de configuración de los servicios, con el siguiente contenido.

```
[Unit]
Description=Node-RED
After=syslog.target network.target
Documentation=http://nodered.org/

[Service]
Environment="NODE_OPTIONS="
Environment="NODE_RED_OPTIONS=-v -u /home/server/.node-red"
ExecStart=/usr/local/bin/node-red $NODE_OPTIONS $NODE_RED_OPTIONS
User=root
Group=root
Nice=10
SyslogIdentifier=Node-RED
StandardOutput=syslog
Restart=on-failure
KillSignal=SIGINT

[Install]
WantedBy=multi-user.target
```

Figura 4.90. Archivo de configuración de servicio para Node-RED.

Es importante tener en cuenta que la ruta de directorio de trabajo (“home/server/.node-red” en este caso) varía dependiendo del usuario que instaló Node-RED en el servidor.

Una vez creado el archivo de servicio, activamos y habilitamos el servicio para que se inicie automáticamente al arrancar el sistema:

```
sudo systemctl start node-red.service
sudo systemctl enable node-red.service
```

Figura 4.91. Comandos para iniciar el servicio de Node-RED y habilitar el inicio automático.

Con esto, Node-RED ya está instalado y configurado en el servidor, protegido con una contraseña para mayor seguridad y configurado como un servicio gestionado por “systemctl”, lo que facilita su administración y arranque automático al inicio del sistema.

```
[server@vxsct2810 system]$ sudo systemctl status node-red
● node-red.service - Node-RED
   Loaded: loaded (/etc/systemd/system/node-red.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 17:15:27 -03; 1min 27s ago
     Docs: http://nodered.org/
 Main PID: 11184 (node-red)
   CGroup: /system.slice/node-red.service
           └─11184 node-red

Aug 02 17:15:28 vxsct2810 Node-RED[11184]: file will not be recoverable, you will have to delet ... ter
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: your credentials.
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: You should set your own key using the 'credentialSec ... in
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: your settings file. Node-RED will then re-encrypt yo ... als
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: file using your chosen key the next time you deploy ... ge.
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: ---------------------------------------------- ... ---
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: 2 Aug 17:15:28 - [warn] Encrypted credentials not found
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: 2 Aug 17:15:28 - [info] Server now running at http:/ ... 80/
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: 2 Aug 17:15:28 - [info] Starting flows
Aug 02 17:15:28 vxsct2810 Node-RED[11184]: 2 Aug 17:15:28 - [info] Started flows
Hint: Some lines were ellipsized, use -l to show in full.
```

Figura 4.92. Estado del servicio de Node-RED en el servidor.

Al intentar acceder al subdominio configurado para Node-RED, se solicitó la contraseña, tal como se había configurado previamente.

Figura 4.93. Vista de inicio de sesión de Node-RED.

Una vez que ingresamos la contraseña, pudimos acceder a la aplicación y así comenzar a configurar nuestra aplicación de flujos en Node-RED.

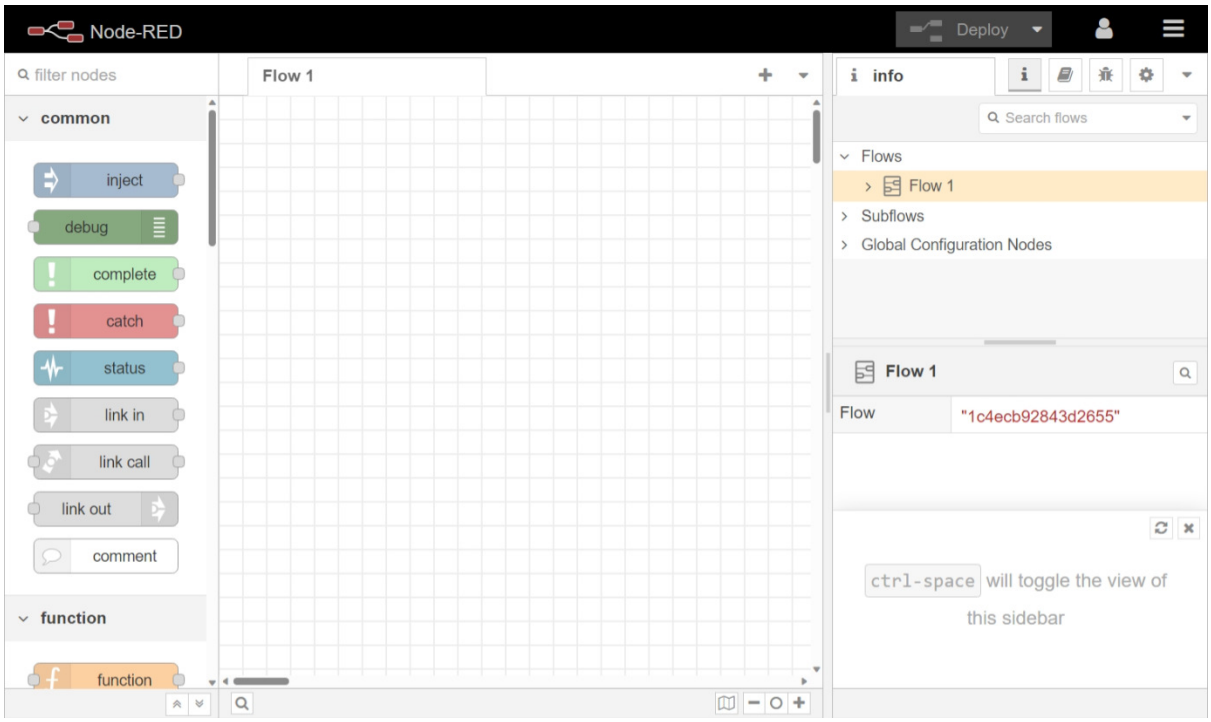

Figura 4.94. Vista principal de Node-RED.

Ya dentro de la aplicación, continuamos con la configuración de Node-RED para capturar los mensajes de control derivados de la detección de gestos enviados mediante MQTT, y a su vez, almacenar estos datos en una base de datos MySQL con el propósito de visualizarlos y analizarlos más adelante. A continuación, se presenta detalladamente el proceso paso a paso de cómo se llevó a cabo esta configuración.

El primer paso consistió en configurar el bróker MQTT. Para ello, en Node-RED, añadimos un nodo “MQTT In” desde la paleta de nodos. Sin embargo, para que el nodo se suscribiera al tópico donde se envían los gestos, era necesario configurar el bróker MQTT.

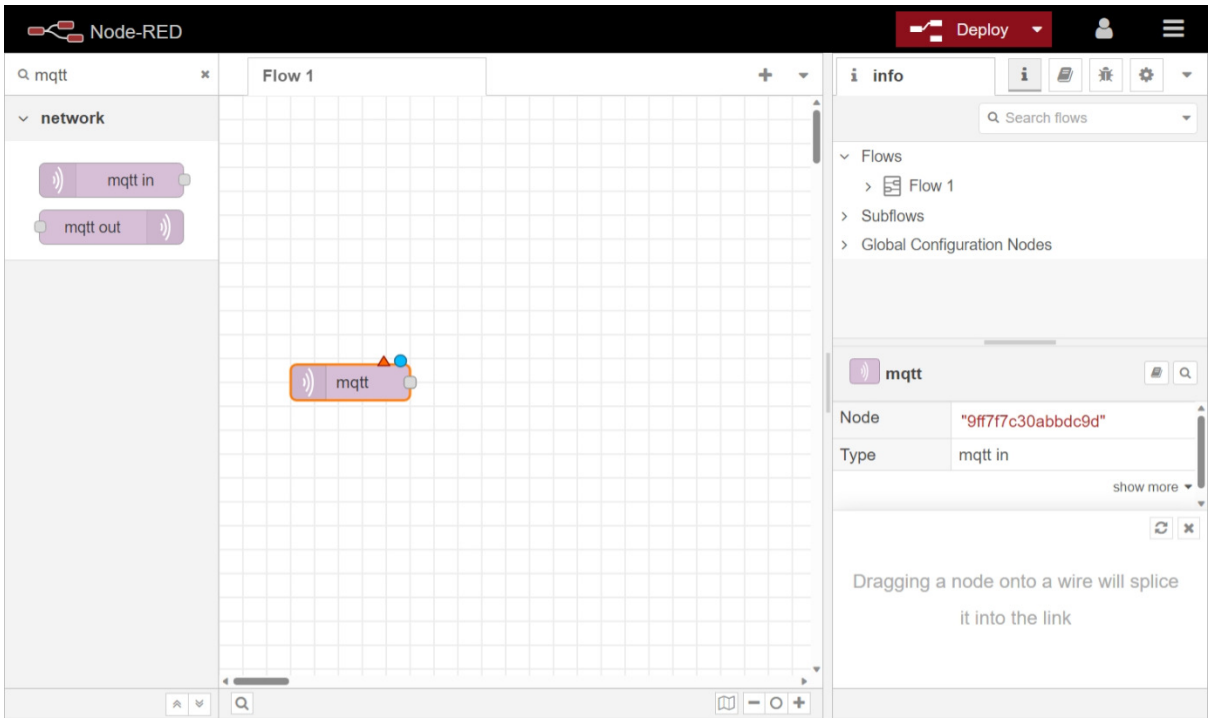

Figura 4.95. Adición del nodo "MQTT In" al flujo en Node-RED.

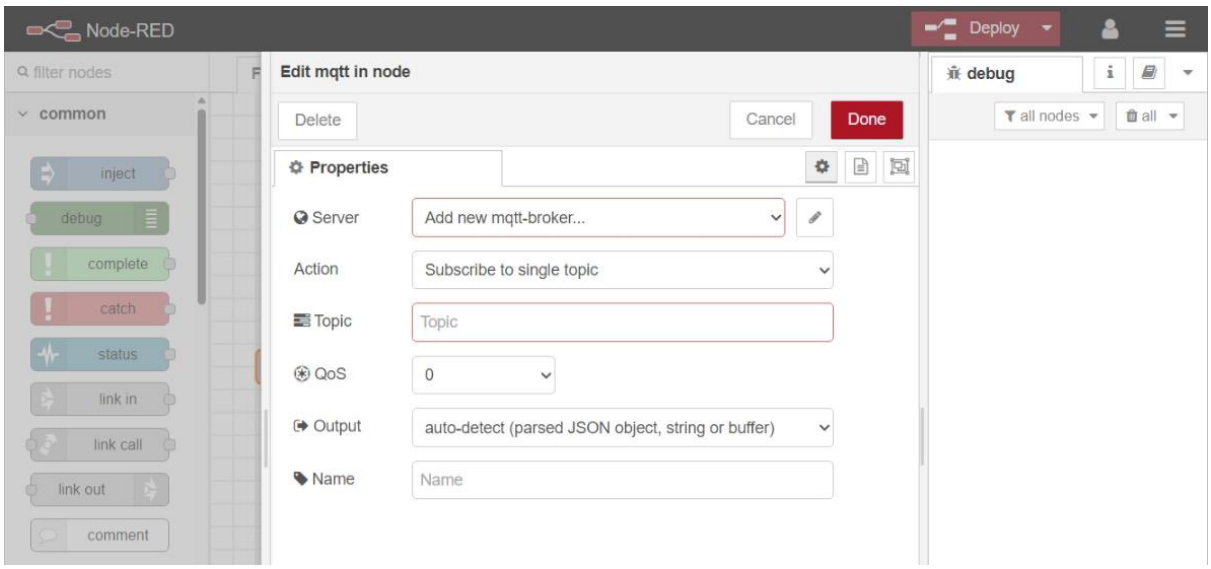

Figura 4.96. Vista de configuración del nodo "MQTT In" en Node-RED.

Para realizar esta configuración, en la pestaña de configuración del nodo "MQTT In", creamos una nueva conexión a un nuevo bróker MQTT ingresando la dirección IP y el puerto del bróker, la versión del protocolo a utilizar, y las credenciales necesarias para conectarse al bróker en la pestaña de seguridad. Una vez que ingresamos todos los datos, guardamos la configuración.

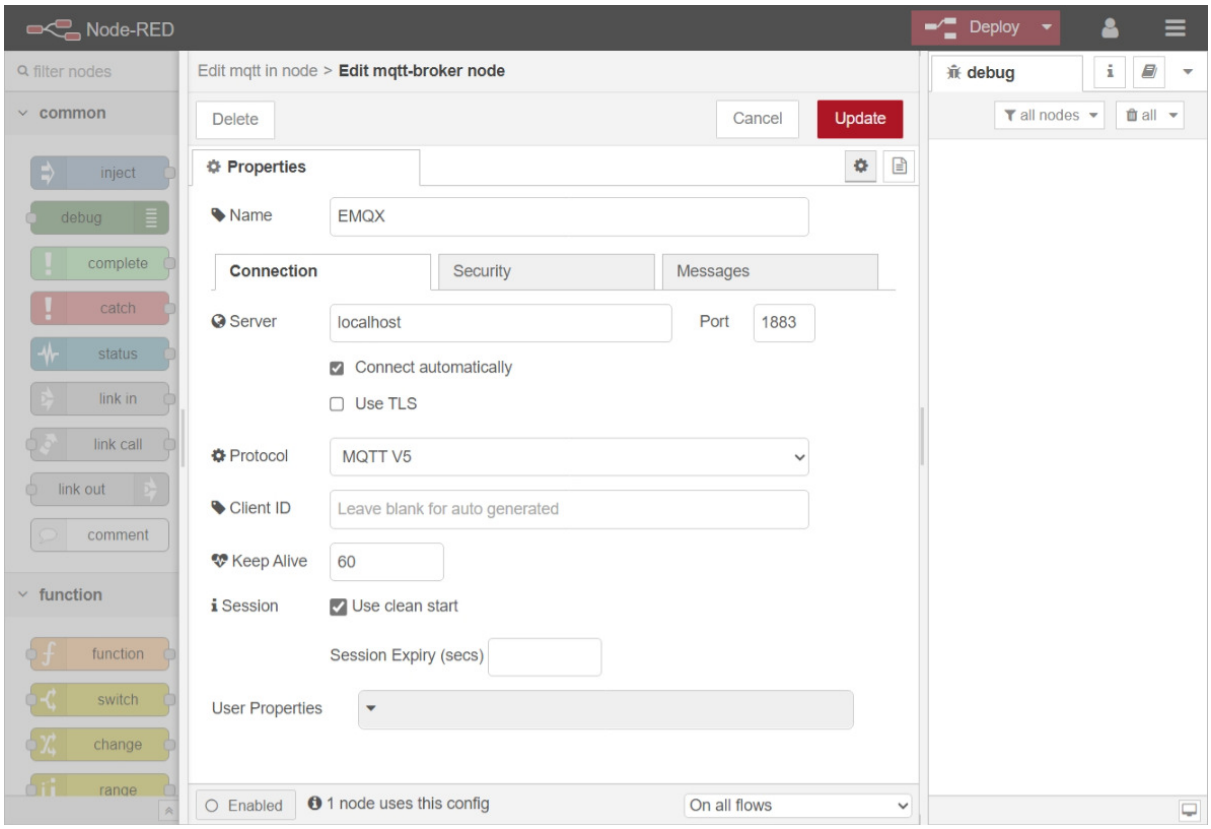

Figura 4.97. Configuración de la conexión con el bróker MQTT en Node-RED.

El siguiente paso fue configurar el tópico MQTT en el nodo “MQTT In”. En la ventana de configuración del nodo, escribimos el tópico al que queremos suscribirnos para escuchar los gestos detectados. En este caso, el tópico será “gesture/#”, para capturar todos los mensajes relacionados con gestos. Por último, guardamos toda la configuración para que el nodo esté listo para recibir los mensajes MQTT.

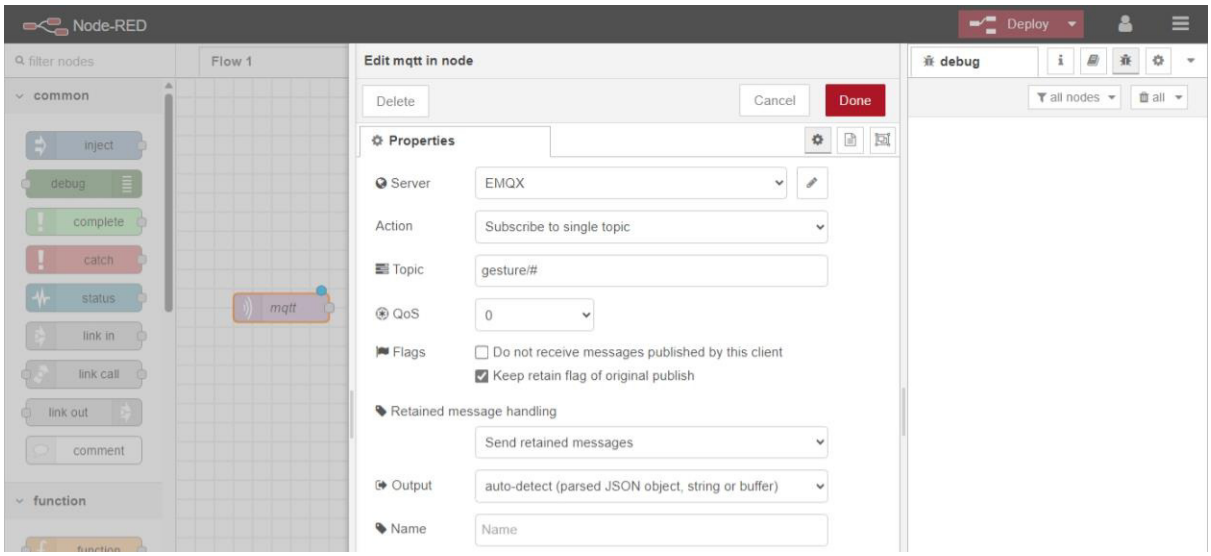

Figura 4.98. Configuración de la conexión al bróker MQTT en Node-RED.

Y de esta manera, ya tenemos el nodo "MQTT In" listo para recibir los mensajes relacionados con la detección de gestos.

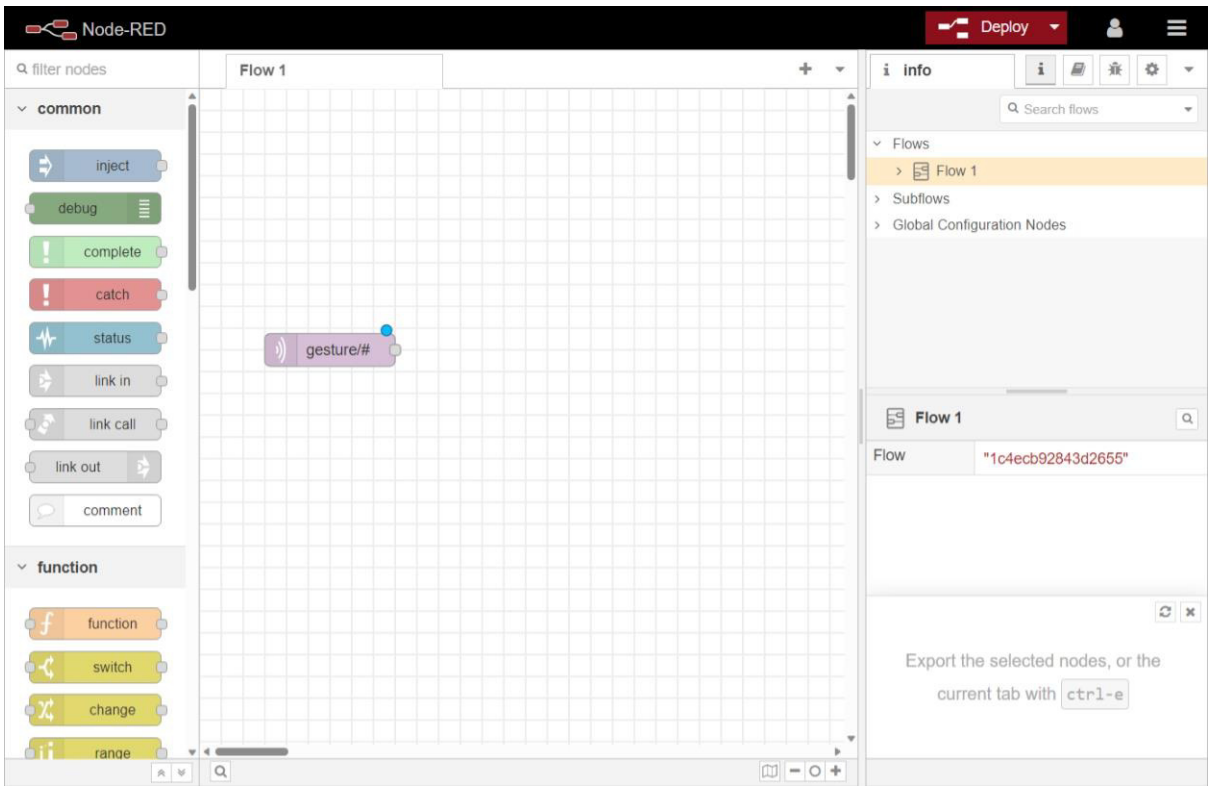

Figura 4.99. Vista del flujo luego de añadir y configurar el nodo "MQTT In" en Node-RED.

Sin embargo, antes de almacenar los datos en la base de datos MySQL, es necesario procesarlos para darles el formato apropiado, de manera que Node-RED pueda almacenarlos

correctamente. Para esto, utilizamos un nodo “function”, que nos permite agregar un script escrito en JavaScript.

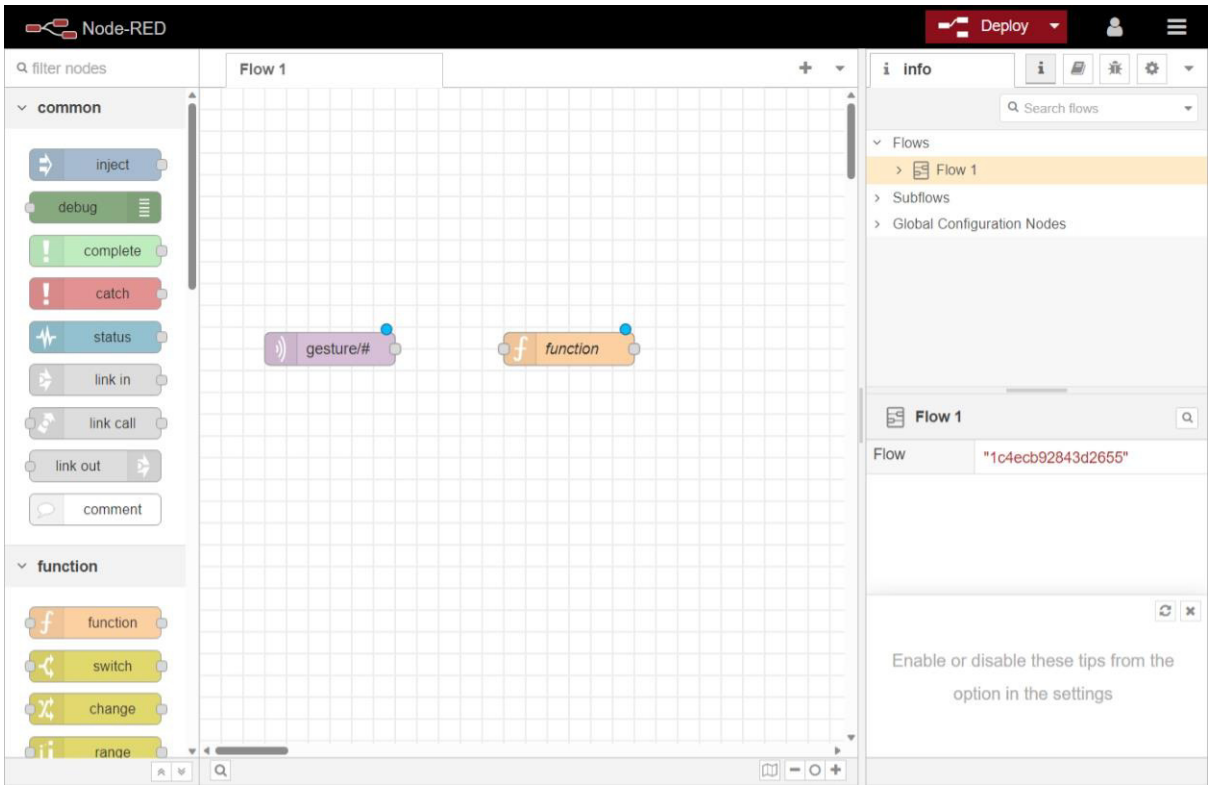

Figura 4.100. Vista del flujo luego de añadir el nodo “function” en Node-RED.

Esta función se encargará de generar la sentencia SQL que añadirá el nuevo registro en la base de datos, con el siguiente formato:

```
INSERT INTO mqtt_data (topic, value, context, created) VALUES ('tópico', 'valor', 'contexto', 'timestamp');
```

Figura 4.101. Comando generado para la inserción de un registro en la tabla ‘mqtt_data’.

La función toma el tópico y los valores recibidos a través del nodo “MQTT In” de Node-RED. Primero, toma el valor enviado en el mensaje MQTT, si existe, y lo almacena en una variable. Luego, verifica si hay una cadena de texto tipo string que brinde información adicional acerca del gesto detectado, y la almacena en una variable llamada “context”. A continuación, realiza lo mismo para el timestamp. Si existe, guarda este valor en la variable ‘timestamp’; de lo contrario, crea uno y lo almacena en esta misma variable.

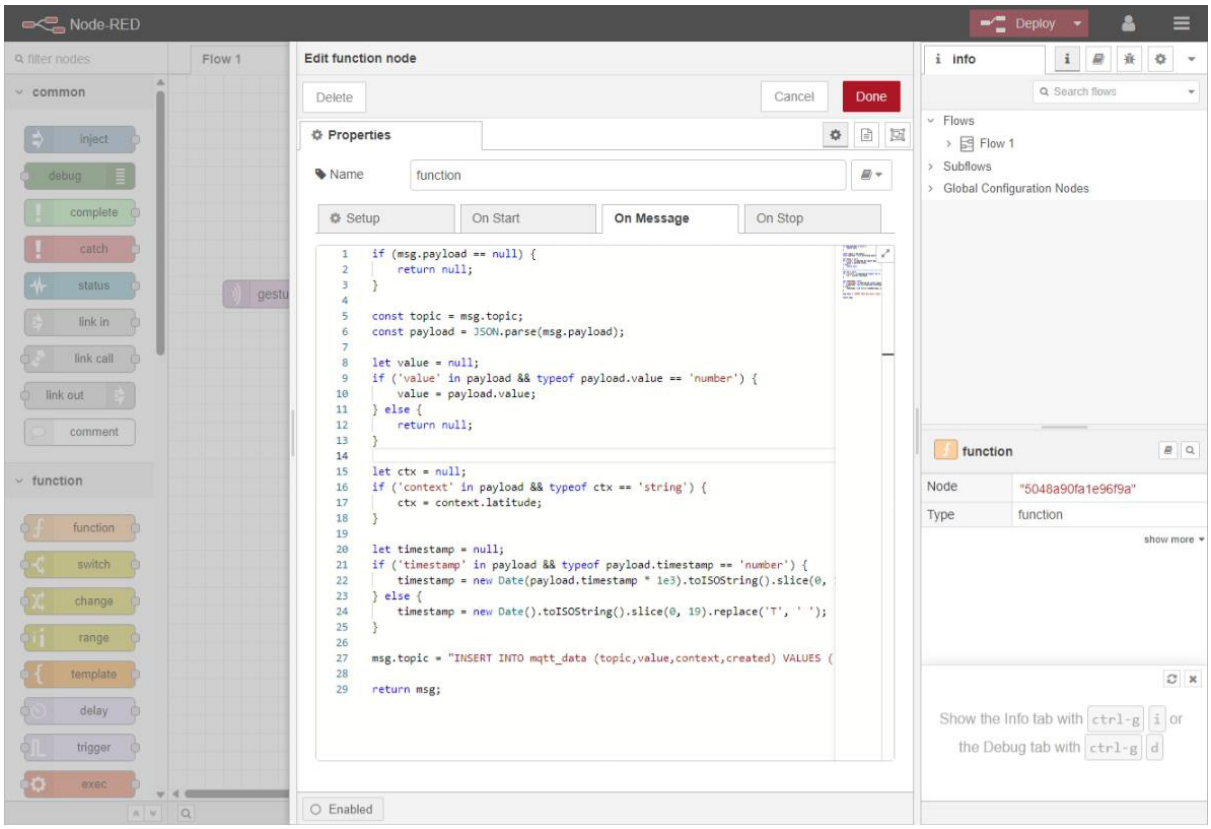

Figura 4.102. Código fuente añadido al nodo “function” en Node-RED.

Con el nodo “function” debidamente configurado para procesar los mensajes recibidos a través del nodo “MQTT In” y prepararlos para su almacenamiento en la base de datos MySQL, procedemos a conectarlo al nodo “MQTT In”. De esta manera, la salida del nodo “MQTT In” actuará como la entrada para el nodo “function”.

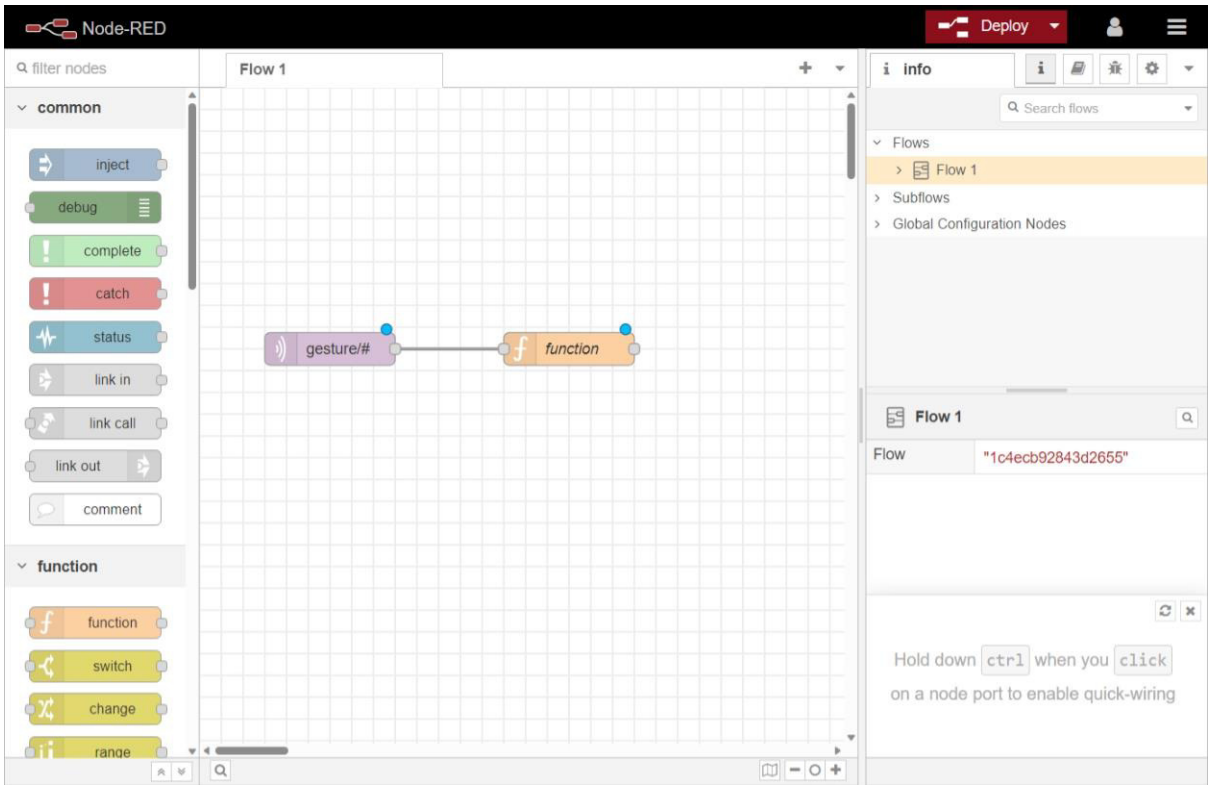

Figura 4.103. Vista del flujo luego de conectar los nodos “MQTT In” y “function” en Node-RED.

Con esta conexión establecida, los mensajes que lleguen al nodo “MQTT In” serán automáticamente procesados por el nodo “function” de acuerdo con el código que hemos definido. La función se encargará de darle el formato apropiado a los datos, creando la sentencia SQL necesaria para insertar un nuevo registro en la base de datos, con la información del tópico, el valor, el contexto y el timestamp correspondientes.

Ya solo nos queda escribir los datos procesados en la base de datos MySQL. Sin embargo, para escribir o leer una base de datos MySQL es necesario instalar el plugin “node-red-node-mysql”. Esto se hace desde la pestaña “Manage palette” en la sección “Home”.

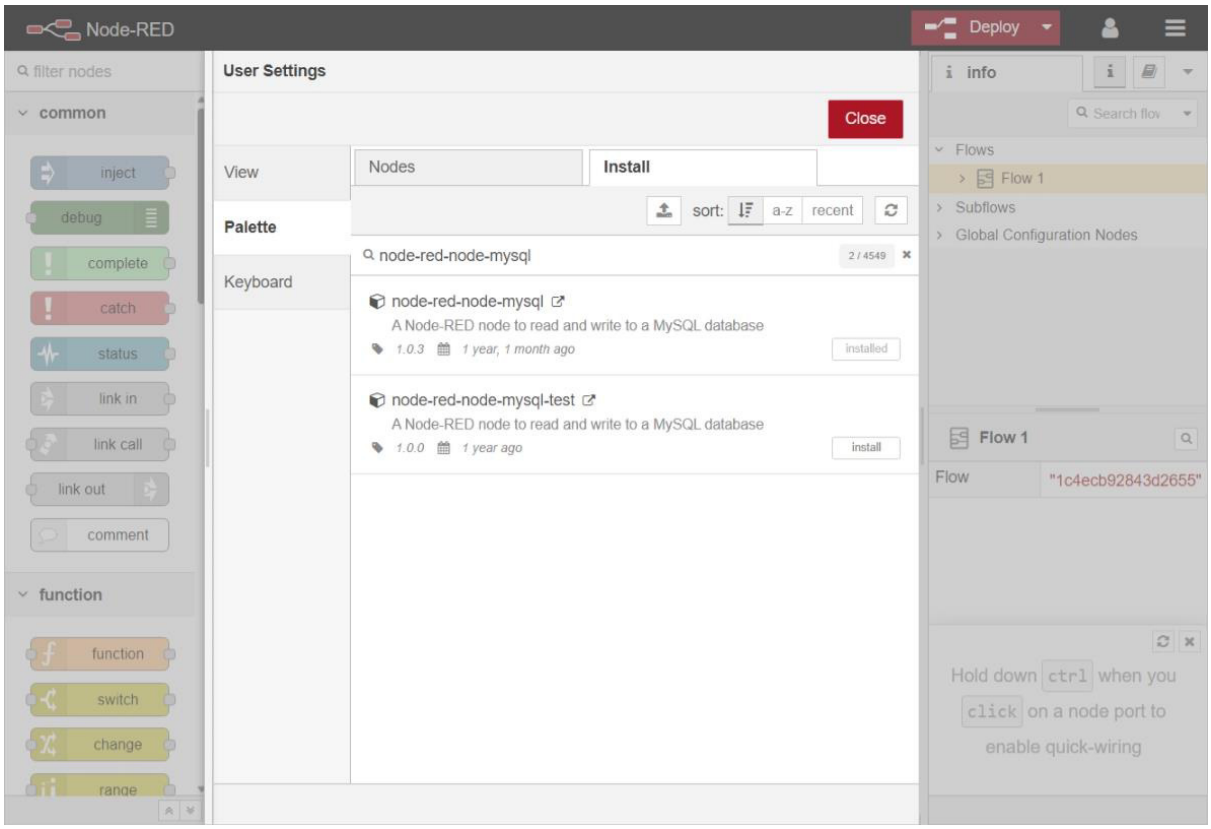

Figura 4.104. Instalación del plugin “node-red-node-mysql” en Node-RED.

Una vez instalado el plugin, ya contamos con los nodos que nos permiten establecer la comunicación con la base de datos MySQL.

El próximo paso es agregar este nodo “MySQL”, que será responsable de enviar al servicio de la base de datos la sentencia SQL generada por el nodo “function” para que se ejecute y se agregue un nuevo registro en la base de datos.

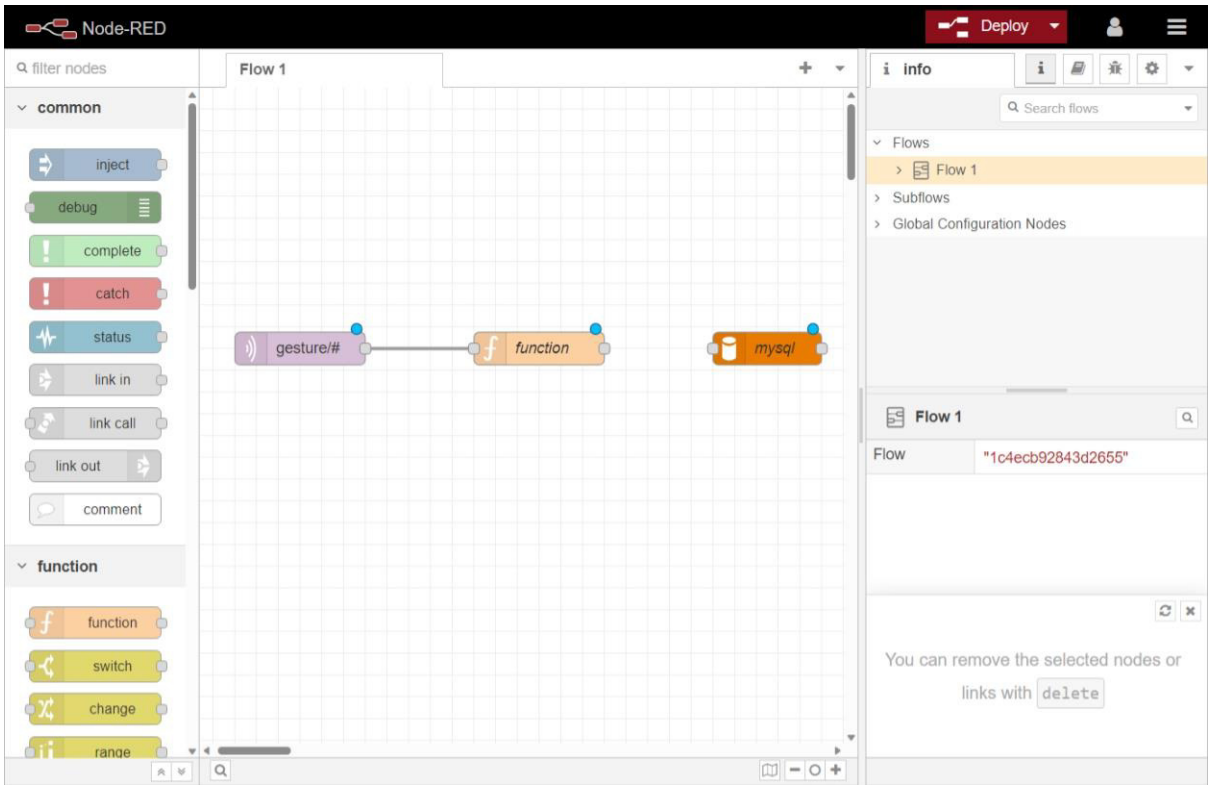

Figura 4.105. Vista del flujo luego de añadir el nodo “mysql” en Node-RED.

Para configurarlo, debemos crear una nueva conexión a una base de datos MySQL.

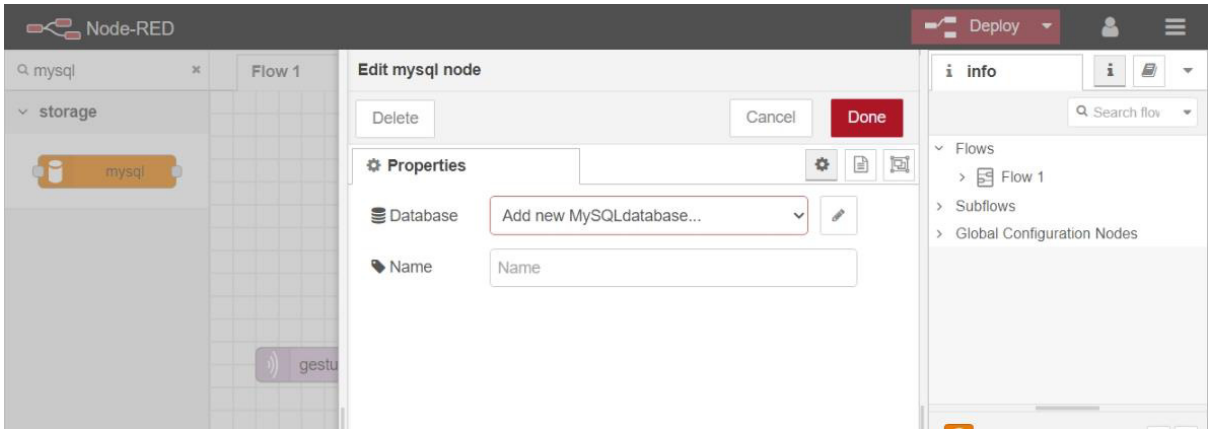

Figura 4.106. Vista de configuración del nodo “mysql” en Node-RED.

De manera similar a como lo hicimos con el bróker MQTT, ingresamos el nombre de usuario, la contraseña y el nombre de la base de datos donde estarán alojadas las tablas que manipularemos.

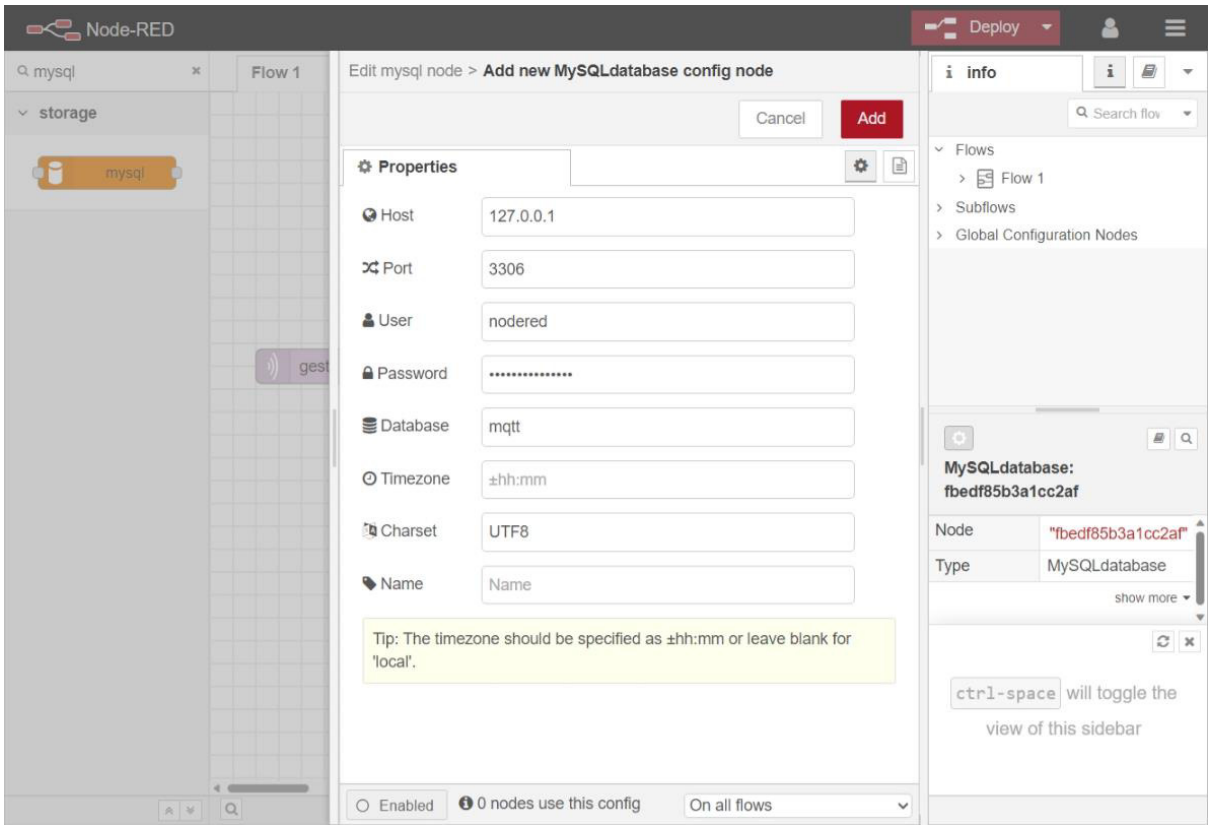

Figura 4.107. Configuración de la conexión con la base de datos MySQL en Node-RED.

Con la configuración de la base de datos lista, procedemos a conectar los tres nodos en el flujo de trabajo. La salida del nodo “function”, que contiene la sentencia SQL formateada, se conecta al nodo “MySQL”, que se encargará de ejecutar dicha sentencia en la base de datos especificada.

Una vez que los nodos están conectados, simplemente hacemos el despliegue o *deploy* en Node-RED, y el sistema estará listo para capturar los mensajes provenientes de MQTT y almacenarlos en la base de datos MySQL para su posterior análisis y visualización.

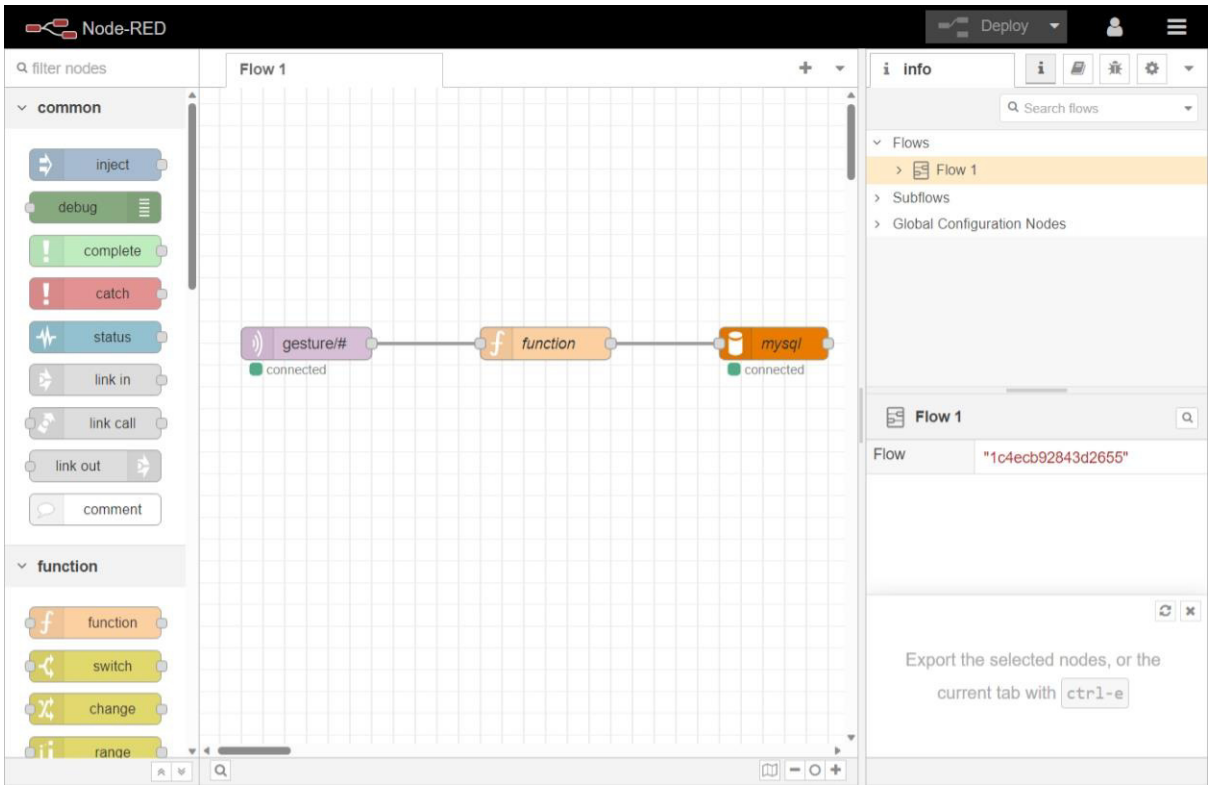

Figura 4.108. Vista del flujo terminado en Node-RED.

Así, ya hemos configurado el flujo completo que abarca desde la captura de mensajes de los gestos detectados mediante MQTT, el procesamiento en el nodo "function" y el almacenamiento de los datos en la base de datos MySQL. Este flujo permite capturar y mantener los datos en una base de datos persistente, lo que facilita su análisis y visualización en el futuro.

4.9.7. Grafana

A continuación, se detalla el procedimiento que se realizó para descargar, instalar y configurar Grafana en el servidor proporcionado por la universidad.

Para comenzar, en la terminal ejecutamos los siguientes comandos para instalar las dependencias necesarias:

```
sudo yum install -y epel-release
sudo yum install -y yum-utils
```

Figura 4.109. Comandos para instalar el repositorio EPEL y utilidades adicionales.

Luego, añadimos el repositorio de Grafana al sistema:

```
sudo rpm --import https://packages.grafana.com/gpg.key
sudo tee /etc/yum.repos.d/grafana.repo <<EOF
[grafana]
name=grafana
baseurl=https://packages.grafana.com/enterprise/rpm
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
EOF
```

Figura 4.110. Comando para añadir el repositorio de Grafana a RPM.

Después de agregar el repositorio, actualizamos el listado de paquetes e instalamos Grafana:

```
sudo yum update
sudo yum install grafana
```

Figura 4.111. Comandos para actualizar el sistema e instalar Grafana.

Como estamos usando systemd, iniciamos el servicio de Grafana y verificamos su estado mediante la herramienta systemctl:

```
sudo systemctl start grafana-server
sudo systemctl status grafana-server
```

Figura 4.112. Comandos para iniciar y ver el estado del servicio de Grafana.

```
[server@vxsct2810 system]$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 18:00:13 -03; 5s ago
     Docs: http://docs.grafana.org
 Main PID: 12020 (grafana)
   CGroup: /system.slice/grafana-server.service
           └─12020 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfi ...

Aug 02 18:00:13 vxsct2810 systemd[1]: Started Grafana instance.
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=modules t=2023-08-02T18:00:13.801597709-03:00 ....."
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=http.server t=2023-08-02T18:00:13.803236211-03 ... et=
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=ngalert.state.manager t=2023-08-02T18:00:13.80 ... up"
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=ngalert.state.manager t=2023-08-02T18:00:13.80…029µs
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=ticker t=2023-08-02T18:00:13.803443689-03:00 l ... :00
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=ngalert.multiorg.alertmanager t=2023-08-02T18: ... er"
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=grafanaStorageLogger t=2023-08-02T18:00:13.813 ... ng"
Aug 02 18:00:13 vxsct2810 grafana[12020]: logger=grafana.update.checker t=2023-08-02T18:00:13.9 ... 5ms
Aug 02 18:00:14 vxsct2810 grafana[12020]: logger=plugins.update.checker t=2023-08-02T18:00:14.0 ... 3ms
Hint: Some lines were ellipsized, use -l to show in full.
```

Figura 4.113. Estado del servicio de Grafana en el servidor.

Para asegurarnos de que Grafana se inicie junto al arranque del sistema, configuraremos el servicio para que se inicie automáticamente:

```
sudo systemctl enable grafana-server
```

Figura 4.114. Comando para habilitar el inicio automático del servicio de Grafana al arranque del sistema.

Para evitar el acceso anónimo a Grafana, editamos el archivo de configuración grafana.ini, que se encuentra en la carpeta "/etc/grafana/". Podemos usar un editor de texto como nano o vi para modificar el archivo:

```
sudo nano /etc/grafana/grafana.ini
```

Figura 4.115. Comando para editar el archivo de configuración de Grafana con el editor nano.

Dentro del archivo, buscamos la sección [auth.anonymous] y estableceremos la variable enabled en "false":

```
[auth.anonymous]
# enable anonymous access
enabled = false
```

Figura 4.116. Segmento del archivo de configuración de Grafana que se debe modificar para deshabilitar el inicio de usuarios anónimos (sin autorizarse).

```
################################## Anonymous Auth ######################
[auth.anonymous]
# enable anonymous access
enabled = false
```

Figura 4.117. Modificación del archivo de configuración de Grafana en el servidor para deshabilitar el inicio de usuarios anónimos en el servidor.

Guardamos los cambios y cerramos el editor. De esta manera, ya tenemos configurado el servicio de Grafana para que no permita el ingreso de usuarios sin credenciales a la aplicación.

Luego de realizar estas configuraciones, ya podemos acceder al dashboard de Grafana a través del subdominio configurado previamente con NGINX.

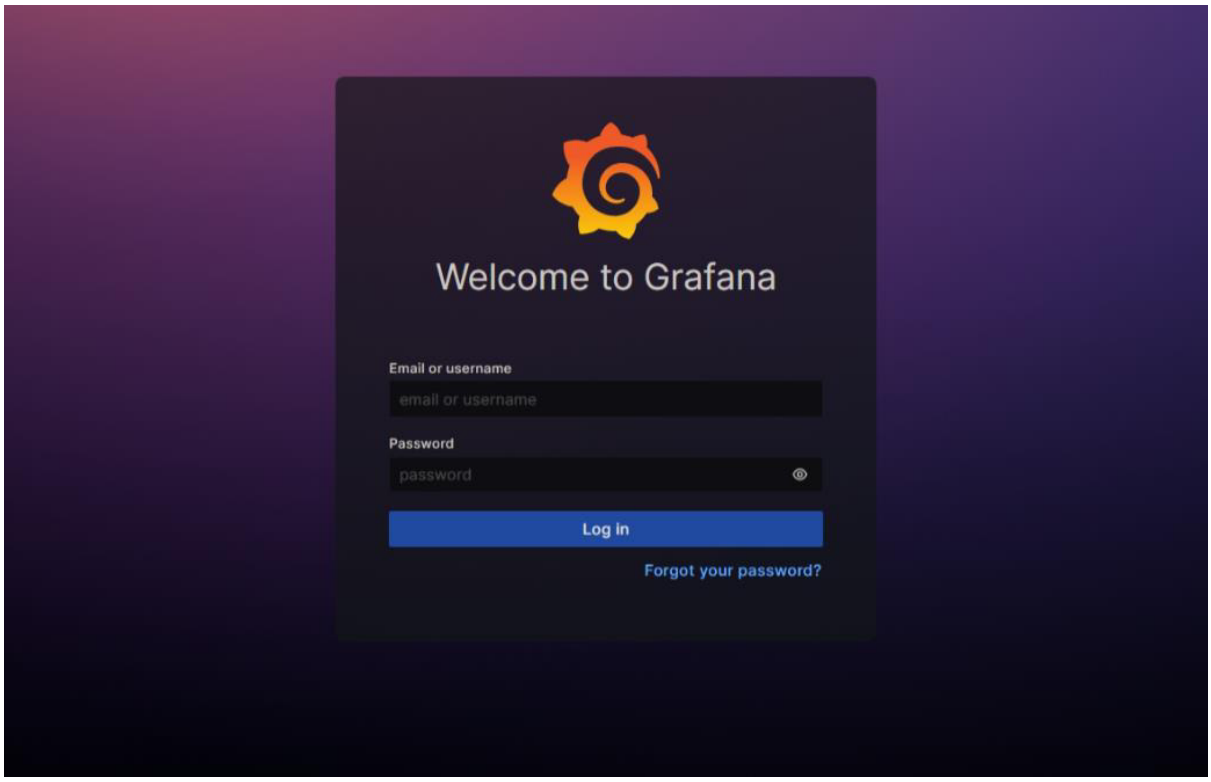

Figura 4.118. Vista de inicio de sesión de Grafana.

En la figura anterior se puede apreciar que, al intentar acceder a la aplicación web de Grafana, se solicita el usuario y contraseña, lo que indica que la configuración fue exitosa. Al igual que ocurrió con EMQX, al iniciar sesión por primera vez en Grafana, se nos solicitará cambiar la contraseña para asegurar una mayor protección contra posibles intrusos. Ingresamos la nueva contraseña y confirmamos para completar el proceso.

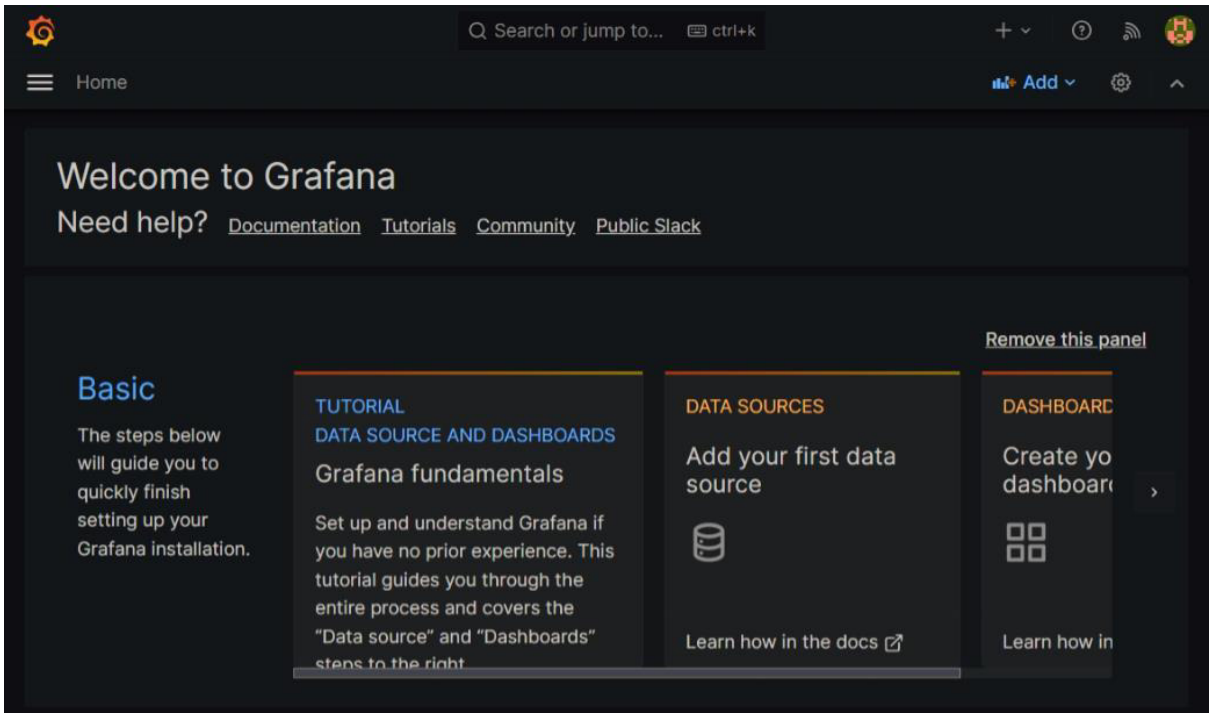

Figura 4.119. Vista de Grafana por defecto al iniciar sesión.

Dentro de la interfaz web de Grafana, se instaló el complemento "MQTT Client Datasource Plugin" con el propósito de permitir la suscripción y recepción de mensajes de control procedentes de la aplicación de procesamiento de imágenes.

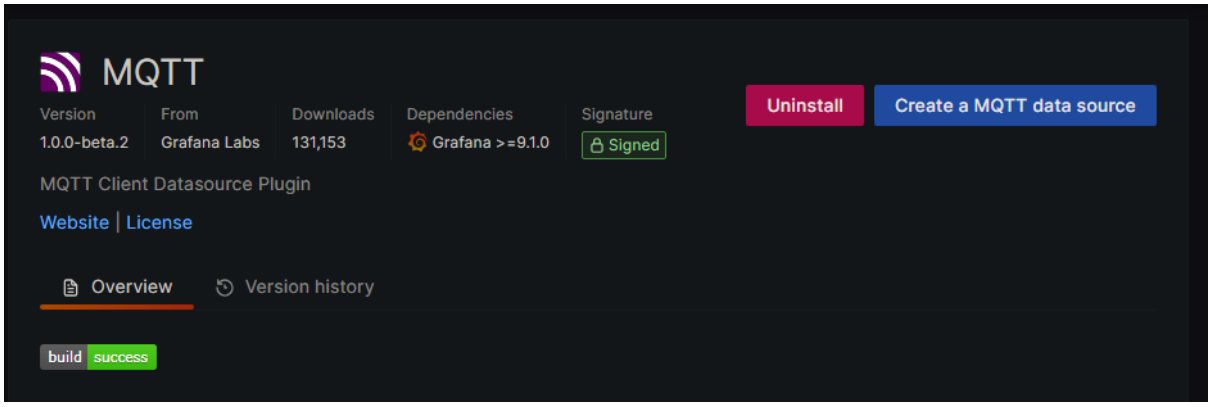

Figura 4.120. Instalación del complemento "MQTT Client Datasource Plugin" en Grafana.

Posteriormente, se configuraron las conexiones tanto para MQTT como para MySQL. De esta manera, Grafana es capaz de establecer conexiones tanto con el bróker MQTT como con la base de datos MySQL. Esto permite obtener y visualizar los datos necesarios en la plataforma de Grafana.

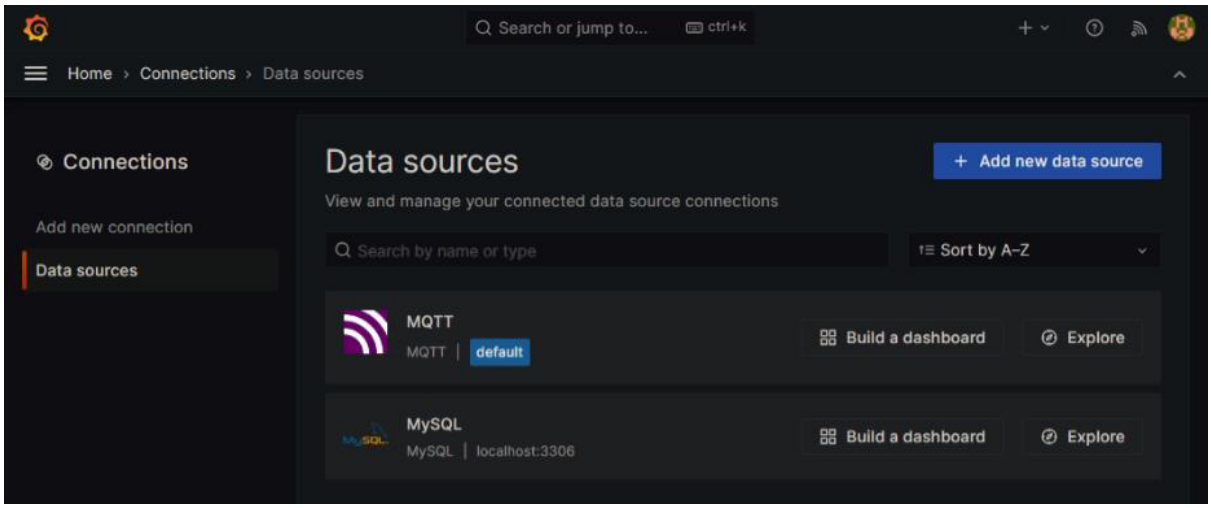

Figura 4.121. Vista de configuración de las fuentes de datos (data sources) en Grafana luego de añadir conexiones para MQTT y MySQL.

Finalmente, se creó el dashboard para que los usuarios puedan monitorear de manera efectiva el resultado del procesamiento de imágenes del sistema.

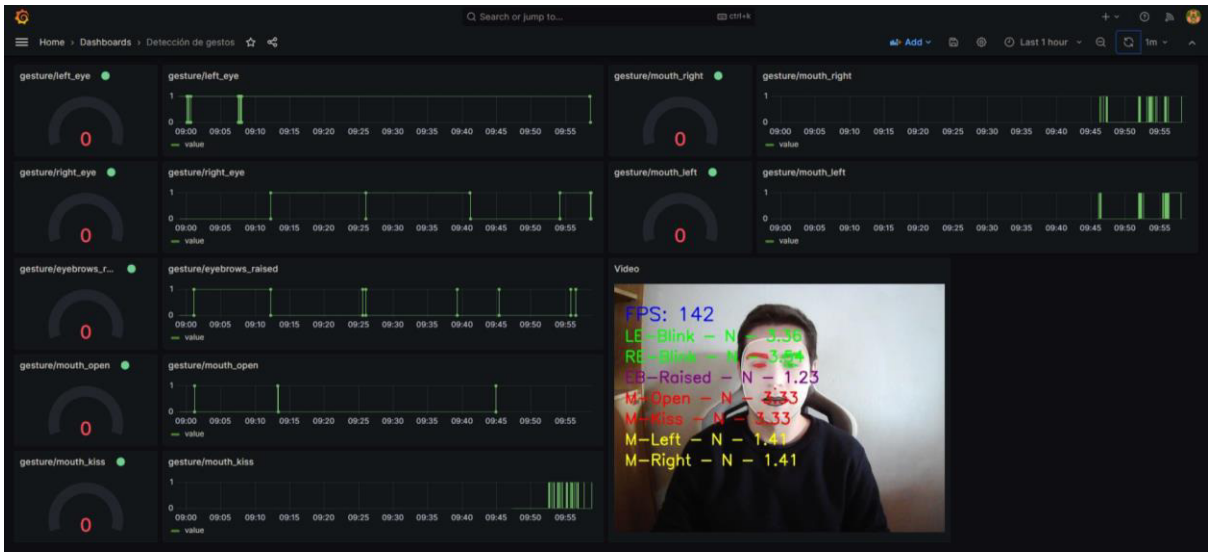

Figura 4.122. Vista del dashboard creado en Grafana.

En Grafana, un dashboard está compuesto por diversos paneles que ofrecen diferentes formas de visualización de datos.

En nuestro caso, hemos utilizado tres tipos de paneles para presentar la información de la manera más clara y comprensible. Los paneles de tipo “gauge” se emplean para mostrar la información en tiempo real de los gestos detectados, obteniendo esta información a través de la suscripción a los tópicos MQTT correspondientes a cada gesto. Cada panel “gauge” tiene

dos valores posibles: 1, cuando el gesto está activo, y 0, cuando ese gesto en particular no está siendo detectado.

Para visualizar la información histórica de los gestos detectados, hemos configurado paneles de tipo “time series”. Estos paneles nos permiten representar gráficamente en series de tiempo cuántas veces se ha detectado cada gesto durante un período determinado.

Por último, hemos incorporado un panel de tipo “text” para embeber código HTTP que muestra la página web proporcionada por la aplicación de procesamiento de imágenes. Gracias a este panel, los usuarios pueden visualizar un video en tiempo real del resultado de la detección de gestos.

5. Subsistema: Aplicación de procesamiento de imágenes

5.1. Descripción

El propósito de la aplicación de procesamiento de imágenes es dar una solución efectiva para detectar gestos en tiempo real utilizando un microcontrolador como el ESP32-CAM o una webcam. Está desarrollada en el lenguaje de programación Python y emplea la librería MediaPipe para llevar a cabo la detección de gestos a través de una malla facial.

A través de esta aplicación, los usuarios pueden interactuar con el sistema de una manera intuitiva y natural. La detección de gestos permite el control de acciones y funcionalidades mediante movimientos específicos de la cara.

El proceso de detección de gestos se realiza en tiempo real, lo que significa que la aplicación es capaz de procesar y analizar los movimientos faciales en tiempo casi instantáneo. Esto garantiza una experiencia fluida y sin demoras para el usuario. Cuando se detecta un gesto, la aplicación envía un mensaje MQTT para comunicarse con otros dispositivos o actuadores. Estos mensajes pueden desencadenar acciones específicas en los actuadores, como encender o apagar luces, abrir o cerrar una puerta, etc. La flexibilidad de MQTT permite una integración sencilla con otros dispositivos.

Por otra parte, la utilización de la librería MediaPipe es fundamental para la aplicación, ya que proporciona un conjunto de algoritmos y modelos pre entrenados para la detección y seguimiento de la cara y los gestos. MediaPipe genera una malla facial, es decir, una estructura de malla tridimensional que representa los puntos clave de la cara, como los ojos, las cejas, la nariz y la boca. Esta representación detallada permite una detección precisa y confiable de los gestos.

Finalmente, la aplicación utiliza la librería Flask para servir las imágenes procesadas en tiempo real. Flask es un framework ligero que permite crear interfaces web para visualizar y compartir las imágenes procesadas.

5.2. Herramientas utilizadas

Las herramientas que se utilizaron para el desarrollo de la aplicación son:

- **Git.** Git es un sistema de control de versiones ampliamente utilizado en el desarrollo de software. Permite gestionar el código fuente de una aplicación, realizar seguimiento de cambios y colaborar de manera eficiente.

- **GitHub.** GitHub es una plataforma online que usa Git como sistema de control de versiones. Proporciona un entorno para alojar el código fuente de nuestra aplicación.

- **Python.** Python es un lenguaje de programación de alto nivel que se emplea ampliamente para el desarrollo de aplicaciones de procesamiento de imágenes y aprendizaje automático. Es reconocido por su simplicidad y legibilidad, lo que lo hace ideal para implementar algoritmos y trabajar con librerías como Flash y MediaPipe.

- **PyCharm.** PyCharm es un entorno de desarrollo integrado (IDE) diseñado especialmente para el desarrollo de aplicaciones en el lenguaje de programación Python. Proporciona herramientas y características avanzadas para facilitar el manejo de librerías, codificación, depuración y prueba de aplicaciones.

- **Webcam.** Una webcam es una cámara de video que se conecta a la computadora para capturar imágenes y video. En este caso, la webcam se utiliza como fuente de imágenes para la detección de gestos en tiempo real.

- **ESP32-CAM.** El ESP32-CAM es un microcontrolador basado en el chip ESP32 de Espressif Systems. Tiene Wi-Fi y una cámara integrada, lo que lo hace adecuado para aplicaciones IoT y de visión por computadora. En el contexto de esta aplicación, se usa este microcontrolador como fuente de imágenes para el procesamiento y la detección de imágenes en tiempo real.

5.3. Librerías de Python utilizadas para el desarrollo

En el desarrollo de la aplicación de procesamiento de imágenes, se han utilizado varias librerías de Python que brindan funcionalidades esenciales para el procesamiento, análisis y comunicación de los datos.

Estas librerías facilitan la detección de gestos en tiempo real, el seguimiento de características faciales, la comunicación a través del protocolo MQTT y la creación de páginas web para la visualización de las imágenes procesadas.

5.3.1. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto que es ampliamente usada para el procesamiento y análisis de imágenes y videos en tiempo real.

Figura 5.1. Ejemplo de aplicación de OpenCV para convertir una imagen a escala de grises. Recuperado de: https://learnopencv.com/read-display-and-write-an-image-using-opencv/.

Fue desarrollada originalmente por Intel en 1999 y, desde entonces, se convirtió en una de las herramientas más populares en el campo de la visión por computadora y la inteligencia artificial.

La principal fortaleza de OpenCV radica en su capacidad de procesar imágenes y videos en tiempo real, haciendo extremadamente útil en una gran variedad de aplicaciones, que van desde sistemas de vigilancia hasta reconocimiento facial y robótica. Ofrece una enorme variedad de funciones y algoritmos predefinidos para realizar tareas comunes para el procesamiento de imágenes como la detección de bordes, segmentación, seguimiento de objetos y reconocimiento de patrones, entre otras.

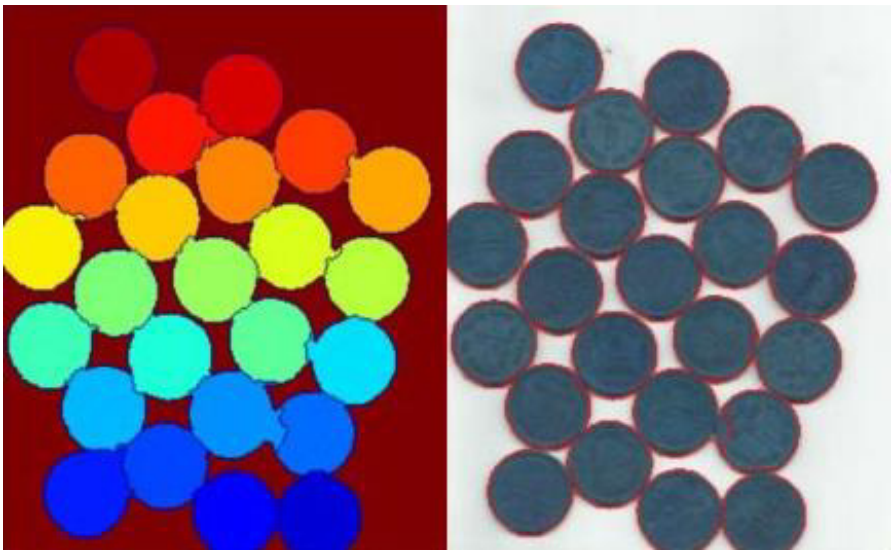

Figura 5.2. Ejemplo de aplicación de OpenCV para realizar la segmentación de un conjunto de monedas. Recuperado de: https://docs.opencv.org/3.4/d3/db4/tutorial_py_watershed.html.

OpenCV está escrito en el lenguaje de programación C++, pero cuenta con interfaces de programación (API) en varios lenguajes como Python y Java, lo que la hace altamente versátil.

La elección de OpenCV se justifica por varias razones. En primer lugar, cuenta con una gran variedad de funciones y algoritmos específicamente diseñadas para tareas de visión por computadora. Además, OpenCV es una biblioteca de código abierto con una enorme comunidad de desarrolladores. Esto significa que existe inmensa cantidad de recursos, tutoriales y ejemplos disponibles que facilitan el aprendizaje y la implementación de la solución que necesitamos.

Existen varias alternativas que también son utilizadas para el procesamiento de imágenes y visión por computadora. Algunas de ellas son:

- **Scikit-image.** Es una biblioteca de procesamiento de imágenes que ofrece una variada gama de algoritmos y herramientas para manipular imágenes. Se basa en la biblioteca NumPy y proporciona una interfaz simple de usar.
- **Mahotas.** Es una biblioteca de procesamiento de imágenes que se centra en algoritmos eficientes y rápidos para tareas de visión por computadora. Ofrece gran cantidad de funciones para operaciones básicas de procesamiento de imágenes, así como características avanzadas como detección de bordes, segmentación y descriptores de texturas.

- **SimpleCV.** Es una biblioteca de visión por computadora de alto nivel que proporciona una interfaz simple de usar para el procesamiento de imágenes. Se basa en OpenCV, pero proporciona una sintaxis más simple y abstracciones más fáciles de utilizar para facilitar el desarrollo de aplicaciones de visión por computadora.

En comparación con las alternativas mencionadas, OpenCV ofrece una mayor cantidad de funciones y algoritmos específicos para el procesamiento de imágenes y visión por computadora. Además, la documentación y tutoriales disponibles de MediaPipe están realizados con OpenCV. Scikit-image es una librería excelente, pero está más enfocada al procesamiento general de imágenes y no tiene tantas funcionalidades específicas para la detección de gestos. Mahotas, a pesar de ser eficiente y rápida, no proporciona la misma cantidad de características y algoritmos como OpenCV.

SimpleCV también es una opción interesante. No obstante, al basarse en OpenCV, puede carecer de algunas funciones y capacidades avanzadas de OpenCV.

5.3.2. MediaPipe

MediaPipe es una librería de código abierto desarrollada por Google que proporciona una serie de herramientas y funciones para el procesamiento de medios en tiempo real, incluyendo el procesamiento de imágenes, video y audio. Está diseñada para ser flexible y fácil de usar para tareas de visión por computadora y seguimiento de objetos.

Esta librería proporciona una interfaz de alto nivel que simplifica el desarrollo de aplicaciones de visión por computadora en tiempo real. Cuenta con componentes predefinidos y optimizados que se pueden ensamblar rápidamente para construir aplicaciones complejas. Permite procesar imágenes y videos para la detección de caras, seguimiento de manos, seguimiento de objetos y la estimación de pose. A través de su capacidad para crear una malla facial tridimensional, podemos con MediaPipe detectar los gestos realizados por los usuarios, facilitando la interacción con el sistema.

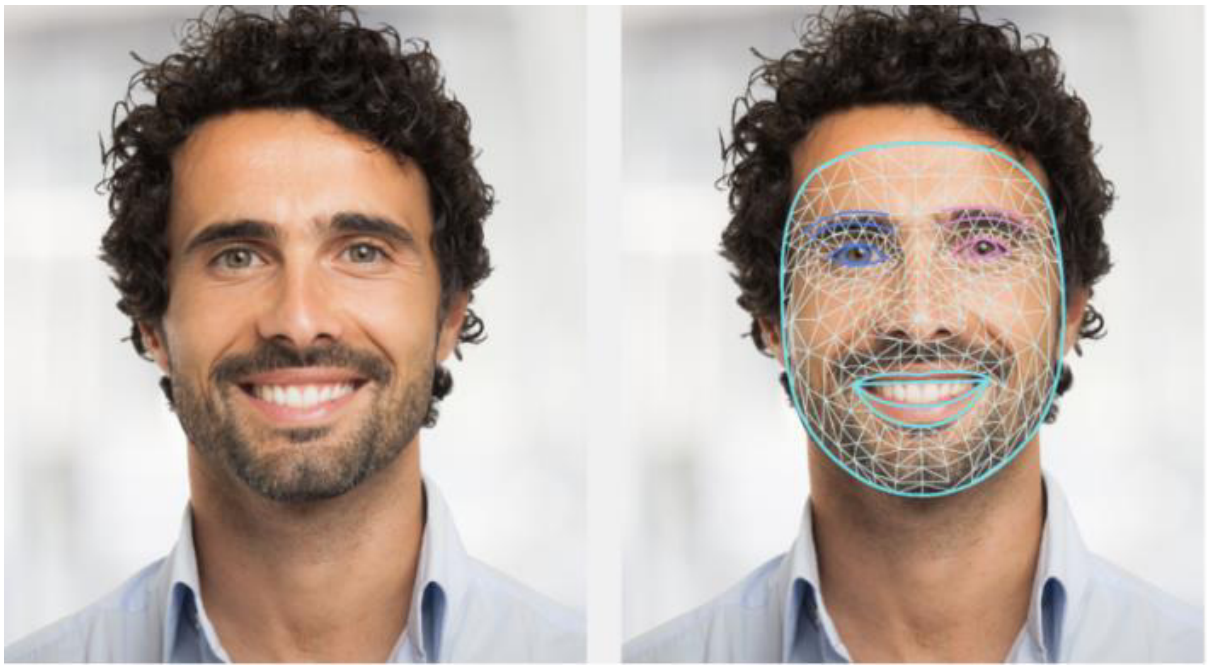

Figura 5.3. Ejemplo de malla facial obtenida con la librería MediaPipe. Recuperado de: https://developers.google.com/mediapipe/solutions/vision/face_landmarker.

Además, MediaPipe puede aprovechar la aceleración por hardware, como GPUs y unidades de procesamiento de tensorial (TPUs). Esto permite un procesamiento de imágenes y video eficiente y rápido, esencial para aplicaciones en tiempo real.

Se consideró utilizar la librería Dlib, que también está enfocada en el procesamiento de imágenes. Está escrita en C++, pero cuenta con una interfaz en Python. Si bien se destaca por su eficiencia y velocidad, ya que utiliza técnicas de optimización y aprovecha la capacidad de procesamiento de la CPU y la GPU, en las pruebas efectuadas se evidenció que esta librería es más lenta que MediaPipe.

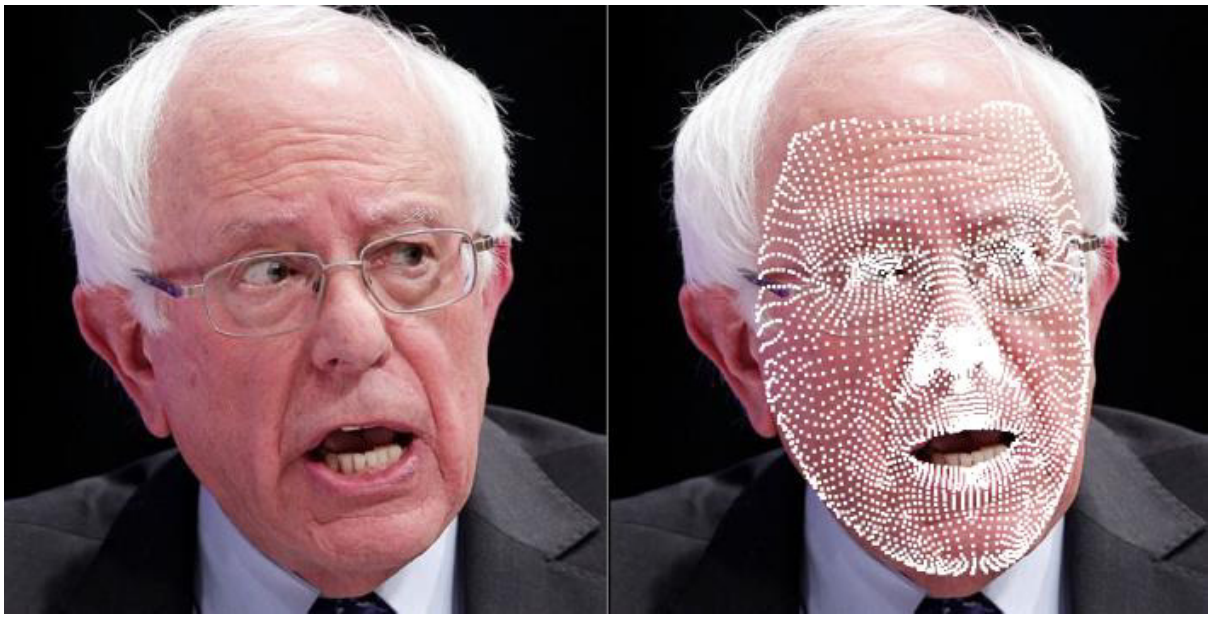

Figura 5.4. Ejemplo de malla facial obtenida con la librería Dlib. Recuperado de: https://github.com/gigadeplex/dlib-eos.

Por lo tanto, se eligió a MediaPipe como librería para el procesamiento de imágenes por su interfaz de alto nivel, su eficiencia y su capacidad de aprovechar la aceleración por hardware.

5.3.3. Paho-MQTT

Paho-MQTT es una librería de Python que implementa el protocolo MQTT, utilizado para la comunicación basada en mensajes. Esta librería permite establecer una conexión con otros dispositivos y actuadores y enviar mensajes que desencadenan acciones específicas en respuesta a los gestos detectados.

Implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT y provee de un cliente que permite que la aplicación se comunique con un bróker MQTT para publicar mensajes, suscribirse a tópicos y recibir mensajes publicados. Además, proporciona mecanismos para gestionar la conexión, como la detección y la reconexión automática en caso de desconexiones temporales.

5.3.4. Flask

Flask es un framework ligero y flexible para el lenguaje de programación Python que se utiliza principalmente para el desarrollo de aplicaciones web. Proporciona las herramientas necesarias para construir una aplicación web de manera rápida y eficiente. Se basa en el concepto de rutas

y vistas, lo que permite definir rápidamente las URL a la que los usuarios pueden acceder y lo que sucederá cuando el usuario acceda a estas rutas.

Gracias a su enfoque modular y su enorme comunidad, Flask se convirtió en una de las opciones más populares para los desarrolladores que desean construir páginas web rápidamente en Python.

En el contexto de la aplicación, Flask se utiliza para crear una interfaz web que permite a los usuarios visualizar en tiempo real las imágenes procesadas y monitorear los gestos identificados. Flask permite la creación de rutas y vistas, lo que facilita la visualización de las imágenes procesadas a partir del procesamiento de imágenes.

5.4. Funcionamiento

La aplicación de procesamiento de imágenes consta de tres partes o módulos principales, cada uno cumpliendo con una función esencial para su correcto funcionamiento:

- **Módulo de procesamiento de imágenes:** Este módulo es responsable de solicitar imágenes desde un ESP32-CAM o una webcam. Una vez recibidas, se encarga de procesarlas en busca de gestos. En caso de detectar un gesto, envía un mensaje al módulo MQTT mediante una cola compartida. Su objetivo primordial es lograr una respuesta ágil y precisa ante los gestos identificados. Además, comparte los resultados de las imágenes procesadas con el módulo web.

- **Módulo MQTT:** Está enfocado en establecer la conexión con el bróker MQTT que se encuentra en el servidor. Al recibir un nuevo mensaje a través de la cola compartida, envía un mensaje que alerta sobre la detección de un nuevo gesto. De esta manera, garantiza la comunicación efectiva entre los distintos componentes del sistema.

- **Módulo web:** Este componente se dedica a servir los “frames” o imágenes procesadas a través del protocolo HTTP. Una vez que el módulo de procesamiento de imágenes ha identificado y clasificado los gestos, estos “frames” están disponibles mediante una URL específica. Esto permite a los usuarios visualizar en tiempo real las imágenes procesadas, facilitándoles la observación de los gestos identificados y el seguimiento del rendimiento del sistema.

5.5. Implementación

5.5.1. Módulo de procesamiento de imágenes

El módulo de procesamiento de imágenes es responsable de analizar las imágenes obtenidas del ESP32-CAM y detectar gestos faciales.

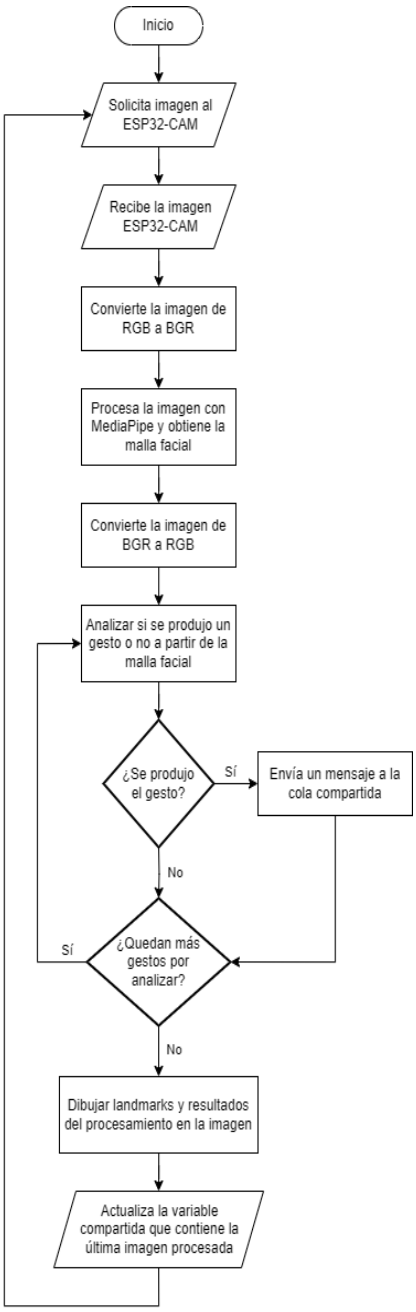

Figura 5.5. Diagrama de flujo que ilustra el comportamiento del módulo de procesamiento de imágenes, mostrando las acciones realizadas y las decisiones tomadas durante su funcionamiento.

El proceso comienza solicitando una imagen al ESP32-CAM, que luego es recibida y almacenada en la memoria de este módulo. Para llevar a cabo el análisis de los gestos faciales, la imagen es transformada del formato RGB al formato BGR utilizando la librería de procesamiento de imágenes OpenCV. Esta conversión es necesaria porque la librería MediaPipe, que se usará más adelante, requiere que las imágenes se encuentren en formato BGR para realizar el procesamiento de manera adecuada y efectiva.

Una vez que la imagen está en formato BGR, se la procesa con la librería MediaPipe. Esta etapa es crítica, ya que se obtiene la "malla facial", que consiste en una serie de puntos clave que representan ubicaciones en el rostro. Estos puntos incluyen las ubicaciones de los ojos, nariz y boca, entre otros elementos relevantes. La malla proporciona información esencial para evaluar la presencia de gestos faciales en la imagen.

Con la malla facial obtenida, la imagen se convierte nuevamente al formato RGB para que pueda ser visualizada correctamente. Esto es útil para evaluar visualmente el resultado a través del dashboard en los dispositivos de visualización.

Posterior al procesamiento, se efectúa la evaluación de los gestos faciales utilizando la información proporcionada por la malla facial. La evaluación implica cálculos y mediciones basados en relaciones proporcionales y comparativas entre los landmarks detectados. Si la relación entre las proporciones supera un umbral establecido, se considera que el gesto ha sido detectado.

Para registrar y monitorizar los gestos detectados, el módulo mantiene un diccionario actualizado de gestos. Este diccionario almacena información valiosa sobre la detección de cada gesto, incluyendo su presencia en la imagen actual y el tiempo que ha estado siendo detectado. De esta manera, se crea un seguimiento temporal de la aparición de gestos en la secuencia de imágenes.

Cuando se detecta un gesto durante un tiempo superior al umbral establecido, se envía un mensaje a una cola compartida con el módulo MQTT. El mensaje contiene el tópico y el contenido que se enviará al bróker MQTT, y contiene información sobre el gesto detectado. A continuación, se presenta una tabla que establece la correspondencia entre cada gesto y su respectivo tópico MQTT.

Gesto	Tópico MQTT
Guiño del ojo derecho	gesture/right_eye
Guiño del ojo izquierdo	gesture/left_eye
Abrir la boca	gesture/mouth_open
Beso	gesture/mouth_kiss
Levantar las cejas	gesture/eyebrows_raised
Mueca hacia la derecha	gesture/mouth_right
Mueca hacia la izquierda	gesture/mouth_left

Figura 5.6. Correspondencia entre los gestos y tópicos MQTT.

Antes de guardar la imagen procesada, se dibujan sobre ella los landmarks relevantes detectados. También se muestra la cantidad de fotogramas procesados por segundo (FPS) y las proporciones o ratios que determinan si cada gesto está siendo detectado en ese momento. Esta información es valiosa para el usuario, ya que le permite interactuar más fácilmente con el sistema y determinar si está realizando correctamente un gesto. Además, facilita el proceso de calibración del sistema.

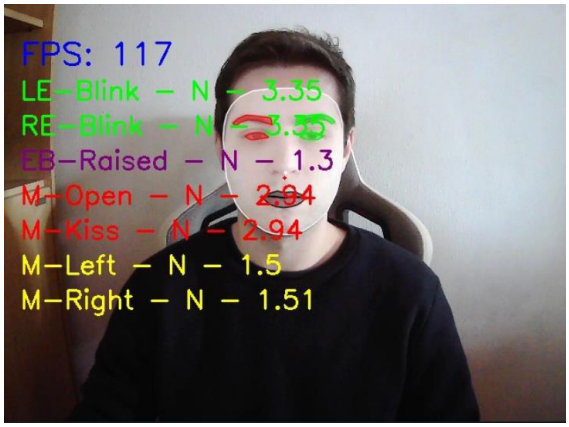

Figura 5.7. Ejemplo de información dibujada sobre la imagen procesada.

Por último, la imagen procesada se almacena en una variable compartida con el módulo web. El módulo web puede acceder a la imagen a través de esta variable y servirla a los usuarios que quieran verla a través de los dispositivos de visualización.

5.5.1.1. Gesto: Guiño de un ojo

El proceso de detección de guiños se basa en el cálculo de la relación entre la distancia horizontal y la distancia vertical de los ojos. Primero, se obtienen las coordenadas de los puntos

clave del ojo. Luego, se calcula la distancia entre los puntos de referencia que determinan el ancho del ojo, así como la distancia entre los puntos de referencia que definen su altura. Por ejemplo, para el ojo derecho:

```
horizontal_distance = euclidean_distance(
              landmarks[LandmarkPoints.RIGHT_EYE_HORIZONTAL_LEFT],

landmarks[LandmarkPoints.RIGHT_EYE_HORIZONTAL_RIGHT])

vertical_distance = euclidean_distance(
              landmarks[LandmarkPoints.RIGHT_EYE_VERTICAL_TOP],
              landmarks[LandmarkPoints.RIGHT_EYE_VERTICAL_BOTTOM])
```

Figura 5.8. Cálculo del ancho y alto del ojo.

Estos cálculos se utilizan para establecer una relación entre las distancias, es decir, el cociente entre la distancia horizontal y la distancia vertical.

```
ratio = horizontal_distance / vertical_distance
```

Figura 5.9. Cálculo de la proporción entre el ancho y el alto del ojo.

Si el valor de esta relación supera un umbral específico, establecido por el usuario, se considera que ha ocurrido un guiño del ojo, ya sea izquierdo o derecho. Tomando como ejemplo la detección para el ojo derecho:

```
right_eye = ratio > threshold
```

Figura 5.10. Condición para que el sistema determine si se realizó un guiño.

Por lo general, el umbral se establece como un valor superior a 6. Esto indica una diferencia significativa entre la distancia horizontal y la distancia vertical, lo cual es comúnmente observado durante un guiño.

Figura 5.11. Gesto de guiño con el ojo derecho.

5.5.1.2. Gesto: Abrir la boca

El gesto de apertura de la boca por parte del usuario es detectado mediante el análisis de puntos clave en la malla facial proporcionada como entrada. Para determinar si el gesto está presente en la imagen o no, se calculan algunas medidas relacionadas con la boca utilizando la distancia euclidiana entre ciertos puntos de interés.

Primero, se calcula la altura de la boca midiendo la distancia vertical entre el punto más alto de los labios superiores y el punto más bajo de los labios inferiores:

```
mouth_height = euclidean_distance(landmarks[UPPER_LIPS_TOP],
                                  landmarks[LOWER_LIPS_BOTTOM])
```

Figura 5.12. Cálculo del alto de la boca.

Luego, se calcula el ancho de la boca, que corresponde a la distancia horizontal entre el punto más a la izquierda de los labios y el punto más a la derecha de los labios:

```
mouth_width = euclidean_distance(landmarks[LIP_LEFT],
                                 landmarks[LIP_RIGHT])
```

Figura 5.13. Cálculo del ancho de la boca.

Asimismo, se mide la altura de la cavidad de la boca, que se obtiene midiendo la distancia vertical entre el punto más bajo de los labios superiores y el punto más alto de los labios inferiores:

```
mouth_cavity_height =
euclidean_distance(landmarks[UPPER_LIPS_BOTTOM],
                   landmarks[LOWER_LIPS_TOP])
```

Figura 5.14. Cálculo de la altura de la cavidad oral.

Con estas medidas, se procede a calcular dos proporciones: la relación entre el ancho y la altura de la boca (ratio_mouth) y la relación entre la altura de la cavidad de la boca y la altura del labio superior (ratio_open):

```
ratio_mouth = mouth_height/mouth_width
ratio_open = lip_height/mouth_cavity_height
```

Figura 5.15. Cálculo de las proporciones empleadas para la detección del gesto.

Finalmente, se evalúan las proporciones en relación con ciertos valores umbral para determinar si la boca está abierta. En primer lugar, se verifica si el ratio_open, la relación entre la altura de la cavidad de la boca y la altura del labio superior, es mayor que un umbral específico. Además, se examina si el ratio_mouth, que representa la relación entre el ancho y la altura de la boca, es menor que otro umbral predeterminado. Si ambas condiciones se cumplen, se registra la detección de una boca abierta.

```
mouth_open = ratio_open > threshold_1 and ratio_mouth < threshold_2
```

Figura 5.16. Condición para que la aplicación determine que se realizó el gesto de abrir la boca.

Este resultado se debe a que, cuando la boca está abierta, la proporción entre la altura de la cavidad bucal y la altura del labio superior (ratio_open) tiende a aumentar, mientras que la relación entre el ancho y la altura de la boca (ratio_mouth) tiende a disminuir. Estos cambios en las proporciones son indicativos de una boca abierta en la imagen analizada.

Figura 5.17. Gesto de abrir la boca.

5.5.1.3. Gesto: Beso

Cuando se realiza la detección del gesto del beso, se toman en cuenta las mismas proporciones que se utilizan para detectar el gesto de abrir la boca, pero con ajustes específicos para adaptarse a las características únicas del beso.

Durante un beso, la forma de la boca se vuelve más simétrica y equilibrada, lo que significa que la relación entre el alto y el ancho de la boca estará más cercana a uno. Esto implica que el umbral empleado para la detección del gesto del beso, denominado "umbral_1", se ajustará y estará más cercano a uno, de manera que pueda capturar los valores que representan una apertura casi cuadrada de la boca, característica típica del beso.

Además, durante un beso, los labios están unidos, lo que da lugar a que la altura del labio superior sea mayor que la altura de la cavidad bucal. La altura de la cavidad bucal se acerca a cero, ya que está prácticamente cerrada debido a que los labios están unidos durante el beso.

```
kiss = ratio_mouth < threshold_1 and ratio_kiss > threshold_2
```

Figura 5.18. Condición para que la aplicación determine que se realizó el gesto de beso.

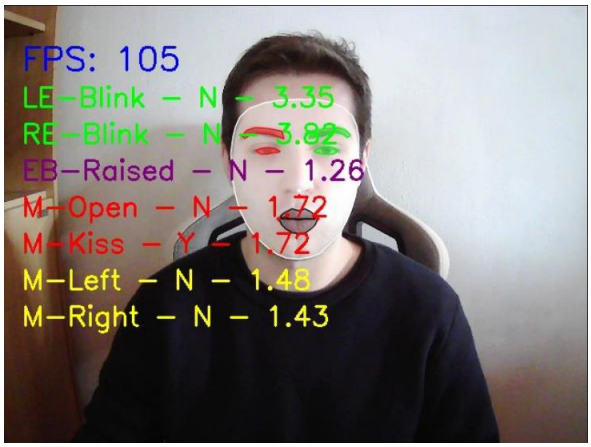

Figura 5.19. Gesto de beso.

5.5.1.4. Gesto: Levantar las cejas

La detección del gesto de levantar las cejas se basa en el cálculo de cuatro distancias: dos desde la parte superior del ojo hasta la parte superior de cada ceja (izquierda y derecha), y otras dos desde la parte superior de la cara hasta la parte superior de cada ceja. Estas distancias se obtienen mediante la fórmula de distancia euclidiana, que calcula la longitud entre dos puntos en un plano cartesiano.

Posteriormente, se calculan dos relaciones (ratios) para cada ceja. Estas relaciones miden la proporción entre la distancia de la parte superior del rostro a la parte superior de la ceja y la distancia de la parte superior del ojo a la parte superior de la ceja. En esencia, evalúan qué tan alejadas están las cejas tanto de la parte superior del rostro como de la parte superior del ojo.

```
left_eyebrow_to_eye_distance = euclidean_distance(
          landmarks[LandmarkPoints.LEFT_EYE_VERTICAL_TOP],
          landmarks[LandmarkPoints.LEFT_EYEBROW_UPPER_MIDPOINT])
right_eyebrow_to_eye_distance = euclidean_distance(
          landmarks[LandmarkPoints.RIGHT_EYE_VERTICAL_TOP],
          landmarks[LandmarkPoints.RIGHT_EYEBROW_UPPER_MIDPOINT])

left_eyebrow_to_face_top_distance  = euclidean_distance(
          landmarks[LandmarkPoints.FACE_TOP_LEFT_EYEBROW],
          landmarks[LandmarkPoints.LEFT_EYEBROW_UPPER_MIDPOINT])
right_eyebrow_to_face_top_distance = euclidean_distance(
          landmarks[LandmarkPoints.FACE_TOP_RIGHT_EYEBROW],
          landmarks[LandmarkPoints.RIGHT_EYEBROW_UPPER_MIDPOINT])

ratio_left = left_eyebrow_to_face_top_distance /
             left_eyebrow_to_eye_distance
ratio_right = right_eyebrow_to_face_top_distance /
              right_eyebrow_to_eye_distance
```

Figura 5.20. Cálculo de las distancias y proporciones empleadas para detectar el gesto de levantar las cejas.

El siguiente paso es determinar cuál de estas dos relaciones (izquierda o derecha) es la más grande y se selecciona esa como la relación principal.

```
ratio = max(ratio_left, ratio_right)
```

Figura 5.21. Obtención del mayor de las dos proporciones anteriormente calculadas.

Finalmente, si la relación principal es menor que un umbral predefinido, se interpreta que las cejas se han levantado, ya que la distancia de la parte superior del rostro a la parte superior de la ceja es menor que la distancia de la parte superior del ojo a la parte superior de la ceja, lo que indica una posición más elevada de las cejas respecto a su estado normal.

```
eyebrows_raised = ratio < threshhold
```

Figura 5.22. Condición para que la aplicación determine que se realizó el gesto de levantar las cejas.

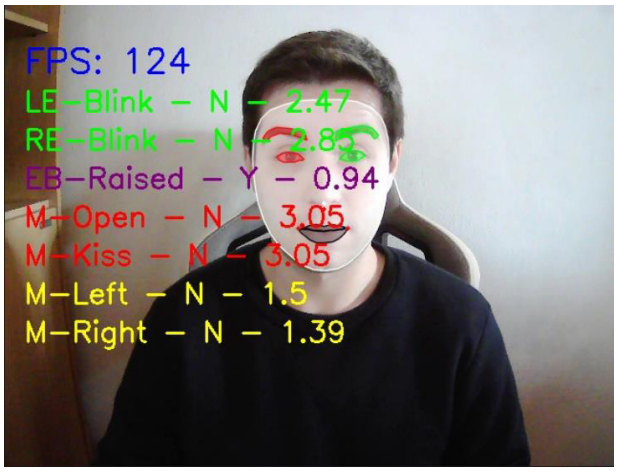

Figura 5.23. Gesto de levantar las cejas.

5.5.1.5. Gesto: Mueca hacia los lados

La detección del gesto de la mueca con la boca hacia un lado se realiza a través de una función que calcula una relación (ratio) entre dos distancias en el rostro detectado por un sistema de reconocimiento facial. La detección del gesto se puede hacer hacia ambos lados, el derecho y el izquierdo.

El análisis comienza midiendo dos distancias mediante la fórmula de distancia euclidiana. La primera distancia es entre el punto superior del labio y la parte inferior de la nariz, lo que mide la altura vertical del labio en relación con la nariz. La segunda distancia es entre el punto superior del labio y el punto central del labio en el lado correspondiente, lo que mide la distancia horizontal desde el centro del labio hasta el lado correspondiente. Por ejemplo, para la mueca hacia el lado izquierdo:

```
mouth_top_to_nose = euclidean_distance(
                    landmarks[LandmarkPoints.UPPER_LIPS_TOP],
                    landmarks[LandmarkPoints.NOSE_UNDER_POINT])

mouth_side_to_middle = euclidean_distance(
                       landmarks[LandmarkPoints.UPPER_LIPS_TOP],
                       landmarks[LandmarkPoints.LIP_RIGHT])
```

Figura 5.24. Cálculo de las distancias utilizadas para detectar el gesto de mueca hacia los lados.

Luego, se obtiene la relación (ratio) dividiendo la distancia desde el punto superior del labio hasta la parte inferior de la nariz entre la distancia desde el punto superior del labio hasta el

punto central del labio en el lado correspondiente. Esta relación indica qué tan alta es la parte superior del labio en comparación con su posición lateral.

```
ratio = mouth_top_to_nose / mouth_side_to_middle
```

Figura 5.25. Cálculo de la proporción utilizada para detectar el gesto de mueca hacia los lados.

Si la relación calculada es menor que este umbral y el gesto de la boca abierta no ha sido detectado (lo cual se verifica con una condición adicional), entonces se considera que se ha detectado el gesto de la mueca hacia el lado especificado (derecho o izquierdo).

```
mouth_side = ratio < threshold and not mouth_open
```

Figura 5.26. Condición para que la aplicación determine que se realizó el gesto de mueca hacia los lados.

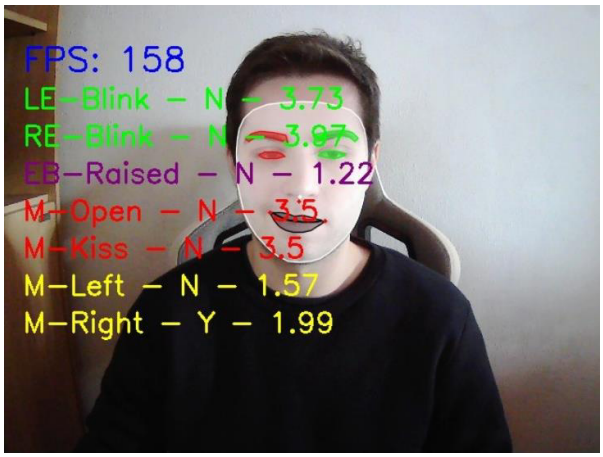

Figura 5.27. Gesto de mueca hacia el lado derecho.

5.5.2. Módulo MQTT

El objetivo principal de este módulo es recibir mensajes del módulo de procesamiento de imágenes, transformarlos en mensajes MQTT y enviarlos al bróker MQTT para que los clientes o dispositivos suscritos puedan recibir y procesar la información.

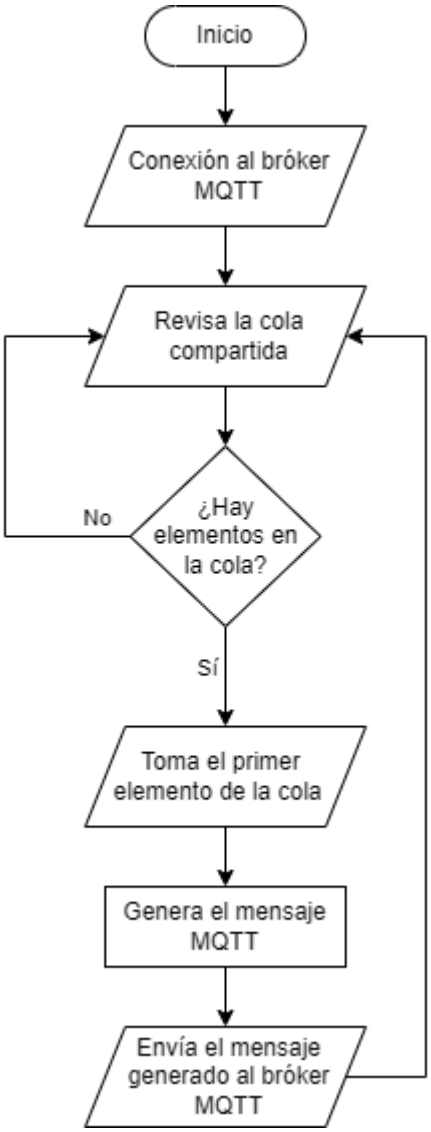

Figura 5.28. Diagrama de flujo que ilustra el comportamiento del módulo MQTT, mostrando las acciones realizadas y las decisiones tomadas durante su funcionamiento.

El funcionamiento del módulo inicia con el establecimiento de la conexión al bróker MQTT utilizando la librería Paho-MQTT, que facilita la conexión con el bróker MQTT. Esta conexión permite que este módulo se comunique de manera bidireccional con el bróker, aunque solo se empleará la conexión para el envío de mensajes.

Una vez conectado al bróker, el módulo comprueba la cola compartida para verificar si existen mensajes pendientes de procesamiento. La cola es una estructura de tipo FIFO (del inglés *first in, first out*), lo que significa que los mensajes que ingresaron primero a la cola serán los primeros en ser procesados. En caso de que existan varios mensajes en espera, el módulo toma el primer mensaje disponible de la cola para su procesamiento.

Si la cola está vacía, lo que significa que no hay mensajes en espera, el módulo entra en un estado de espera activa, donde permanecerá atento a nuevos mensajes que se añadan a la cola.

Luego de que el módulo ha tomado un mensaje de la cola compartida, procede a generar un mensaje MQTT basado en el contenido del mensaje obtenido. Es decir, toma el objeto enviado por el módulo de procesamiento de imágenes y lo transforma en un tópico y mensaje MQTT.

El mensaje generado lo envía al bróker MQTT para que sea distribuido a los clientes o dispositivos suscritos que estén interesados en este tipo de información. Como hemos mencionado anteriormente, el bróker MQTT es el intermediario entre los productores de mensajes (en este caso, la aplicación de procesamiento de imágenes) y los consumidores de mensajes (los clientes o dispositivos suscritos).

Este proceso de toma, generación y envío de mensajes MQTT se ejecuta en un bucle continuo mientras existan elementos en la cola compartida. En otras palabras, el módulo seguirá procesando mensajes hasta que la cola quede temporalmente vacía. En ese momento, el módulo entra nuevamente en el estado de espera activa, donde permanecerá hasta que lleguen nuevos mensajes a la cola para su procesamiento.

5.5.3. Módulo web

El propósito de este módulo es servir las imágenes resultantes del proceso realizado en el módulo de procesamiento de imágenes, utilizando la librería Flask.

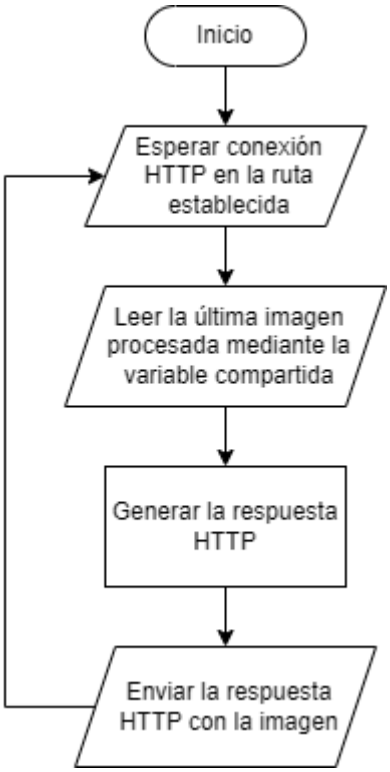

Figura 5.29. Diagrama de flujo que ilustra el comportamiento del módulo web, mostrando las acciones realizadas y las decisiones tomadas durante su funcionamiento.

Este módulo se configura para esperar y manejar las conexiones HTTP en una ruta específica, desde el cual las imágenes se servirán. En otras palabras, cuando un cliente hace una solicitud HTTP a esta ruta, el módulo web entra en acción y responde con la imagen más reciente que ha sido procesada.

Para lograr esto, el módulo web establece una comunicación con el módulo de procesamiento de imágenes mediante una variable compartida. De esta manera, el módulo web puede acceder y leer la última imagen procesada, la cual se encuentra almacenada en dicha variable.

Una vez que se obtiene la imagen, el módulo web genera una respuesta HTTP con la misma. La particularidad de esta respuesta es que utiliza el MIME Type "*multipart/x-mixed-replace; boundary=frame*". Este MIME Type permite enviar múltiples partes de datos en una única respuesta HTTP, y al mismo tiempo, ir reemplazando su contenido a medida que se generan nuevas imágenes. En otras palabras, el cliente recibe una corriente de datos de imágenes en constante actualización, como si fuera una secuencia de fotogramas o un video.

Esta técnica de envío de datos mediante "*multipart/x-mixed-replace*" es útil en aplicaciones donde se necesita una actualización continua de información como, en este caso, el flujo de imágenes procesadas. Además, esta técnica evita la necesidad de realizar múltiples solicitudes HTTP repetidas al servidor para obtener las imágenes más recientes, lo que ahorra ancho de banda y mejora la eficiencia de la comunicación.

6. Subsistema: Microcontroladores

6.1. Descripción

El subsistema de microcontroladores utiliza dos tipos de dispositivos, el ESP32 y el ESP32-CAM para establecer una red interna a través del protocolo ESP-NOW y permitir la conexión entre el servidor y los microcontroladores.

El ESP32-CAM se encarga de proporcionar las imágenes necesarias para el procesamiento de imágenes y la detección de gestos en el servidor. También cumple la función de gateway, actuando como un enlace entre la red interna e Internet.

Por otra parte, los ESP32 son los responsables de ejecutar acciones basadas en los comandos recibidos del servidor a través del ESP32-CAM. Estos mensajes de control le comunican a los ESP32 qué acciones deben efectuar. Los actuadores conectados al ESP32, como luces, relés u otros dispositivos, se activan en respuesta a estos comandos.

6.2. Dispositivos utilizados

A continuación, se presenta un cuadro comparativo que destaca las características principales del ESP8266, ESP32, ESP32-CAM, Arduino Uno y Raspberry Pi 4 Model B.

Todos estos dispositivos son frecuentemente utilizados en proyectos de electrónica, IoT y computación, contando cada uno con sus propias fortalezas y aplicaciones específicas. La comparación se centra en aspectos claves para este proyecto como el tipo de dispositivo, procesador, velocidad de reloj, memoria, conectividad, capacidad de procesamiento y uso principal.

Característica	ESP8266	ESP32	ESP32-CAM	Arduino Uno	Raspberry Pi 4 Model B
Tipo	Microcontrol a- dor	Microcontrola - dor	Microcontrola - dor con cámara	Microcontrola - dor	Computadora de placa única
Arquitectura	32 bits	32 bits	32 bits	8 bits	64 bits ARM v8
Núcleos del procesador	1	2	2	1	4
Velocidad de reloj	80 MHz	160 o 240 MHz	160 o 240 MHz	16 MHz	1,5 GHz
Memoria RAM	160 KB	Hasta 520 KB	Hasta 520 KB	2 KB	1 GB, 2 GB o 4 GB (dependiendo del modelo)
Memoria Flash	4 MB	Hasta 16 MB	Hasta 16 MB	32 KB	-
Conectividad inalámbrica	Wi-Fi	Wi-Fi, Bluetooth, BLE	Wi-Fi, Bluetooth, BLE	No integrada (se pueden añadir módulos)	Wi-Fi, Bluetooth, BLE (dependiendo del modelo)
Cámara integrada	No	No	Sí, cámara OV2640 de 2 MP	No	No (se pueden conectar cámaras externas)
Sistema operativo	No	No	No	No	Linux (varias distribuciones disponibles)
Puertos de entrada/salida (I/O)	Limitados (dependiendo de la placa)	Amplia variedad de pines I/O	Amplia variedad de pines I/O	Amplia variedad de pines I/O	Amplia variedad de pines I/O
Uso principal	Proyectos IoT, aplicaciones de bajo consumo	Proyectos IoT, aplicaciones de mayor capacidad de procesamiento	Proyectos IoT, vigilancia, monitoreo, captura de imágenes	Prototipado electrónico, proyectos simples	Proyectos IoT, aplicaciones que requieren de capacidades informáticas avanzadas
Costo	Bajo	Medio	Medio	Bajo	Alto (depende del modelo)

Figura 6.1. Comparativa entre los dispositivos considerados.

Los dispositivos que se eligieron son el ESP32 y el ESP32-CAM. Su elección se debe a las características y funcionalidades clave que ofrecen, las cuales son fundamentales para el funcionamiento del sistema.

Primeramente, el ESP32 es un microcontrolador de alto rendimiento que cuenta con dos núcleos de procesador, lo que le permite realizar tareas de manera eficiente y concurrente. A comparación del ESP8266 y el Arduino Uno, ofrece una mayor capacidad de memoria y una velocidad de reloj de hasta 240 MHz, lo que garantiza una ejecución fluida de instrucciones y una gestión eficiente de los recursos del sistema.

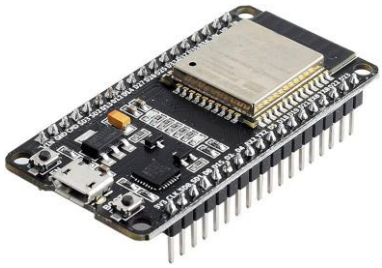

Figura 6.2. Imagen ilustrativa de un ESP32. Recuperado de: https://www.amazon.com/-/es/DIYmall-ESP32-WROOM-32-desarrollo-ESP-32S-Arduino/dp/B084KWNMM4.

En cuanto al ESP32-CAM, su característica más destacada es su cámara integrada OV2640 de 2 megapíxeles. Esto nos otorga la capacidad de capturar imágenes, lo cual es esencial para nuestro sistema. Las imágenes son enviadas al servidor para ejecutar el procesamiento de imágenes y la detección de gestos.

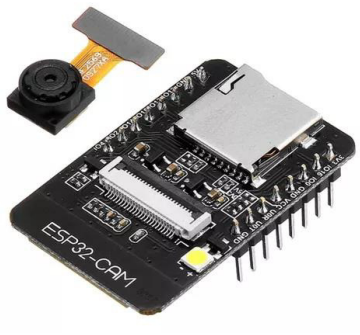

Figura 6.3. Imagen ilustrativa de un ESP32-CAM. Recuperado de: https://www.amazon.com/-/es/Taidacent-ESP32-CAM-OV2640-Esp32-desarrollo/dp/B07YL13RHP.

Adicionalmente, el ESP32 y el ESP32-CAM ofrecen varios tipos de conexión inalámbrica, como Wi-Fi y Bluetooth, permitiendo comunicarnos con otros dispositivos y sistemas.

Tanto el ESP32 como el ESP32-CAM son compatibles con el protocolo ESP-NOW, lo que permite que se comuniquen entre sí de manera inalámbrica. De esta manera, podemos crear una red interna en la que el ESP32-CAM actúa como gateway entre la red interna, formada por los ESP32 y el ESP32-CAM, y la red externa para la comunicación con el servidor.

Finalmente, ambos son más fáciles de adquirir y cuentan con un precio más accesible que los Raspberry Pi 4 Model B, convirtiéndolos en la opción ideal entre las alternativas consideradas para este subsistema.

6.3. Comunicación entre los microcontroladores

6.3.1. Protocolo ESP-NOW

La comunicación entre los ESP32 y el ESP32-CAM se realiza a través del protocolo ESP-NOW. El protocolo ESP-NOW es una tecnología de comunicación inalámbrica desarrollada por Espressif Systems específicamente para los dispositivos basados en el chip ESP8266 y ESP32. Brinda una forma eficiente y de bajo consumo de transmitir datos entre estos dispositivos de forma directa, sin necesidad de un enrutador o punto de acceso Wi-Fi.

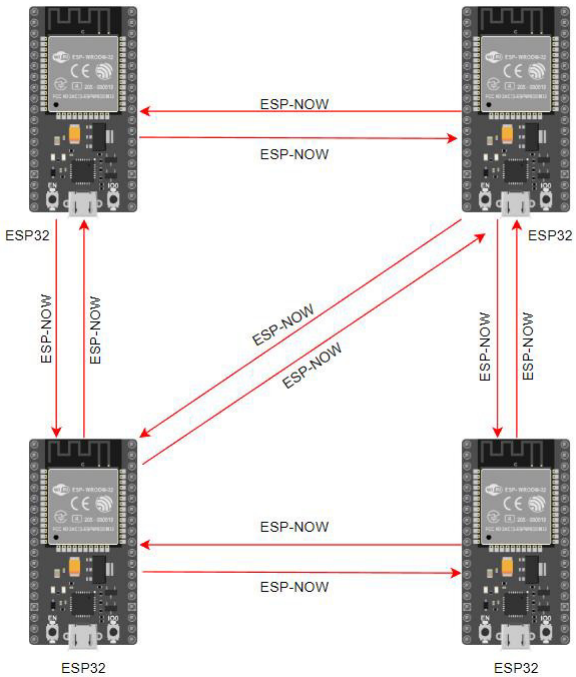

Figura 6.4. Ejemplo de esquema de comunicación inalámbrica entre ESP32 utilizando el protocolo ESP-NOW.

El funcionamiento de este protocolo se basa en un esquema de comunicación de dos pasos: el registro y el envío de datos. En primer lugar, se realiza el registro de los dispositivos participantes de la red. Cada dispositivo se registra con un identificador único y se establece una clave de encriptación compartida para garantizar la seguridad de la información.

Una vez que los dispositivos están registrados, pueden intercambiar datos entre ellos de manera eficiente. El protocolo utiliza el direccionamiento MAC para el envío de los paquetes de datos. Cada dispositivo cuenta con una dirección MAC única, que se utiliza como identificador en la red. Entonces, el dispositivo emisor codifica los datos en un paquete y lo envía a la dirección MAC del dispositivo receptor.

Es importante destacar que el envío de datos es unidireccional y sin confirmación de entrega. Esto quiere decir que el dispositivo emisor envía los datos al receptor, pero no espera una confirmación de que los datos se hayan recibido correctamente. Esta característica permite una comunicación rápida, ideal para aplicaciones de tiempo real o baja latencia, pero no asegura que los datos enviados sean recibidos correctamente.

Por último, una característica destacada del protocolo ESP-NOW es la capacidad de establecer una red de dispositivos en malla, que permite que los dispositivos se comuniquen directamente entre sí, sin necesidad de pasar por un punto centralizado. Esto facilita la creación de redes de dispositivos distribuidos, como la implementación de un sistema de actuadores en un edificio o en un hogar.

6.3.2. Aplicación de ESP-NOW en el sistema

El protocolo ESP-NOW es ideal para aplicaciones IoT que requieren de una comunicación rápida y eficiente de dispositivos, como es el caso de este subsistema.

El subsistema de microcontroladores utiliza el protocolo ESP-NOW para establecer una red interna y permitir la comunicación entre los dispositivos. Tenemos un dispositivo central, el ESP32-CAM, que actúa como gateway o puerta de enlace entre la red interna ESP-NOW y la conexión a Internet.

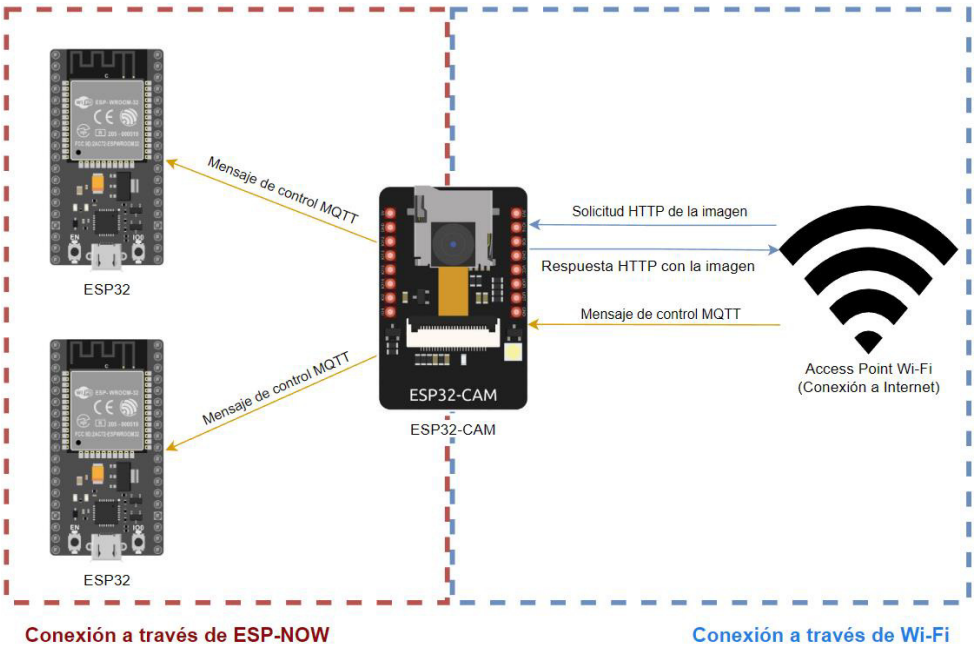

Figura 6.5. Esquema que ilustra los mensajes recibidos y enviados por el ESP32-CAM.

Por otra parte, también tenemos varios ESP32 adicionales que se conectan al ESP32-CAM a través del protocolo ESP-NOW. Estos ESP32 están equipados con actuadores, por lo que son capaces de ejecutar acciones físicas como encender o apagar luces, activar motores o abrir puertas, entre otras posibilidades.

La principal ventaja del protocolo ESP-NOW para este subsistema es su capacidad para formar una malla (mesh) entre los dispositivos conectados. Esto quiere decir que cada ESP32 puede actuar como un repetidor de señal, extendiendo el alcance de la red y asegurando una cobertura más amplia en comparación con una simple conexión punto a punto a través de un punto de acceso Wi-Fi. Si un dispositivo está fuera del alcance directo del ESP32-CAM, puede enviar sus datos o comandos a través de otros dispositivos de red hasta que lleguen al destino final.

El ESP32-CAM, al ser el gateway, recibe los comandos de control a través de Internet y luego los transmite a los ESP32 conectados a través del protocolo ESP-NOW. Esto permite controlar los actuadores de los dispositivos cuando se efectúa la detección de gestos en el servidor.

6.4. Conexión a la red externa

El ESP32-CAM se conecta a Internet a través de una red Wi-Fi doméstica. Esto quiere decir que utiliza la red existente en el hogar para establecer una conexión a Internet.

El ESP32-CAM ya cuenta con un módulo Wi-Fi integrado, de manera que puede conectarse a las redes Wi-Fi disponibles. Sin embargo, es necesario configurarlo para conectarse a la red Wi-Fi que deseemos.

El proceso de configuración del ESP32-CAM para conectarlo a una red Wi-Fi generalmente implica configurar los parámetros necesarios, como el nombre de la red (SSID) y la contraseña de la red. Estos datos son propios de la red Wi-Fi y deben ser ingresados en el código fuente que se carga en el dispositivo.

Una vez que se cargaron los datos de autenticación de la red Wi-Fi y se prende el dispositivo, el ESP32-CAM intentará establecer una conexión con el punto de acceso Wi-Fi de la red doméstica. Utiliza el protocolo estándar de Wi-Fi (por ejemplo, 802.11 b/g/n) para establecer una conexión segura y cifrada.

Luego de conectarse a la red Wi-Fi doméstica, el ESP32-CAM puede aprovechar esa conexión para acceder a Internet. Así, puede enviar y recibir datos a través de la red Wi-Fi desde y hacia servicios en la nube, como el servidor que empleamos para esta aplicación.

6.5. Implementación

6.5.1. Herramientas utilizadas

Las herramientas utilizadas para el desarrollo de los programas que se ejecutan dentro de los microcontroladores incluyen:

- **Git.** Es un sistema de control de versiones ampliamente utilizado en el desarrollo de software. Facilita la administración del código fuente de una aplicación, realiza un seguimiento de los cambios y permite una colaboración eficiente entre los miembros de un equipo.
- **GitHub.** Es una plataforma en línea que se basa en el sistema de control de versiones Git. Ofrece un entorno donde es posible hospedar el código fuente de una aplicación y brinda diversas herramientas para facilitar el trabajo en equipo.
- **Arduino IDE.** Se emplea como el entorno de desarrollo integrado para programar los microcontroladores de la plataforma Arduino y similares. Con esta herramienta, se

pueden escribir, compilar y cargar programas en los microcontroladores para su funcionamiento.

6.5.2. Librerías utilizadas

Las librerías que se emplearon para simplificar el desarrollo de los programas que están corriendo en los microcontroladores son:

- **esp32cam.** Proporciona una API orientada a objetos para usar la cámara OV2640 en el microcontrolador ESP32-CAM. Abstrae la librería "esp32-camera" y está diseñada para funcionar con la placa AI Thinker ESP32-CAM y la cámara OV2640. Con esta librería, se puede acceder y controlar fácilmente la cámara del ESP32, permitiendo la captura de imágenes y el procesamiento de video.

- **WebServer.** Brinda una interfaz que facilita la creación y gestión de un servidor web en el microcontrolador ESP. Con esta librería, es posible recibir y responder a solicitudes HTTP, lo que resulta especialmente útil para construir interfaces web para interactuar con el microcontrolador

- **WiFi.** Proporciona funciones para configurar y administrar la conectividad Wi-Fi del microcontrolador ESP. Permite conectar el dispositivo a redes Wi-Fi existentes y configurar su modo de operación como estación o punto de acceso. Con esta librería, el microcontrolador puede conectarse a la red local o a Internet, lo que abre posibilidades de comunicación y control remoto.

- **esp_now.** Facilita la comunicación directa y rápida entre dos o más microcontroladores ESP, sin la necesidad de un punto de acceso Wi-Fi, a través del protocolo ESP-NOW. Permite el envío y recepción de datos de manera eficiente, lo que es especialmente útil para aplicaciones de Internet de las cosas (IoT) donde se requiere una comunicación directa entre dispositivos cercanos.

- **pubsubclient.** Se utiliza para implementar el protocolo MQTT en el microcontrolador ESP. A partir de esta librería, el microcontrolador puede publicar y suscribirse a tópicos, lo que es fundamental para el intercambio de información en tiempo real en aplicaciones de IoT.

6.5.3. ESP32-CAM

El ESP32-CAM tiene la función principal de capturar imágenes y proporcionar acceso a estas a través de un servidor web. Este dispositivo crea un servidor web que responde a solicitudes de imágenes en diferentes calidades: baja, media y alta. Para lograr esto, se generan tres rutas o endpoints específicos, cada uno asociado a una calidad de imagen diferente:

- **"/cam-lo.jpg"** para imágenes de baja calidad (320x240 píxeles).
- **"/cam-mi.jpg"** para imágenes de calidad media (530x350 píxeles).
- **"/cam-hi.jpg"** para imágenes de alta calidad (800x600 píxeles).

Cada una de estas rutas utiliza el método HTTP GET. Cuando un cliente accede a una de estas URL mediante un navegador o cualquier cliente HTTP, envía una solicitud GET al servidor web del ESP32-CAM para obtener la imagen con la calidad deseada.

El servidor web, al recibir la solicitud, pasa el control al manejador (handler) correspondiente según la ruta solicitada. Por ejemplo, si se accede a "/cam-lo.jpg", se ejecuta el manejador "handleJpgLo" que realiza la captura de una imagen de baja calidad. Lo mismo sucede para las rutas "/cam-mi.jpg" y "/cam-hi.jpg", donde se ejecutan los handlers "handleJpgMi" y "handleJpgHi", respectivamente, para capturar imágenes de calidad media y alta.

```
127   // Add callback for each URL
128   server.on("/cam-lo.jpg", handleJpgLo);
129   server.on("/cam-mi.jpg", handleJpgMi);
130   server.on("/cam-hi.jpg", handleJpgHi);
```

Figura 6.6. Configuración de los handlers para cada ruta del servidor web del ESP32-CAM.

Simultáneamente, el ESP32-CAM se mantiene a la escucha de mensajes MQTT provenientes del tópico "gesture/#". En el contexto de MQTT, el símbolo "#" es conocido como el comodín de nivel múltiple y representa todos los subniveles bajo la jerarquía especificada. En otras palabras, permite suscribirse a todos los tópicos que comiencen con "gesture/", independientemente de los niveles adicionales que puedan seguir a continuación.

Además de su función de servidor web, el ESP32-CAM desempeña el papel de gateway o puerta de enlace entre la red interna ESP-NOW y la conexión a Internet. Cuando recibe mensajes MQTT a través de la conexión a Internet, se encarga de re-enviarlos a otros

dispositivos ESP32 conectados a través de la red interna ESP-NOW. De esta manera, se asegura de que los ESP32 reciban los mensajes y ejecuten las acciones correspondientes según la detección de cada gesto, manteniendo una comunicación efectiva entre los distintos dispositivos.

6.5.4. ESP32

Los módulos ESP32 están equipados con actuadores que responden a los mensajes de control recibidos. Para la implementación y las pruebas, se ha optado por utilizar luces LED como actuadores, ya que ofrecen una manera clara y visualmente perceptible de indicar la ejecución de acciones específicas.

En el primer ESP32, se han conectado cuatro luces LED, dispuestas de acuerdo al esquema proporcionado. Este módulo responderá a los mensajes de control enviados a los tópicos: “gesture/mouth_open”, “gesture/mouth_kiss”, “gesture/mouth_left” y “gesture/mouth_right”.

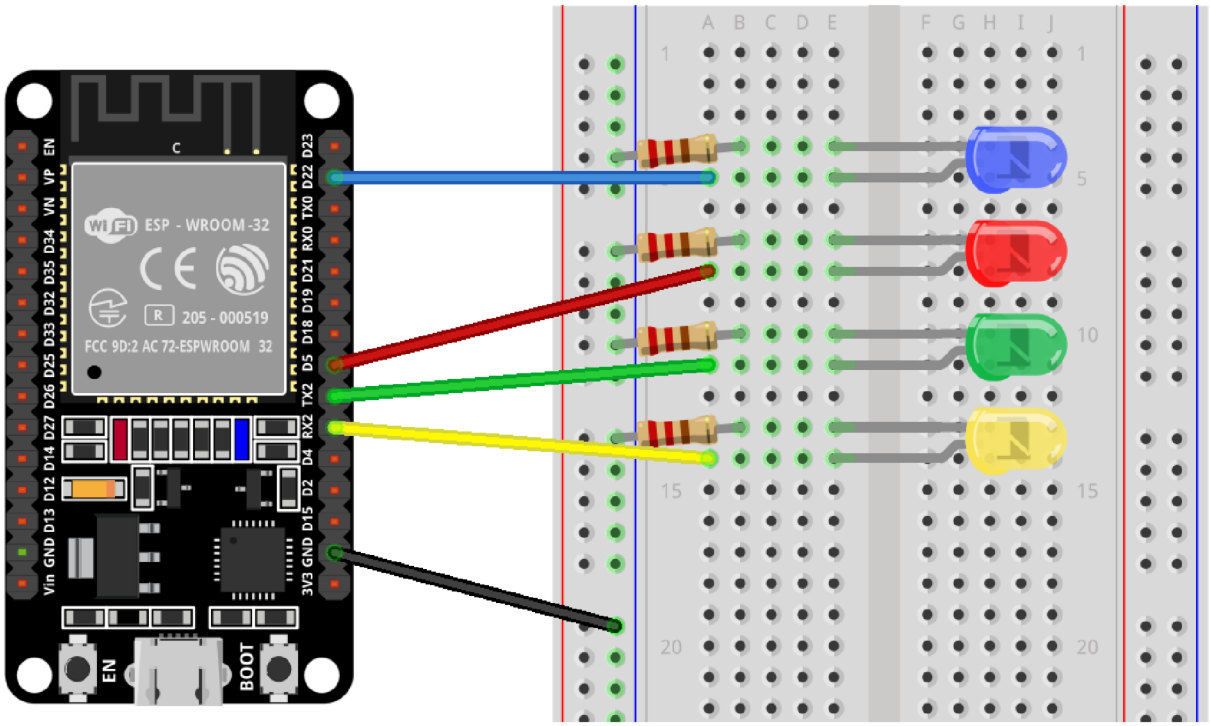

Figura 6.7. Esquema de conexionado del primer ESP32.

Por otro lado, el segundo ESP32 cuenta con tres luces LED, distribuidas de acuerdo con el segundo esquema. Este módulo procesa los mensajes de control recibidos en los siguientes tópicos: “gesture/eyebrows_raised”, “gesture/left_eye” y “gesture/right_eye”.

La aplicación de procesamiento de imágenes tiene la capacidad de detectar un total de siete gestos diferentes, y cada uno de estos gestos se asigna a uno de los tópicos mencionados anteriormente. De esta manera, el procesamiento de los gestos se divide entre los dos ESP32.

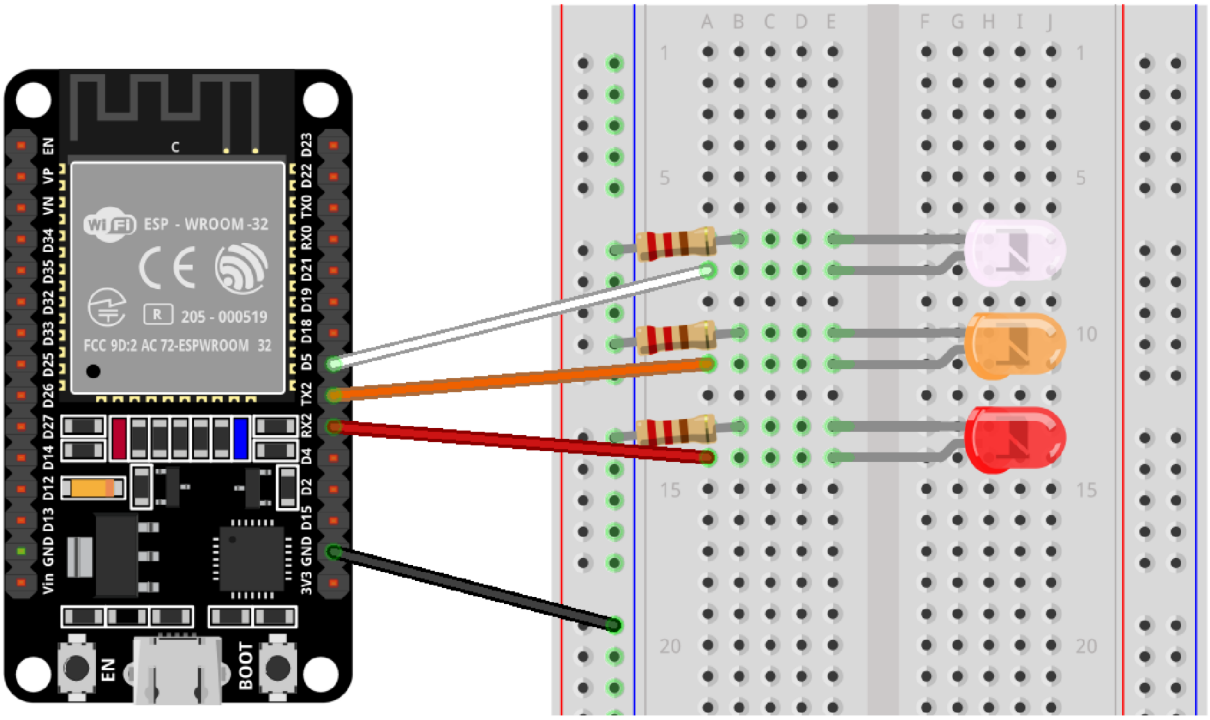

Figura 6.8. Esquema de conexionado del segundo ESP32.

La comunicación entre el ESP32-CAM y los ESP32 se realiza mediante el protocolo ESP-NOW, lo que permite una transmisión eficiente de los mensajes de control. Estos mensajes son interpretados por los ESP32 y se traducen en acciones concretas llevadas a cabo a través de los actuadores.

La configuración mencionada abre la posibilidad de aplicaciones futuras mucho más amplias, especialmente en el ámbito del control y la automatización del hogar. Por ejemplo, se podrían emplear estos dispositivos para la apertura de puertas, el control de la iluminación, la gestión de la calefacción o aire acondicionado, el control de persianas y cortinas, e incluso el riego automático, entre otras funciones. De esta manera, los gestos detectados por la aplicación de procesamiento de imágenes podrían facilitar la interacción y el control del entorno, especialmente para personas con discapacidades físicas.

7. Subsistema: Dispositivos de visualización

7.1. Descripción

El subsistema de dispositivos de visualización desempeña un papel fundamental al proporcionar a los usuarios la capacidad de supervisar y visualizar en tiempo real el funcionamiento del sistema en cuestión.

A través de un navegador web, los usuarios pueden acceder a Grafana, que está instalado en el servidor, para visualizar dashboards y gráficos que presentan datos sobre la detección de gestos y mensajes transmitidos mediante MQTT. Una de las funcionalidades más destacadas es la capacidad de acceder a un video en tiempo real proporcionado por la aplicación de procesamiento de imágenes. Este video muestra las imágenes procesadas en tiempo real, lo que brinda una visión actualizada y dinámica del proceso de imágenes y facilita el seguimiento de los gestos detectados.

7.2. Dispositivos utilizados

Se ha otorgado una gran importancia al estudio de los dispositivos que se utilizarán para interactuar con el sistema y obtener información en tiempo real de su funcionamiento.

Con este propósito, se presenta a continuación un cuadro comparativo que ofrece una visión general de las características más importantes de cuatro tipos de dispositivos comunes: la computadora de escritorio, la notebook, el celular y la tablet. Estos dispositivos son ampliamente empleados en diversos escenarios y ofrecen ventajas distintas en términos de tamaño de pantalla, portabilidad, potencia de procesamiento, conectividad, autonomía de batería y versatilidad.

Característica	Computadora de escritorio	Notebook	Celular	Tablet
Tamaño de pantalla	Grande	Mediano	Pequeño	Pequeño a mediano
Portabilidad	No portátil	Portátil	Portátil	Portátil
Potencia de procesamiento	Alta	Alta	Media	Media
Interfaz de usuario	Mouse y teclado	Mouse, trackpad y teclado	Pantalla táctil	Pantalla táctil
Conectividad	Ethernet y Wi-Fi (opcional)	Ethernet y Wi-Fi	Móvil y Wi-Fi	Móvil y Wi-Fi

Figura 7.1. Comparativa entre los distintos tipos de dispositivos de visualización considerados para el sistema.

El análisis detallado de cada dispositivo se realizó con el objetivo de asegurar que el sistema sea accesible, funcione de manera eficiente y brinde una experiencia óptima y satisfactoria a los usuarios. En este sentido, se eligieron la computadora de escritorio y el celular como dispositivos de visualización para el sistema debido a sus características y ventajas distintivas.

La computadora de escritorio fue seleccionada por su pantalla grande, lo cual permite una visualización detallada de datos, gráficos y videos en tiempo real generados por el sistema. Aunque carece de portabilidad, su tamaño de pantalla proporciona una experiencia de visualización cómoda y agradable.

Por otro lado, el celular fue elegido por su portabilidad y facilidad de uso en diferentes lugares y situaciones. El tamaño compacto lo convierte en una opción conveniente para usuarios que necesitan monitorear el sistema mientras se desplazan. Además, la presencia de pantallas táctiles y opciones de accesibilidad garantiza una interfaz de usuario intuitiva y facilita la interacción con el sistema, lo que es especialmente útil cuando la movilidad y la accesibilidad son prioritarias.

Seguidamente, se describe el procedimiento para acceder al dashboard utilizando los dos dispositivos seleccionados: la computadora de escritorio y el celular.

7.3. Conexión al dashboard a través de Internet

7.3.1. Computadora de escritorio

Para acceder a un dashboard de Grafana desde una computadora de escritorio es necesario cumplir con ciertos requisitos y seguir una serie de pasos específicos. A continuación, se proporciona una descripción detallada del proceso para el acceso a la plataforma de Grafana.

Los requisitos para acceder al dashboard son:

- Una computadora de escritorio con acceso a Internet.
- Un navegador web instalado en la computadora, como Google Chrome, Mozilla Firefox o Microsoft Edge.
- La dirección URL del dashboard de Grafana al que se desea acceder.
- Credenciales de inicio de sesión válidas para acceder al dashboard de Grafana.

Por otra parte, los pasos a seguir son:

1) Asegurarse que la computadora está encendida y conectada a Internet a través de un cable de Ethernet o por Wi-Fi.

2) Abrir el navegador web a elección.

3) En la barra de direcciones del navegador, ingresar la dirección URL del dashboard de Grafana. Luego, presionar “Enter” o el botón que permite navegar a la URL ingresada.

4) Si se solicita, ingresar las credenciales de inicio de sesión (nombre de usuario y contraseña) en los campos correspondientes. Una vez ingresados, presionar el botón de inicio de sesión.

5) Listo, ya se puede visualizar el dashboard de Grafana a través de la computadora de escritorio.

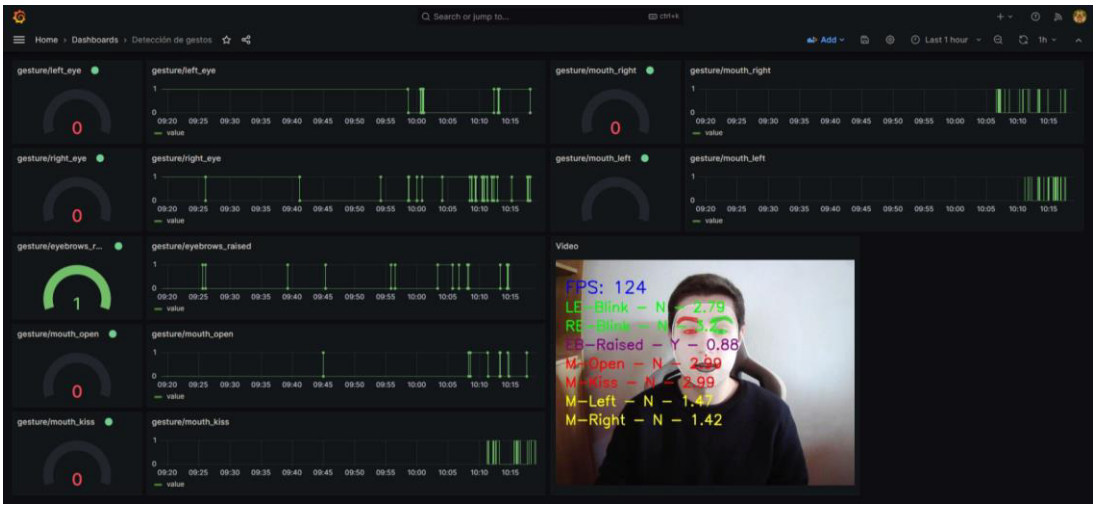

Figura 7.2. Vista del dashboard de Grafana en una computadora de escritorio.

7.3.2. Celular

Acceder a la aplicación de Grafana a través de un navegador web en un celular también es un proceso sencillo. A continuación, se detallan los requisitos y pasos a seguir para realizarlo de manera efectiva.

En primer lugar, los requisitos para acceder al dashboard son:

- Un celular con acceso a Internet.
- Un navegador web instalado.
- La dirección URL del dashboard de Grafana al que se quiere acceder.
- Credenciales de inicio de sesión válidas para acceder al dashboard de Grafana.

Una vez cumplidos los requisitos, los pasos a seguir son:

1) Asegurarse que el celular está conectado a una red Wi-Fi o tiene una conexión de datos móviles activa.
2) Abrir el navegador web en el celular.
3) En la barra de direcciones del navegador, ingresar la URL del dashboard de Grafana y presionar el botón para acceder al sitio.

4) Si se solicita, introducir las credenciales de inicio de sesión (nombre de usuario y contraseña) en los campos correspondientes. Luego, presionar el botón de inicio de sesión.

5) Listo, ya se puede visualizar el dashboard de Grafana en el celular.

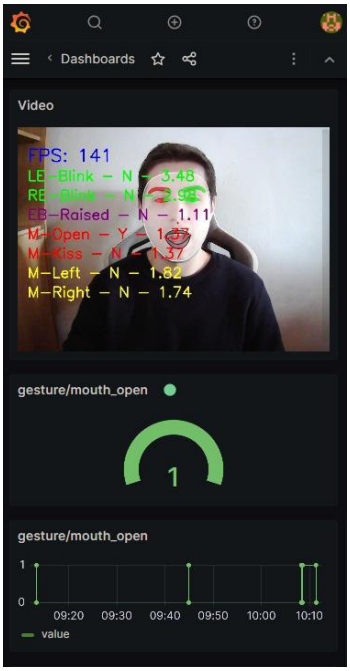

Figura 7.3. Vista del dashboard de Grafana en un celular.

8. Resultados

8.1. Calibración del sistema

Después de completar la implementación del sistema, procedimos a llevar a cabo su calibración. Para esto, aseguramos condiciones de luz favorables y posicionamos la cámara directamente hacia el rostro del usuario, manteniendo una distancia de entre 50 y 70 cm.

El objetivo de esta fase fue definir los umbrales necesarios para reconocer los diferentes gestos con precisión. Durante este proceso, evitamos el uso de objetos que pudieran obstruir el rostro, como auriculares, gorras, bufandas o lentes, ya que podrían afectar la malla facial devuelta por la librería MediaPipe.

Figura 8.1. Capturas tomadas durante el proceso de calibración.

Para establecer los umbrales, analizamos los valores generados por cada algoritmo en pruebas que se realizaron para cada uno de los gestos. Utilizando estos datos, definimos un rango que representara los resultados obtenidos cuando no se realizaba ningún gesto y cuando el gesto se ejecutaba correctamente. Este rango nos permitió definir un umbral específico para cada gesto, ubicado entre el valor registrado sin gesto y el valor obtenido al realizar el gesto de manera correcta.

Durante esta etapa, también se efectuaron pruebas para determinar el tiempo necesario para detectar un gesto de manera válida. Se concluyó que el tiempo válido es de un segundo. Detectar el gesto en un segundo asegura que el usuario lo realiza de manera deliberada y consciente, evitando falsos positivos que podrían ocurrir si el sistema detectara gestos

involuntarios o movimientos no intencionales. Además, un segundo es un tiempo razonable para que el usuario ejecute el gesto de forma natural.

8.2. Prueba completa del sistema

Una vez que el sistema fue debidamente calibrado y los umbrales fueron definidos, se procedió a realizar una prueba completa del mismo con el objetivo de verificar la precisión en la detección de los gestos. Estas pruebas se llevaron a cabo en condiciones idénticas a las utilizadas durante la calibración, garantizando que el usuario se encontrará en la misma posición, bajo la misma iluminación y con el rostro en la misma posición en relación a la cámara.

Durante las pruebas, los gestos fueron detectados con éxito y las respuestas proporcionadas por todas las aplicaciones fueron las esperadas. Este resultado comprueba la efectividad del sistema bajo condiciones favorables, lo que representa un paso significativo hacia su óptimo rendimiento en entornos reales.

8.3. Evaluación bajo condiciones desfavorables

Las pruebas que se presentan a continuación se efectuaron con el fin de verificar la fiabilidad del sistema en entornos reales y situaciones adversas. Se hizo hincapié en evaluar la respuesta del sistema en condiciones de estrés y la capacidad de la aplicación de procesamiento de imágenes bajo condiciones ambientales desfavorables.

8.4. Evaluación bajo condiciones de estrés

Durante la evaluación bajo condiciones de estrés, se sometió el sistema a una prueba exhaustiva durante varios minutos, en los que se realizaron gestos repetitivos para verificar el correcto funcionamiento de cada subsistema y aplicación.

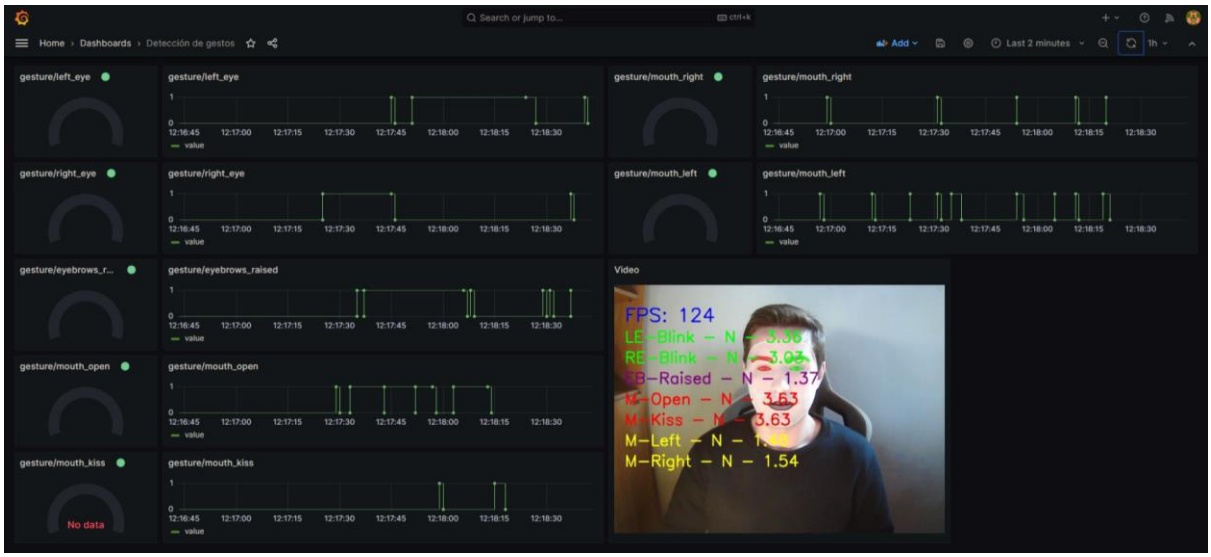

Figura 8.2. Captura tomada durante la prueba de estrés al sistema.

Todas las aplicaciones en el servidor respondieron de manera adecuada ante estas condiciones. EMQX continuó cumpliendo su función como bróker MQTT, asegurando que no se perdieran mensajes durante el proceso. Node-RED, por su parte, captó todos los mensajes y los almacenó sin problemas en la base de datos. Cabe destacar que Grafana experimentó cierto impacto, mostrando dificultades en la actualización en tiempo real de los paneles que estaban suscritos a tópicos MQTT, mientras que los paneles relacionados con la conexión a la base de datos funcionaron sin inconvenientes.

El rendimiento de la aplicación de procesamiento de imágenes se mantuvo sólido, con un ritmo constante y sin disminución en la cantidad de fotogramas procesados por segundo.

Los microcontroladores demostraron un funcionamiento adecuado. Los ESP32 recibieron los mensajes de control provenientes del ESP32-CAM y respondieron adecuadamente a cada tópico, ejecutando las acciones correspondientes. El ESP32-CAM, responsable de proporcionar las imágenes a la aplicación de procesamiento, cumplió eficientemente las solicitudes en tiempo real.

Es importante mencionar que, al operar la aplicación durante un tiempo prolongado, se observó que el ESP32-CAM experimentó un alarmante calentamiento. Este aspecto debe ser objeto de una evaluación más detallada para determinar su capacidad de trabajo sostenible sin afectar su rendimiento.

8.4.1. Evaluación de la robustez de la detección de gestos

Posteriormente, se llevó a cabo una evaluación de la robustez del sistema frente a condiciones ambientales desfavorables. Para ello, se realizaron diversas pruebas en diferentes entornos para evaluar su respuesta.

En primer lugar, se examinó cómo se comportaba el procesamiento de imágenes al utilizar lentes. Lamentablemente, se observó que la malla facial proporcionada por la librería MediaPipe perdía considerable precisión en las zonas cercanas a los ojos cuando se empleaba este accesorio. Como resultado, la capacidad del sistema para detectar guiños con ambos ojos se vio gravemente afectada al usar lentes.

Figura 8.3. Problemas al detectar el guiño del ojo derecho al usar lentes.

Asimismo, se realizó una prueba similar al emplear un auricular over-ear, y en esta ocasión, los gestos fueron detectados satisfactoriamente por el sistema.

Figura 8.4. Captura tomada durante la prueba de detección de gestos utilizando auriculares over-ear.

La segunda prueba consistió en evaluar la respuesta del sistema en condiciones de baja luminosidad. Dado que el sistema había demostrado un buen desempeño en condiciones de luz favorables, se buscó determinar cómo afectaría la falta de luz a la detección de gestos.

En escenarios donde aún era posible reconocer los rasgos faciales, el sistema no presentó problemas para detectar la mayoría de los gestos realizados. Sin embargo, la precisión en la detección de los guiños se vio comprometida, debido a que la falta de luz dificulta la generación de una malla facial precisa en la región ocular.

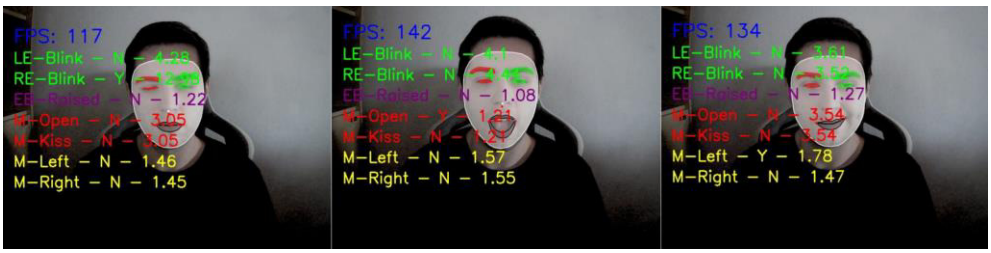

Figura 8.5. Capturas de pruebas efectuadas en condiciones de baja luminosidad. Los gestos capturados incluyen un guiño del ojo derecho, la apertura de la boca y una mueca hacia la izquierda.

Se determinó que cuando la imagen muestra un alto nivel de ruido debido a la escasez lumínica, la malla facial pierde mucha precisión, lo que resulta en una detección de gestos completamente aleatoria.

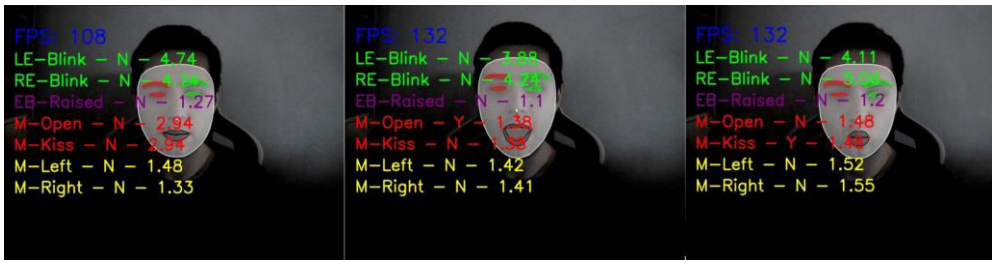

Figura 8.6. Capturas tomadas en condiciones de escasa luminosidad. El sistema logró detectar con precisión los gestos de apertura de boca y beso, aunque ya no puede capturar el guiño con el ojo derecho.

La tercera prueba tuvo como objetivo evaluar la detección de gestos al modificar el ángulo entre la cabeza del usuario y la cámara con respecto al eje longitudinal.

Para ángulos menores a aproximadamente 30 grados, el sistema seguía detectando correctamente los gestos.

Figura 8.7. Capturas tomadas con la cabeza en ángulo de aproximadamente 30 grados con respecto a la cámara. Los gestos detectados incluyen el guiño del ojo derecho, una mueca hacia la derecha y levantar las cejas.

Sin embargo, al superar este umbral, la detección de muchos gestos comenzó a presentar fallos. Por ejemplo, el gesto de la mueca hacia cualquiera de los dos lados se activaba de forma involuntaria. El motivo de esta falla radica en que el cálculo de las proporciones arroja resultados distintos según el ángulo en que se encuentra el usuario, y esta variabilidad no ha sido considerada en la implementación de la aplicación de procesamiento de imágenes. Además, es importante mencionar que la malla facial generada a través del procesamiento de la imagen también presentaba graves errores, debido a que no puede visualizar todas las regiones de la cara necesarias para su correcta formación.

La última prueba consistió en evaluar los efectos de modificar la distancia entre el usuario y la cámara, estando por fuera del rango establecido durante la calibración de 50 a 70 cm.

Figura 8.8. Captura de la detección de un guiño a una distancia aproximada de un metro.

Los resultados de estas pruebas demostraron que la detección de gestos continuó siendo satisfactoria, siempre y cuando el usuario no se alejara más de un metro de la cámara ni se posicionara a una distancia menor a 40 cm.

8.5. Análisis de la solución y mejoras

El sistema ha demostrado ser efectivo en la detección de gestos en condiciones favorables, como cuando el usuario se encuentra en la misma posición y con buena iluminación. Asimismo, los distintos componentes del sistema respondieron adecuadamente bajo situaciones de estrés. También se observó una respuesta positiva del sistema ante la modificación de la distancia entre el usuario y la cámara dentro de parámetros razonables.

Sin embargo, durante las pruebas, se identificaron varias limitaciones en el funcionamiento del sistema, especialmente en la detección de gestos. La calibración manual de los umbrales para los gestos representa una dificultad y puede afectar la precisión de la detección en diferentes escenarios. Además, el sistema mostró dificultades en la detección precisa de gestos en situaciones cotidianas. Por ejemplo, cuando los usuarios llevaban lentes, la detección de guiños se vio afectada negativamente. Otro aspecto a mejorar es la detección de gestos cuando el usuario se encuentra a ángulos mayores a 30 grados con respecto a la cámara, ya que esto generaba errores en la detección de algunos gestos.

En vista de estas limitaciones, se plantean diversas mejoras que podrían implementarse en un futuro:

- **Incorporación de un sistema de calibrado automático:** Implementar un sistema de calibrado automático para los usuarios permitiría ajustar dinámicamente los umbrales

y parámetros de detección, mejorando la precisión en diversas condiciones y para diferentes usuarios.

- **Mejora en la detección de gestos bajo condiciones desfavorables:** Se pueden aplicar técnicas avanzadas de procesamiento de imágenes, como algoritmos de aprendizaje profundo, para mejorar la detección de gestos en situaciones desafiantes, como condiciones de iluminación variables o la presencia de accesorios.

- **Ampliar el rango de detección de gestos:** Adaptar los algoritmos para tener en cuenta la variabilidad en las proporciones faciales según el ángulo en que el usuario se encuentra con respecto a la cámara podría mejorar la precisión de detección en cualquier posición.

- **Explorar distintas tecnologías de detección de gestos:** Investigar y comparar diferentes librerías y tecnologías de detección de gestos podría ayudar a encontrar soluciones más precisas y robustas en diferentes entornos.

- **Diseñar una plataforma web para visualizar los datos analizados:** Implementar una aplicación web permitiría visualizar las imágenes resultantes del proceso de la imagen y la información relacionada con los gestos detectados, ofreciendo una mayor adaptabilidad a las necesidades específicas de la aplicación.

- **Probar el sistema con otro tipo de actuadores:** Se podrían explorar diferentes tipos de actuadores que permitan, por ejemplo, controlar la apertura de puertas, iluminación y calefacción. Integrar nuevos actuadores al sistema ampliaría sus capacidades y posibilidades de uso en aplicaciones de control y automatización del hogar.

- **Mejorar el enfriamiento del ESP32-CAM para evitar sobrecalentamiento:** Se podría mejorar el enfriamiento mediante la incorporación de un cooler adicional u otros métodos que aumenten la disipación del calor. Esto ayudaría a mantener la temperatura del ESP32-CAM en rangos adecuados, evitando problemas relacionados con el sobrecalentamiento y asegurando un funcionamiento más estable y prolongado del sistema.

9. Conclusiones

El sistema desarrollado ha demostrado un gran potencial en la utilización de tecnologías de IoT y visión por computadora para mejorar significativamente la calidad de vida de personas con discapacidades físicas.

La implementación ha logrado superar con éxito los objetivos establecidos al inicio del proyecto. La detección de gestos se ha mostrado altamente efectiva y precisa en condiciones adecuadas. Gracias a la integración de herramientas IoT, es posible ejecutar acciones en microcontroladores como resultado de la detección de gestos.

Las pruebas exhaustivas realizadas bajo diversas condiciones han sido fundamentales para obtener un conocimiento del rendimiento y las limitaciones del sistema. Estos hallazgos proporcionan una base sólida para futuras mejoras y optimizaciones, con el objetivo de hacer frente a las limitaciones identificadas y expandir el alcance del sistema.

Además de su aplicabilidad para personas con discapacidades físicas, el sistema ha demostrado un potencial enorme para su uso en otras áreas. La adaptabilidad de las acciones resultantes de la detección de gestos a diferentes tareas y aplicaciones de IoT abre un amplio abanico de posibilidades para mejorar la experiencia de usuario en diversos contextos, más allá de los planteados al comienzo del trabajo. Por ejemplo, la detección de gestos podría utilizarse para la apertura de puertas, el control de la iluminación, la gestión de la calefacción o aire acondicionado, el manejo de persianas y cortinas, e incluso para funciones como el riego automático de jardines.

A nivel personal, este trabajo ha sido una experiencia enriquecedora y gratificante. Me ha permitido sumergirme en el mundo de las tecnologías de IoT y visión por computadora, investigar y emplear numerosas herramientas, y desarrollar habilidades técnicas y de resolución de problemas. Considero que ha sido una excelente manera de sintetizar los conocimientos adquiridos a lo largo de toda la carrera.

10. Bibliografía

1) *MQTT - The Standard for IoT Messaging*. (s. f.). https://mqtt.org/

2) *Arduino Docs*. (s. f.). Arduino Documentation. https://docs.arduino.cc/

3) *ESP-IDF Programming Guide - ESP32 - — ESP-IDF Programming Guide latest documentation*. (s. f.). https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

4) *ESP-NOW - ESP32 - — ESP-IDF Programming Guide Latest Documentation*. (s. f.). https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html

5) Cameron, N. (2020). *Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices*. Apress.

6) Aguilar, L. J. (2017). *Industria 4.0: la cuarta revolución industrial*. Grupo Editorial Patria.

7) Lea, P. (2020). *Internet of Things for Architects: Architecting IoT Solutions by Implementing Sensors, Communication Infrastructure, Edge Computing, Analytics, and Security*. Van Haren Publishing.

8) Rahman, L. F., Ozcelebi, T., & Lukkien, J. J. (2016). Choosing Your IoT Programming Framework: Architectural Aspects. *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. https://doi.org/10.1109/ficloud.2016.49

9) Pajankar, A. (2020). *Raspberry Pi Computer Vision Programming -Second Edition: Design and Implement Computer Vision Applications with Raspberry Pi, OpenCV, and Python 3*. Van Haren Publishing.

10) Laganière, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 Recipes to Master this Library of Programming Functions for Real-time Computer Vision*. Van Haren Publishing.

11) *OpenCV: OpenCV modules*. (s. f.). https://docs.opencv.org/4.x/index.html

12) MediaPipe | Google for Developers. (s. f.). *Google for Developers*. https://developers.google.com/mediapipe

13) *Woerner, T. (s. f.). Documentation. firewalld. https://firewalld.org/documentation/*

14) *NGINX Reverse Proxy*. (s. f.). NGINX Documentation. https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

15) *Documentation : Node-RED*. (s. f.). https://nodered.org/docs/

16) *Grafana Documentation | Grafana Documentation*. (s. f.). Grafana Labs. https://grafana.com/docs/grafana/latest/

17) *MariaDB Server Documentation*. (s. f.). MariaDB KnowledgeBase. https://mariadb.com/kb/en/documentation/

18) *Introduction*. (s. f.). EMQX 5.1 Documentation. https://www.emqx.io/docs/en/v5.1/

19) Yoursunny. (s. f.). *GitHub - YourSunny/ESP32Cam: OV2640 Camera on ESP32-CAM, Arduino Library*. GitHub. https://github.com/yoursunny/esp32cam

20) *Welcome to ESP32 Arduino Core's documentation — Arduino-ESP32 2.0.6 documentation*. (s. f.). https://docs.espressif.com/projects/arduino-esp32/en/latest/