Implementación de un sistema de control de acceso basado en detección y reconocimiento facial utilizando un computador embebido.

Pogo Torres, Elías David

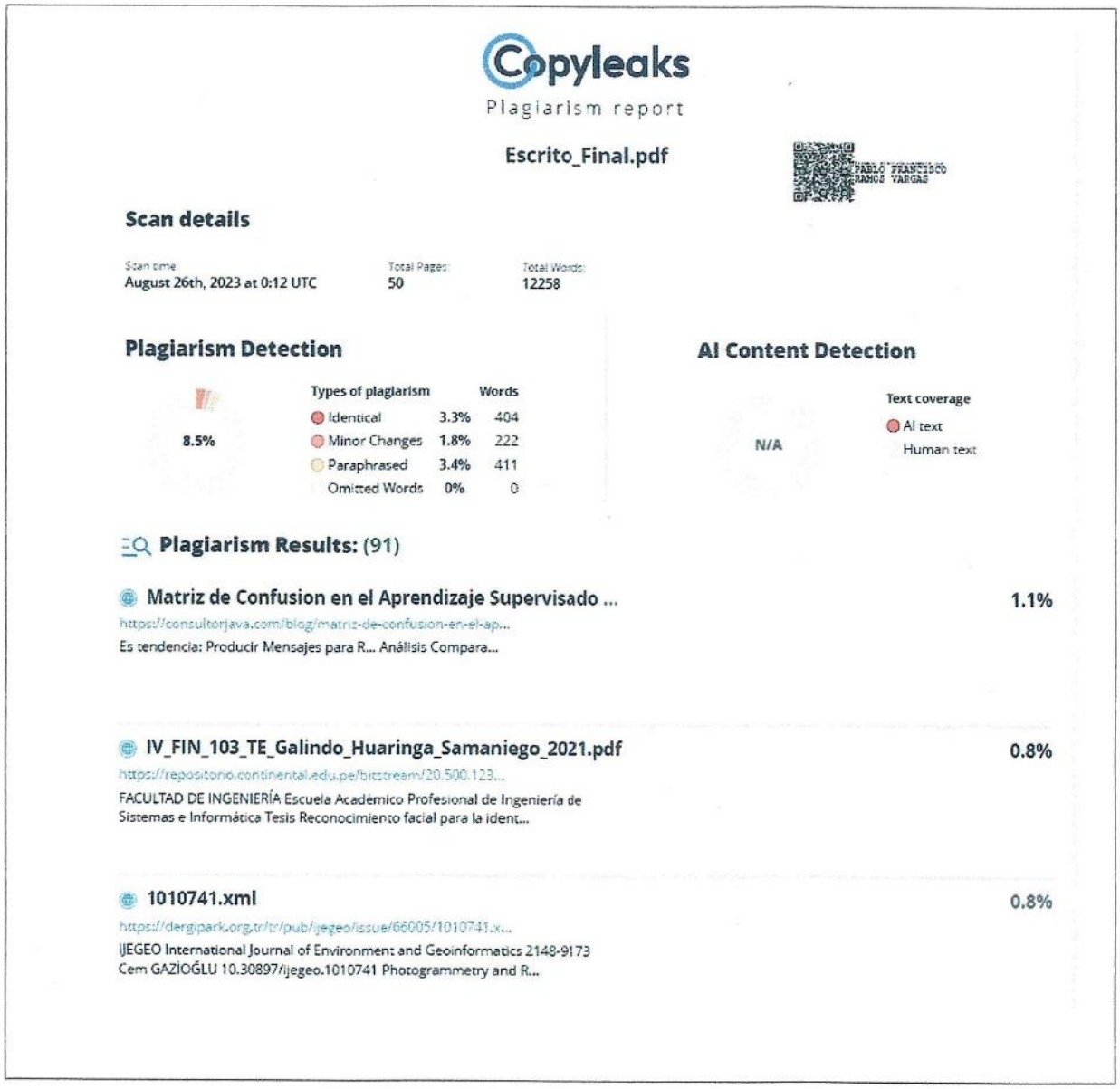
Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Telecomunicaciones

Trabajo de Integración Curricular, previo a la obtención del título de Ingeniero en Telecomunicaciones

Ing. Ramos Vargas, Pablo Francisco

25 de agosto del 2023



Copyleaks

Plagiarism report

Escrito_Final.pdf

PABLO FRANCISCO RAMOS VARGAS

Scan details

Scan time	Total Pages:	Total Words:
August 26th, 2023 at 0:12 UTC	50	12258

Plagiarism Detection

8.5%

Types of plagiarism		Words
Identical	3.3%	404
Minor Changes	1.8%	222
Paraphrased	3.4%	411
Omitted Words	0%	0

AI Content Detection

N/A

Text coverage
- AI text
- Human text

Plagiarism Results: (91)

Matriz de Confusion en el Aprendizaje Supervisado ... 1.1%

https://consultorjava.com/blog/matriz-de-confusion-en-el-ap...

Es tendencia: Producir Mensajes para R... Análisis Compara...

IV_FIN_103_TE_Galindo_Huaringa_Samaniego_2021.pdf 0.8%

https://repositorio.continental.edu.pe/bitstream/20.500.123...

FACULTAD DE INGENIERÍA Escuela Académico Profesional de Ingeniería de Sistemas e Informática Tesis Reconocimiento facial para la ident...

1010741.xml 0.8%

https://dergipark.org.tr/tr/pub/ijegeo/issue/66005/1010741.x...

IJEGEO International Journal of Environment and Geoinformatics 2149-9173 Cem GAZİOĞLU 10.30897/ijegeo.1010741 Photogrammetry and R...

Firmado electrónicamente por: PABLO FRANCISCO RAMOS VARGAS

.....

Ing. Ramos Vargas, Pablo Francisco PhD.

Director

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Telecomunicaciones

Certificación

Certifico que el trabajo de integración curricular: "**Implementación de un sistema de control de acceso basado en detección y reconocimiento facial utilizando un computador embebido**" fue realizado por el señor **Pogo Torres, Elías David** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente

Sangolquí, 25 de agosto de 2023

PABLO FRANCISCO RAMOS VARGAS

.....

Ing. Ramos Vargas, Pablo Francisco PhD.

C.C.: 1712447976

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Telecomunicaciones

Responsabilidad de Autoría

Yo, **Pogo Torres, Elías David**, con cédula de ciudadanía n° 1718715046, declaro que el contenido, ideas y criterios del trabajo de integración curricular: **Implementación de un sistema de control de acceso basado en detección y reconocimiento facial utilizando un computador embebido** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 25 de agosto de 2023

Pogo Torres, Elías David

C.C.: 1718715046

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Telecomunicaciones

Autorización de Publicación

Yo, **Pogo Torres, Elías David**, con cédula de ciudadanía n° 1718715046, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Implementación de un sistema de control de acceso basado en detección y reconocimiento facial utilizando un computador embebido** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 25 de agosto de 2023

.....

Pogo Torres, Elías David

C.C.: 1718715046

Dedicatoria

El presente trabajo está dedicado de manera muy especial a mis padres, Elías y Miriam, por su amor incondicional y apoyo constante, quienes me han colmado de sabiduría y me han enseñado el valor del trabajo. A mis hermanos, Andrea y Fernando, por su apoyo inquebrantable en cada etapa de mi vida. A mis amigos y seres queridos, por brindarme el ánimo y la alegría que necesitaba en los momentos más difíciles. En general para todos aquellos que de alguna manera contribuyeron con este logro, gracias por ser parte de mi vida.

Pogo Torres, Elías David

Agradecimiento

Quiero expresar mi más profunda gratitud a todas las personas que han contribuido de manera significativa a la realización de este trabajo. Agradezco a mi director de proyecto, Ing. Pablo Ramos, por su orientación, paciencia y compromiso. Sus valiosas sugerencias y comentarios han enriquecido enormemente este trabajo de investigación. A todos los profesores del Departamento de Eléctrica, Electrónica y Telecomunicaciones de la Universidad de las Fuerzas Armadas ESPE, por sus enseñanzas y conocimientos que han servido como una base sólida para la elaboración de este documento. A mi familia, por su amor incondicional y constante aliento. Gracias por ser mi fuente de inspiración.

Pogo Torres, Elías David

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El hogar inteligente "*Smart Home*" se ha convertido a lo largo de los años, en una de las aplicaciones más conocidas de IoT, debido a que proporciona seguridad y confort a los domicilios, por lo que en la actualidad se hace cada vez más necesaria. En el mercado hay disponibles sistemas de hogar inteligente propietarios y de código abierto que usan nodos que se encargan del monitoreo de variables físicas por medio de sensores, o del control de objetos por medio de actuadores. La carga computacional del sistema recae generalmente sobre servidores locales o en la nube, pero para ciertas aplicaciones sensibles a la latencia, como la seguridad electrónica, podría ser conveniente que el propio nodo realice el procesamiento de datos para la toma de decisiones. Esto se podría lograr mediante el uso de la computación de borde "*Edge computing*". En este contexto, se desea probar esta tecnología en un nodo IoT de hogar inteligente para el control de accesos. De allí que, el presente trabajo de Unidad de Integración Curricular intenta fusionar las ventajas de la computación de borde y de IoT para proyectos de hogar inteligente, con el fin de mejorar la seguridad y comodidad. Para ello, se explorará un computador embebido que implementa un procesador de redes neuronales de arquitectura RISC V de 64 bits, con el cual se trabajará en el diseño e implementación de un sistema de control de acceso basado en detección y reconocimiento facial. El sistema debe ajustarse hasta obtener una efectividad de al menos 90%, es decir, debe reconocer correctamente al menos nueve de cada diez personas que lo usen. Para poder evaluar el desempeño de la detección y reconocimiento facial del módulo de inteligencia artificial hardware, se lo comparará con el reconocimiento facial software utilizando librerías tales como OpenCV sobre un computador de escritorio o laptop de similares características.

Palabras clave: Hogar inteligente, detección facial, reconocimiento facial, internet de las cosas, computador embebido

Abstract

"Smart Home" has become over the years, one of the best-known IoT applications, because it provides security and comfort to homes. For this reason, nowadays it is becoming a growing need. On the market there are proprietary and open-source smart home systems that use nodes for monitoring physical variables through sensors or controlling objects through actuators. The computational load of the system generally lies on local servers or in the cloud, but for certain applications latency-sensitive, such as electronic security, it could be convenient that the node itself performs data processing for decision- making. This could be achieved through the use of “Edge Computing”. In this context, edge computing concept will be applied in smart home IoT for access control. Hence, the present Curricular Integration work attempts to merge the advantages of edge computing and IoT for smart home projects for improving the security and comfort. To accomplish this goal, an embedded computer that implements a 64-bit RISC V architecture neural network processor will be explored and used for the design and implementation of an access control system based on facial detection and recognition. The system must be adjusted to obtain an effectiveness of at least 90%, that is, it must correctly recognize at least nine out of ten people who use it. In order to evaluate the performance of facial recognition and detection of the hardware AI module, it will be compared with software facial recognition using libraries such as OpenCV on a desktop or laptop computer with similar characteristics.

Keywords: Smart home, face detection, face recognition, internet of things, embedded computer

Capítulo I: Introducción

Antecedentes

Las áreas de trabajo del presente documento tales como reconocimiento facial y *Edge Computing* han sido objeto de numerosos estudios. A continuación, se realizará una breve descripción de algunas investigaciones lo cual permitirá obtener un punto de vista focalizado en el estado actual de estos campos.

Por el lado de reconocimiento facial, el trabajo de Cabello Pardos (2004) señala el avance que se ha obtenido en esta área gracias al uso de PCA (Análisis de componentes, del inglés *Principal Component Analysis*) y distintas técnicas de clasificación como SVM (Máquinas de vector de soportes, del inglés *Support Vector Machines*), o KNN (K-Vecinos más cercanos, del inglés *K-Nearest Neighbors*) como parte de la introducción de este campo en el ML (Aprendizaje de Máquina, del inglés *Machine Learning*). Por su parte Scarel & Müller (2010) muestra una estructura generalizada de las etapas requeridas en los programas de reconocimiento facial, desde la captura del rostro en una imagen hasta su clasificación e identificación final. De igual manera Bravo et al. (2018) y Jaramillo (2021) indican el potencial y la aceptación que el reconocimiento facial tiene dentro la materia de seguridad, trabajos donde se denota que en la actualidad el uso de datos biométricos como el reconocimiento facial pueden ayudar a identificar y autenticar a individuos a través de características físicas únicas.

El *Edge Computing* es un amplio campo de investigación. Para obtener un enfoque general se puede leer el trabajo de Cao et al. (2020), donde se cuenta de manera concisa como la Computación frontera aporta al presente y futuro del IoT (Internet de las Cosas, del inglés *Internet of Things*). Del mismo modo es de gran ayuda conocer los trabajos realizados por Khan et al. (2019) y Zhang et al. (2019) ambos se centran en la ejecución de un sistema de reconocimiento facial desarrollado con la ayuda de algoritmos CNN (Redes neuronales convolucionales, del inglés *Convolutional neuronal*

networks) y *Edge Computing,* trabajos que alcanzan una efectividad del 97.9% y 88.56% respectivamente.

Por otra parte, en el mercado de inteligencia artificial podemos encontrar la tarjeta MaixDuino de la empresa china Sipeed (Lanzada en el año 2020), que al ser de bajo costo se convierte en una solución prometedora para proyectos iniciales. No existen muchos registros de investigaciones realizados con la MaixDuino, sin embargo, podemos mencionar a (Basilio et al., 2021), trabajo que se centró en la detección de tipos de mascarilla usando procesamiento de imágenes y DL (Aprendizaje profundo, del inglés *Deep Learning*) alcanzando una efectividad de entre el 80% y 90%. Adicionalmente tenemos el trabajo realizado por Bouaouda et al. (2022), el cual refleja la integración de un sistema de visón artificial, como medida de accesibilidad para personas con capacidades especiales en extremidades superiores. Este trabajo implementado en una MaixDuino muestra también la adhesión del software domótico de código abierto Home Assistant para el uso en un entorno de Hogar Inteligente.

Motivación

Continuamente el ser humano siempre ha buscado la comodidad y seguridad en su hogar, lo que se suma a la eficiencia energética para mejorar la calidad de vida. Estos aspectos en las últimas décadas han tenido grandes avances gracias al Smart Home. Dentro del tema de seguridad, el estudio de datos biométricos ofrece muchas ventajas significativas por medio del reconocimiento facial. Este es uno de los campos más explotados en aplicaciones como desbloqueo de celulares, identificación de personas sospechosas, autenticación biométrica entre otras funciones. Todo esto conlleva a la adopción de dispositivos inteligentes y la generación masiva de datos, problemas en los cuales la computación de borde juega un papel muy importante porque mejora la eficiencia, la seguridad y la privacidad de las aplicaciones y servicios conectados, además de habilitar nuevas oportunidades en áreas como IoT.

Por todo lo expuesto anteriormente, el presente trabajo de titulación busca unir las ventajas de la computación de borde y del IoT para implementar un sistema de control de acceso basado en detección y reconocimiento facial utilizando un computador embebido.

Importancia

En los últimos años el Ecuador ha visto un aumento en la inseguridad y delitos cometidos. Basándonos en las cifras dadas por la fiscalía general del Estado, entre el año 2020 y 2021 (cifras públicas más actuales) existió en promedio un aumento del 12% en el robo a domicilios y viviendas, llegando a la cifra de 7449 cometidos (fiscalía general del Estado, 2022). Esta temática continúa siendo un problema para las personas que a diario deben salir a cumplir con sus deberes laborales y cotidianos. Estos problemas han orillado a la ciudadanía a buscar soluciones para evitar ser víctimas de estos delitos. Es así que la instalación de sistemas de seguridad como cámaras, alarmas y los sistemas de reforzamiento de puertas y ventanas han crecido considerablemente.

Aunque no muy conocidos en el país, los hogares inteligentes también pueden ser parte de la solución, pues además de ser sistemas que facilitan la comodidad, se integran perfectamente con los sistemas de seguridad antes mencionados.

Alcance

Para el desarrollo de este proyecto se plantea trabajar con la tarjeta de desarrollo MaixDuino, una placa basada en el chip Kendryte K210, el cual es un procesador que cuenta con una arquitectura de doble núcleo RISC-V, dos NPU (Unidades de procesamiento de redes neuronales, del inglés *Neural Network Processing Unit*), una memoria RAM interna de 8MB entre otras conexiones y periféricos integrados. Junto con el entorno de desarrollo MaixPy, el cual es una versión personalizada de MicroPython optimizada para el chip Kendryte K210. Esto facilita la programación y el desarrollo de aplicaciones de IA en la placa. Además, la empresa fabricante de la placa, SiPeed, cuenta con varias redes oficiales, en las cuales existe una comunidad que continuamente se va retroalimentando para lograr un mejor desarrollo de proyectos.

Para la parte de Smart Home se trabajará con el software Home Assistant, una plataforma de domótica de código abierto muy popular que permite controlar y automatizar diversos dispositivos y servicios inteligentes en el hogar. Su objetivo es unificar y facilitar la gestión de todos los dispositivos conectados en una sola interfaz y brindar a los usuarios la posibilidad de crear automatizaciones y escenarios personalizados. Además, una de sus ventajas más importantes y motivo para trabajar con él, es su simplicidad a la hora de usar distintos protocolos, específicamente el protocolo MQTT, el cual es en la actualidad el más usado para temas de IoT.

El presente trabajo de titulación busca ayudar a la sociedad de investigadores con el diseño e implementación de un sistema de reconocimiento facial dentro de un computador embebido. Se espera que la implementación de este modelo proporcione una valiosa contribución al aprendizaje de sistemas inteligentes, reconocimiento facial y provea una buena retroalimentación para mejorar el desarrollo de productos de seguridad de bajo costo.

Objetivos

Objetivo general

Implementar un sistema de control de acceso basado en detección y reconocimiento facial utilizando un computador embebido.

Objetivos específicos

- Estudiar un computador de placa única con capacidad de inteligencia artificial hardware e IoT.
- Realizar un programa para el reconocimiento facial utilizando el módulo de inteligencia artificial del computador embebido.
- Realizar un programa de reconocimiento facial software utilizando programación en Python y librería OpenCV.
- Realizar las pruebas del sistema enrolando varios usuarios para determinar su efectividad y establecer comparaciones con el programa de reconocimiento facial software realizado.

Capítulo II: Marco Teórico

Reconocimiento Facial

El reconocimiento facial automático es un área de investigación que se ha venido desarrollando en las últimas décadas, los primeros trabajos en este ámbito fueron realizados en los principios del siglo XXI. Al igual que una persona reconoce a sus semejantes, el objetivo principal de este campo es identificar un rostro emulando el proceso cognitivo que realiza el cerebro humano. (W et al., 2003)

Aunque existen varias técnicas usadas para lograr este objetivo, el RF necesita al menos las siguientes tres etapas: 1) detección y captura de un rostro, 2) extracción de características de la imagen y 3) la identificación del individuo mediante la clasificación de sus características únicas. (Cabello Pardos, 2004)

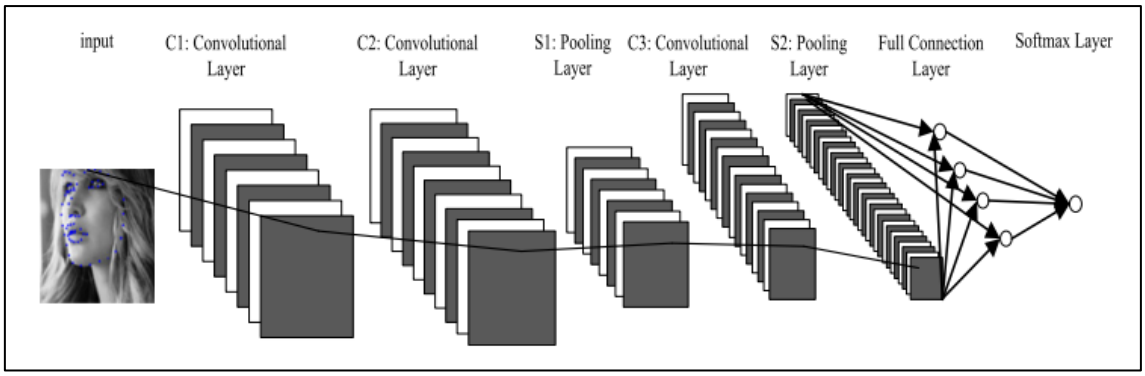
Redes Neuronales Convolucionales (CNN)

Una red neuronal convolucional es un algoritmo de Deep Learning, que puede diferenciar varios objetos de una imagen de entrada. Está inspirada en la funcionalidad de la corteza visual del cerebro humano. Uno de los aspectos más destacados de las CNN es que reducen el número de parámetros de entrada en comparación con las ANN (Redes neuronales artificiales, del inglés *Artificial Neural Networks*) (Albawi et al., 2017).

Su funcionamiento se basa en extraer características a diferentes niveles de la imagen. Esto permite obtener una imagen personalizada, la cual, junto con el primer filtro, definen sus características. Luego, se aplica un segundo filtro a una segunda imagen para detectar otro tipo de características y así sucesivamente, tal y como se muestra en la **Figura *1***. De esta forma una red neuronal convolucional utiliza las predicciones de las capas para representar la probabilidad de que una característica particular pertenezca a una clase específica. Así, produce una salida final que presenta un vector de puntos de probabilidad (Atik & Ipbuker, 2021).

Figura 1

Estructura CNN para reconocimiento facial



Nota. Estructura de redes neuronales convolucionales para un sistema de reconocimiento facial. Tomado de (Zhang et al., 2019)

YOLO v2

Presentado por Redmon et al. (2016), YOLO es un algoritmo de visión por computadora que permite detectar objetos en imágenes y videos en tiempo real. En la actualidad es uno de los más usados debido a su eficiencia y capacidad para la detección de objetos. Desde su primera presentación, se han publicado varias versiones posteriores de YOLO, siendo las principales YOLOv2, YOLOv3, YOLOv4 y YOLOv5.

La primera versión de YOLO comete errores de localización. Por lo tanto, se puede decir que YOLO tiene un error de sensibilidad relativamente alto. En YOLOv2 (Redmon & Farhadi, 2017), el enfoque se centra en mejorar la exhaustividad y la localización mientras se mantiene la precisión de la clasificación. En esta versión se introdujo un nuevo diseño de red introduciendo la normalización por lotes, técnica que añade operaciones antes o después de activar cada capa oculta. Después se normaliza cada entrada y se hace un ajuste en cero. Luego, se ajusta la escala y se transforma el resultado mediante dos conjuntos de nuevos valores de parámetros específicos para cada capa. Este proceso capacita al modelo para entender las escalas y valores medios ideales para cada una de las entradas en cada capa (Rozada Raneros & otros, 2021).

Librería OpenCV

A mediados del 2000, Intel anunció que reuniría a investigadores para desarrollar una librería en lenguaje C enfocada en la visión por computador. Esta noticia tuvo lugar en la inauguración del IEEE-CVPR (La Conferencia sobre visión por computadora y reconocimiento de patrones-IEEE, del inglés *IEEE*-Computer Society Conference on Computer Vision and Pattern Recognition). De esta manera nació la librería OpenCV (Librería abierta de visión por computadora, del inglés *Open Computer Vision Library*) bajo una licencia BSD (Distribución de software Berkeley, del inglés *Berkeley Software Distribution*) (Bradski et al., 2000).

Si bien el proyecto originalmente fue desarrollado en lenguaje C, OpenCV también ofrece interfaces para varios lenguajes de programación, incluido Python. Por todo esto OpenCV se convierte en una herramienta poderosa y versátil para cualquier proyecto que involucre procesamiento de imágenes y visión por computadora, desde tareas básicas de manipulación de imágenes hasta proyectos más avanzados de detección, seguimiento y análisis de objetos en tiempo real.

Domótica

Un sistema domótico es un sistema que permite recopilar información de sensores para procesarla y transmitir órdenes a las salidas. El internet de las cosas hace referencia a la interconexión de dispositivos, y al estar conectado a la red permite el control de los aparatos de un hogar sin estar en él. Dentro de este contexto generalmente se suele usar el término hogar inteligente, derivado de *Smart Home,* muy popular en Estados Unidos y empleado antes de que naciese el de domótica (Huidobro & Tejedor, 2010).

Seguridad

La seguridad es una de las aplicaciones más conocidas de Smart Home, ya que puede aumentar el nivel de seguridad de diferentes formas, como son: Seguridad de bienes a través del

control de acceso y alarmas, seguridad de las personas mediante el servicio de teleasistencia, e incidentes que son detectados por sensores especiales.

Edge Computing

En la actualidad la demanda de servicios y aplicaciones que requieren acceso y uso de Internet ha crecido constantemente, aumentando también los requisitos de procesamiento y almacenamiento de datos lo cual se convirtió en un problema por la cantidad de información generada por sensores, actuadores y otros dispositivos.

La computación de borde o *Edge Computing* representa una solución para minimizar la carga de procesamiento y almacenamiento en la nube. Apareció en el año 2002 aproximadamente y trata de ubicar los recursos de computación en el borde de la red, de modo que la computación ocurra cerca de las fuentes de datos (Zhao et al., 2019).

En los últimos años el Edge Computing se encarga de los datos más cerca de la fuente de información. Por lo tanto, los datos pueden almacenarse y procesarse localmente sin necesidad de cargar todos los datos a la nube. La reducción de la carga en la red mejora en gran medida la eficiencia de utilización del ancho de banda de la red. Por esta razón, la computación en la nube y la computación en el borde desempeñan un papel importante en el desarrollo futuro del Internet de las Cosas inteligente (Jararweh et al., 2016).

Edge Computing vs Cloud Computing

Se puede decir que el *Edge Computing* es una extensión de la computación en la nube, que tiene sus propias características en comparación con la nube. La característica principal de la computación en la nube es que puede abarcarlo todo, procesar una gran cantidad de datos, llevar a cabo análisis profundos y también desempeñar un papel importante en el procesamiento de datos no en tiempo real, como la toma de decisiones empresariales y otros campos (Cao et al., 2020). El *Edge Computing* se enfoca en lo local y puede tener un mejor rendimiento en análisis inteligentes en tiempo real a pequeña escala, como satisfacer las necesidades en tiempo real de las empresas

locales. Por lo tanto, en aplicaciones inteligentes, la computación en la nube es más adecuada para el procesamiento centralizado de datos a gran escala, mientras que el *Edge Computing* puede usarse a pequeña escala y servicios locales (Zhao et al., 2019).

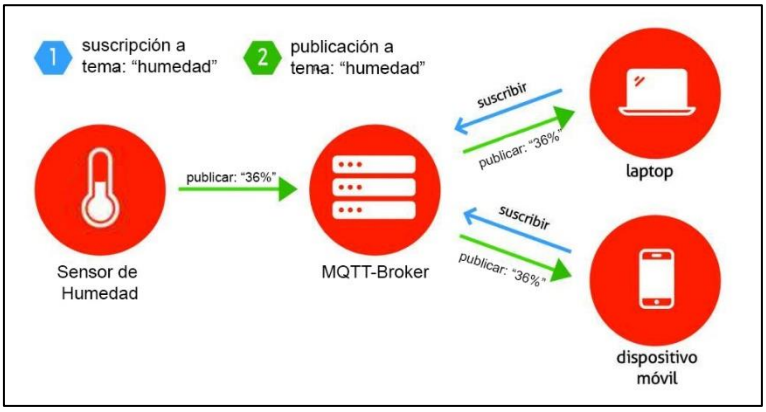
Protocolo MQTT

Arquitectura MQTT

Como se observa en la **Figura 2** el funcionamiento de este protocolo se basa en un modelo de publicación/suscripción, a continuación, se detallan los conceptos necesarios para comprenderlo de mejor manera:

Figura 2

Arquitectura del protocolo MQTT



Nota. Modelo de la arquitectura publicación/suscripción del protocolo MQTT

Clientes MQTT. Los dispositivos que participan en la comunicación MQTT se denominan "clientes". Pueden ser sensores, actuadores, microcontroladores, aplicaciones, servidores, etc. Estos se conectan a un "broker MQTT", que actúa como intermediario en la comunicación.

Broker MQTT. El broker MQTT es el intermediario central en el protocolo. Recibe mensajes de los clientes y luego los envía a los destinatarios adecuados. Maneja la lógica de suscripción y publicación, lo que permite a los clientes enviar mensajes a canales específicos (temas) y recibir mensajes de estos canales.

Temas (Topics). Los temas son canales virtuales a los que los clientes pueden suscribirse y publicar mensajes. Los temas son cadenas de texto que actúan como identificadores de mensajes. Los mensajes publicados en un tema se distribuyen a todos los clientes que están suscritos a ese tema.

QoS (Quality of Service). MQTT ofrece diferentes niveles de calidad de servicio (QoS) para garantizar la entrega de mensajes. Puede tener 3 valores que son: 0) Entrega al menos una vez. No se confirma la entrega, 1) Entrega al menos una vez con confirmación y 2) Entrega exactamente una vez mediante un mecanismo de confirmación.

MQTT vs HTTP

MQTT es un protocolo de mensajería M2M (Máquina a máquina, del inglés Machine-to-mahcine) basado en estándares, que se usa para la comunicación de un equipo a otro. A diferencia de HTTP, en MQTT la conexión queda abierta y se "reutiliza" en cada comunicación. Por esta razón se ha convertido en la principal opción al momento de trabajar con dispositivos IoT (Amazon, s/f).

HTTP (Protocolo de Transferencia de Hipertexto, del inglés *Hypertext Transfer Protocol*) es un protocolo cliente/servidor orientado a transacciones. El uso más recurrente de HTTP es entre un navegador y un servidor Web. Por razones de seguridad generalmente hace uso de TCP (Protocolo de control de transporte, del inglés *Transport control transport*) (Stallings, 2014).

La principal diferencia entre ambos protocolos es que MQTT se centra en la transmisión de información a bajo nivel (nivel de byte), mientras que HTTP lo hace en la transmisión de documentos. En general MQTT puede compartir pings para mantener la conexión abierta, a diferencia, HTTP solo establece una conexión cuando necesita enviar una petición. Esto causa que HTTP tenga una conexión TCP half-duplex, mientras que en MQTT es full-duplex, siendo esta última la mejor opción para trabajar en dispositivos IoT.

Capítulo III: Recursos y Materiales

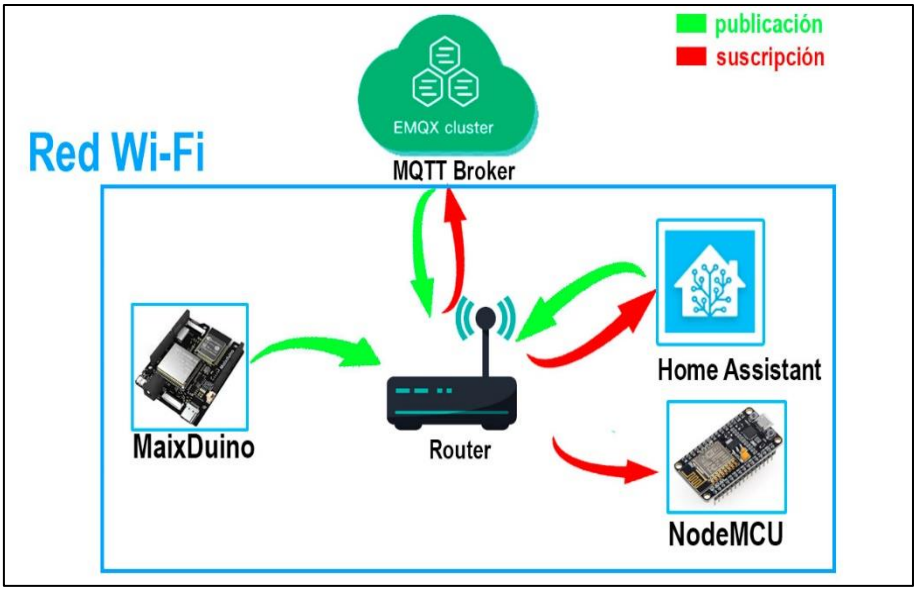
Este capítulo se centra en los recursos y materiales utilizados durante el proyecto. La elección y el uso adecuado de recursos y materiales desempeña un papel crucial en la obtención de resultados confiables y significativos. En esta sección, se presenta una descripción detallada de los recursos físicos, tecnológicos y humanos empleados a lo largo del proyecto, así como las justificaciones detrás de estas selecciones.

Antes de describir los recursos utilizados se detallará la arquitectura del sistema, con el fin de poner en contexto en donde y como se van a usar cada uno de ellos.

Arquitectura del sistema

Figura 3

Arquitectura del proyecto



Para comprender de mejor manera el sistema, se estudiará su arquitectura con la ayuda de la **Figura 3**. Por lo tanto, como primer punto se establece el uso de una red Wi-Fi, a la cual se van a conectar todos los dispositivos. Por otra parte, a través del protocolo MQTT se realizará la comunicación entre todos ellos con el objetivo de aprovechar todas las ventajas que este protocolo ofrece dentro del campo IoT, como se detalló anteriormente.

El sistema comienza en el computador embebido, el cual actuará como un sensor dentro del contexto de hogar inteligente, el cual, a través de su programación, podrá determinar si existe o no un reconocimiento correcto de las personas enroladas en el sistema. De ser correcto el reconocimiento esta placa publicará un mensaje al Broker usado, el cual lo distribuirá a los respectivos suscriptores del tema. De esta manera, por ejemplo, cuando el NodeMCU, escuche un mensaje relacionado a un tema de interés, podrá realizar las acciones detalladas en su programación. Otro ejemplo, es el caso de Home Assistant el cual al momento de escuchar algo relacionado a un tema en específico podrá hacer uso de sus automatizaciones y por lo tanto gestionar cualquier dispositivo que se encuentre conectado a su interfaz.

Tarjeta de Desarrollo MaixDuino

MaixDuino es una placa de desarrollo basada en el microcontrolador RISC-V MAIX, que combina el poder del procesador K210 y el ecosistema de Arduino. El K210 es un microcontrolador de alto rendimiento y bajo consumo de energía que integra una unidad de procesamiento de inteligencia artificial (IA) capaz de realizar tareas de visión por computadora y reconocimiento de voz.

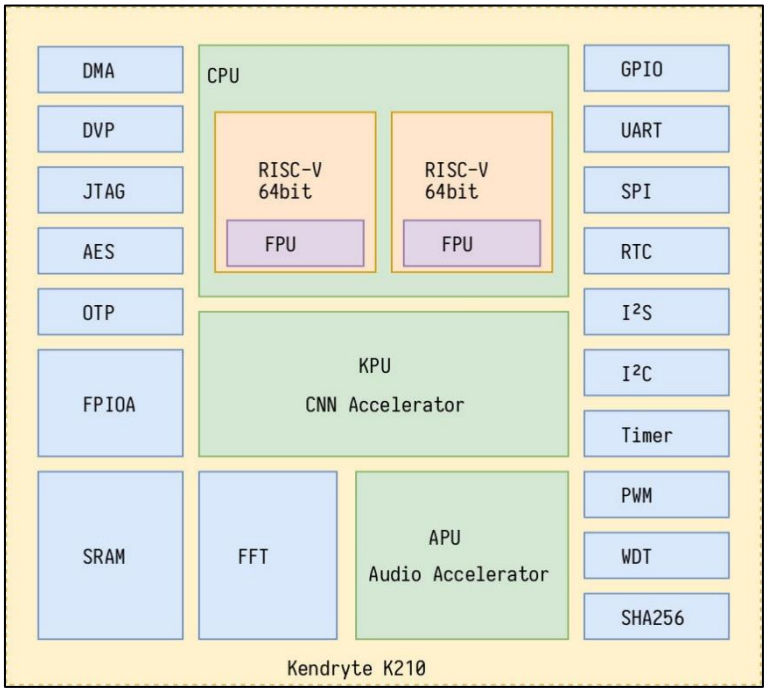
Es importante mencionar que, si se desea estudiar más acerca del hardware de la tarjeta, en la página oficial de Sipeed se encuentra el HDK (Kit de distribución de Hardware, del inglés *Hardware Distribution Kit*), específicamente en (Sipeed 4, 2023).

Chip K210

El K210 fue lanzado en el año 2018 por Canaan. Cuenta con un acelerador de hardware de red neuronal KPU de desarrollo propio que puede realizar operaciones de red neuronal convolucional con alto rendimiento. En la **Figura 4** y en la se puede observar la arquitectura de este chip y en la **Tabla 1** la descripción de sus principales especificaciones. Además, en la página oficial de Canaan se encuentran todos los recursos necesarios para los desarrolladores como: SDK (Kit de distribución de Software, del inglés *Software Distribution Kit*), o el HDK la documentación oficial del chip, entre otros.

Figura 4

Arquitectura del chip K210



Nota. Se detallan la arquitectura y todos los componentes que posee el chip K210

Tabla 1

Especificaciones del K210

	Especificaciones
Rendimiento	Procesador: Reconocimiento de imagen: Reconocimiento de voz:
Seguridad	Acelerador avanzado de encriptación hardware (AES) Memoria (OTP) SHA256
Consumo de potencia	Consumo de potencia del Chip <300mW
Escalabilidad	Sistema Operativo: FreeRTOS Modelo IA: TinyYOLOv2 Estructura Deep Learning: TensorFlow/Keras/Darknet Periféricos: FPIOA, UART, GPIO, SPI.

Nota. Se detallan la arquitectura y todos los componentes que posee el chip K210 (Canaan, 2018).

Arquitectura RISC-V

La arquitectura RISC-V es una arquitectura de conjunto de instrucciones reducido (RISC) de código abierto que se utiliza en el diseño de microprocesadores y microcontroladores. A diferencia de las arquitecturas de CPU propietarias, como x86 o ARM, RISC-V está disponible públicamente y se rige por una especificación abierta y libre de regalías.

ISA. La arquitectura del conjunto de instrucciones (ISA) de RISC-V y las especificaciones relacionadas son desarrolladas, ratificadas y mantenidas por los miembros contribuyentes de RISC-V International dentro varios grupos de trabajo internacionales de RISC-V. Para conocer de mejor forma la ISA se puede acceder su página oficial (RISC-V, 2019), donde se encuentran los volúmenes 1 y 2 de las especificaciones ISA de la propia empresa.

Ventajas. Los procesadores que usan esta tecnología poseen un mejor rendimiento debido a la cantidad limitada de instrucciones que soportan, razón por la cual requieren de un hardware físico reducido. Además, tienen poco consumo energético y generación de calor, lo cual permite que sean ideales para dispositivos móviles.

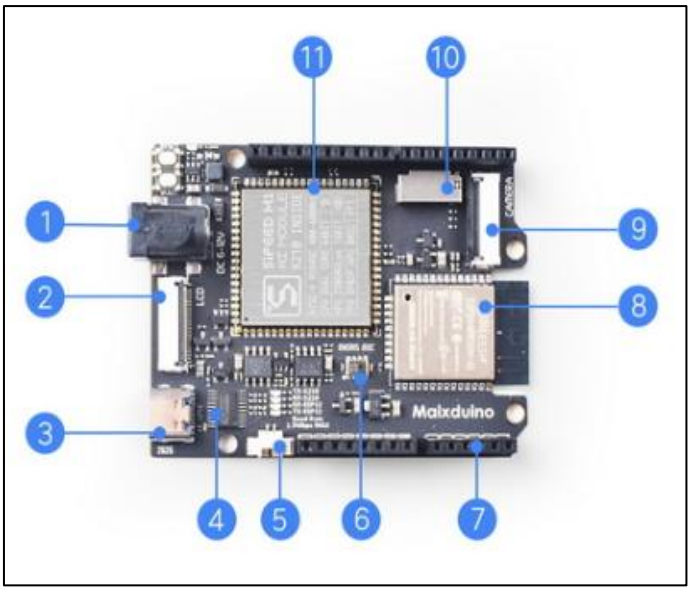
Desventajas. Entre los principales inconvenientes esta su rendimiento variable ya que depende directamente del software que se ejecute. De igual manera estos procesadores necesitan de memoria una memoria caché para que puedan realizar las instrucciones asignadas en poco tiempo.

Arquitectura física de la tarjeta

Como se puede ver en la **Figura 5** y su respectivo complemento en la **Tabla *2***, esta tarjeta de desarrollo cuenta con varias partes de hardware que le permiten realizar las tareas indicadas anteriormente.

Figura 5

Arquitectura de la tarjeta de Desarrollo MaixDuino



Nota. Esta figura muestra las partes de la tarjeta MaixDuino, posteriormente detalladas en la **Tabla *2***.

Tomado de (SiPeed 1, s/f)

Tabla 2

Detalle de las partes de la Tarjeta de Desarrollo MaixDuino

Parte	Descripción
1	Entrada DC
2	Conector a Pantalla LCD
3	Conexión USB Tipo C
4	CH552 (USB TTL)
5	Conector de parlante de 1.25mm
6	Audio DAC+PA (3W)
7	GPIOs
8	Módulo ESP32
9	Conector para cámara de 24p
10	Ranura para tarjeta TF
11	Módulo SiPeed M1

Nota. Esta tabla muestra la descripción de las partes numeradas en la **Figura 5**

Características

Tabla 3

Detalle de las partes de la tarjeta de desarrollo MaixDuino

Nombre	Descripción
CPU	Dual-core 64bit RISC-V / 400 MHz
Memoria	8MiB 64bit on-chip SRAM
Almacenamiento	16MiB Flash, soporta expansión micro SDXC (max 128GB)
Pantalla	2.4 pulgadas TFT, Resolución: 320*240
Cámara	Cámara pixel GC0328
DVP	Interfaz DVP de 24 pines para cámara estándar
Alimentación + USB	Interfaz USB Tipo C
ESP32	Conexión SPI ESP32 (Soporta Wi-Fi y Bluetooth)
DAC	I2C DAC
Ranura de tarjeta TF	Expansión de recursos multimedia, cuenta con gran capacidad de almacenamiento

Nota. Esta tabla muestra el detalle de cada una de las características de la tarjeta MaixDuino, Recuperado de (SiPeed 1, s/f)

Almacenamiento

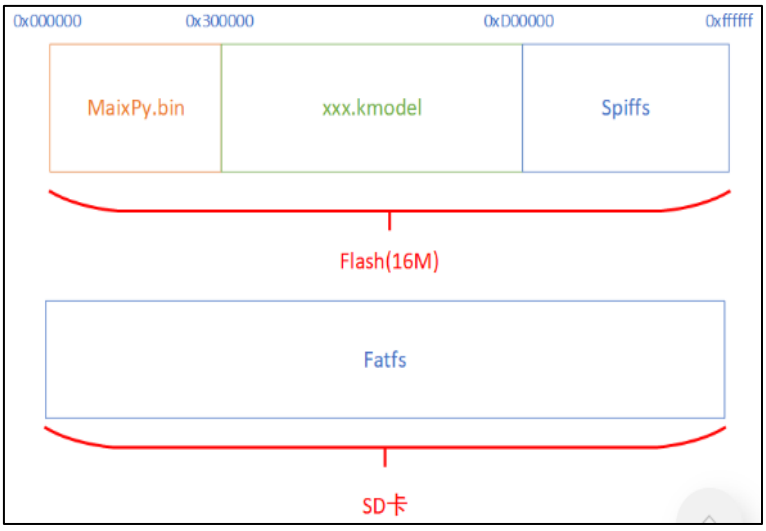
Como se puede observar en la **Figura 6**, el sistema de almacenamiento de MaixDuino se compone por una parte interna Flash y la tarjeta SD principalmente, a su vez la memoria interna se divide en tres partes, detalladas a continuación:

- **MaixPy.bin.** Se usa para almacenar el firmware, comienza en la dirección 0x000000, debido a que el chip K210 empezará a ejecutar desde allí.
- **Xxx.kmodel.** Generalmente comienza en 0x300000. Es aquí donde los archivos *model* se deben grabar. Como no existe un gestor de almacenamiento es necesario operar los archivos desde su dirección inicial.

- **Spiffs.** Usualmente comienza desde la dirección 0xD00000, esta área está gestionada por el sistema de archivos. Este sistema de archivos provee interfaces para leer y escribir archivos a través de su nombre en lugar de su dirección inicial.

Figura 6

Sistema de almacenamiento de la MaixDuino



Nota. Se detalla la capacidad de cada una de las partes que conforman el almacenamiento de la MaixDuino. Tomado de (SiPeed , s/f)

NodeMCU ESP8266 v2

El NodeMCU ESP8266 es una placa de desarrollo muy popular para el IoT. Está basada en el microcontrolador ESP8266 de *Express Systems*, que es un SoC (Sistema en un chip, del inglés *System of Chip*) con capacidad Wi-Fi integrada. Esta placa ofrece una solución completa y de bajo costo para proyectos de IoT, prototipado y desarrollo rápido de aplicaciones conectadas a la red. Generalmente se trabaja con el IDE (Entorno integrado de desarrollo, del inglés *Integrated Development Environment*) de Arduino, sin embargo, también puede ser programado a través de MicroPython o la plataforma ESPHome.

Además, hay que considerar las diferencias entre sus versiones (v1, v2 y v3), pues según su versión tiene diferentes características. Por ejemplo, una de las principales es que la versión v2 monta un chip ESP12E en lugar del ESP12 (de la v1), por lo tanto, posee más pines disponibles que el

modelo original. Además, a diferencia de las versiones v1 y v3, la v2 es más estrecha tapando sólo 8 hileras de un *protoboard*, dejando una hilera adicional a cada lado para realizar conexiones, siendo esta, la razón principal por la cual se la eligió para este trabajo (No, 2016).

Home Assistant

Es un software gratuito y de código abierto para la automatización del hogar. Permite crear un entorno centralizado para integrar y administrar una amplia gama de dispositivos inteligentes, como luces, termostatos, cerraduras, cámaras y más, independientemente de la marca o protocolo que utilicen, de manera que se puedan gestionar de forma más sencilla y con integraciones como asistentes de voz u otros. Además, está enfocado en la privacidad del usuario (Home Assistant, s/f).

Instalación

Existen cuatro métodos de instalación de Home Assistant, sin embargo, recomienda usar un sistema dedicado para la correcta ejecución de este. Por lo tanto, en su página oficial sugiere usar uno de los dos siguientes métodos:

- **Sistema Operativo Home Assistant.** Es un sistema operativo optimizado exclusivamente para Home Assistant. Es el método más recomendado por la empresa desarrolladora de la plataforma, pues cuenta con acceso a todas las opciones configurables del software.
- **Contenedor Home Assistant.** Es un contenedor independiente para Home Assistant Core, es la segunda opción más viable, solo después de la anterior nombrada.

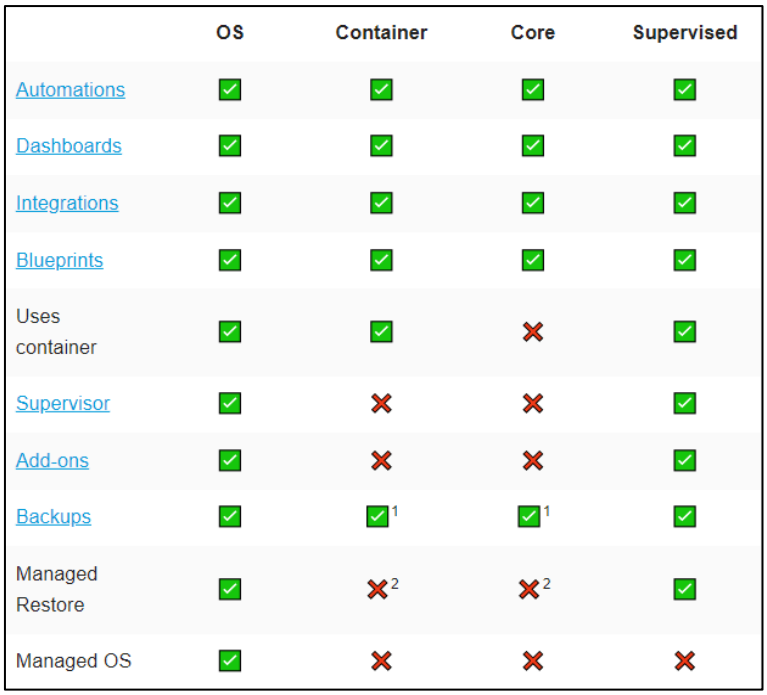
Sin embargo, también existen dos métodos de instalación alternativos disponibles, aunque sugeridos para usuarios más experimentados, estos son:

- **Home Assistant Supervised.** Instalación manual del Supervisor.
- **Home Assistant Core.** Instalación manual utilizando un entorno virtual de Python.

A continuación, se puede observar en la **Figura 7** una comparación entre los distintos métodos.

Figura 7

Comparación de métodos de instalación de Home Assistant



	OS	Container	Core	Supervised
Automations	✓	✓	✓	✓
Dashboards	✓	✓	✓	✓
Integrations	✓	✓	✓	✓
Blueprints	✓	✓	✓	✓
Uses container	✓	✓	✗	✓
Supervisor	✓	✗	✗	✓
Add-ons	✓	✗	✗	✓
Backups	✓	✓[1]	✓[1]	✓
Managed Restore	✓	✗[2]	✗[2]	✓
Managed OS	✓	✗	✗	✗

Nota. Se establecen las ventajas y desventajas de los distintos tipos de instalación de HA. Tomado de (Home Assistant, s/f)

ESPHome

Es un sistema para controlar dispositivos ESP8266/ESP32 a través de archivos YAML simples y poderosos. Siendo esta una de las mejores opciones para controlarlos remotamente en sistemas de Hogares Inteligentes. Se ha optado por su uso en el presente proyecto porque posee una integración directa con Home Assistant, lo que facilita el manejo y programación del NodeMCU ESP8266

MaixPy

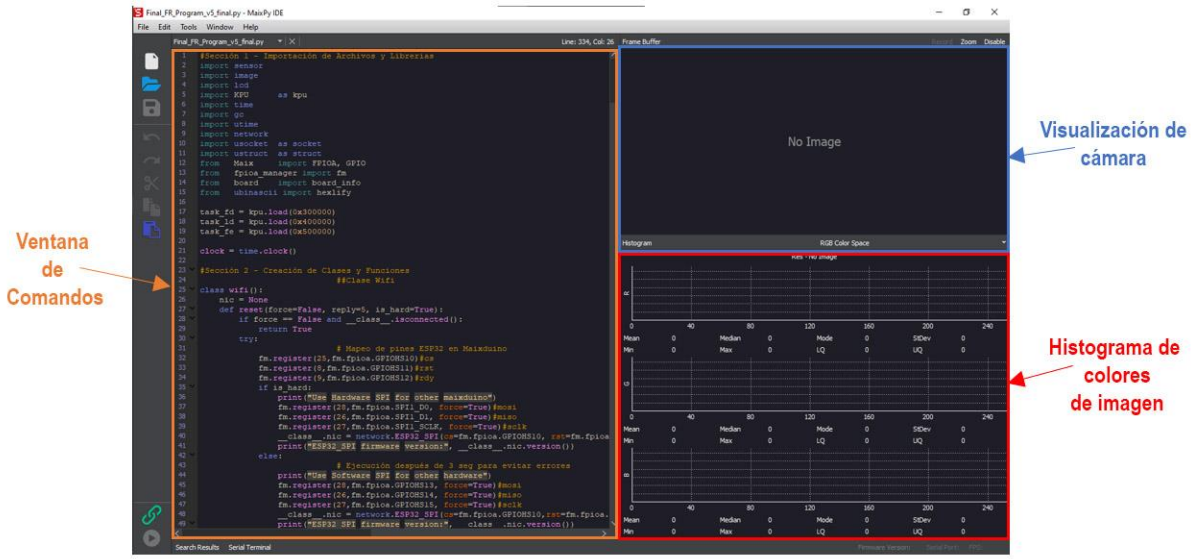
Es un proyecto que tiene como objetivo facilitar el desarrollo de aplicaciones en el procesador K210 (procesador con el que cuenta la tarjeta MaixDuino) y otros MCU (Unidades de microcontrolador, del inglés *microcontroller unit*).

Se puede considerar como un IDE de MicroPython que, además, cuenta con varias librerías especialmente diseñadas para el trabajo con periféricos en entrada y salida de la tarjeta, razón por la cual se ha escogido como principal opción para trabajar en el desarrollo del presente trabajo. Cabe

destacar que es un software multiplataforma, que puede trabajar en sistemas operativos como: Windows, MacOs y Linux y que cuenta con una amplia retroalimentación de sus colaboradores, pues su código fuente es abierto y se encuentra disponible en GitHub (SiPeed 2, s/f). En la **Figura 8** se presenta la interfaz de usuario que posee este programa.

Figura 8

Interfaz de MaixPy



Nota. Se detalla las secciones de interfaz gráfica de MaixPy.

MicroPython

MicroPython es una implementación eficiente del lenguaje de programación Python3 y que está optimizada para poder ejecutarse en un microcontrolador. Además, incluye una selección de bibliotecas fundamentales de Python y módulos que permiten al programador el acceso al hardware en bajo nivel (George, 2014).

Blender

Es un software de modelado 3D, renderización y animación por computadora de código abierto y gratuito. Es ampliamente utilizado en industrias como la animación, los efectos visuales, los videojuegos, la arquitectura y la producción de medios para crear contenido visualmente atractivo y

realista (Flavell, 2011). Para el presente trabajo se decidió usar este programa por su facilidad al momento de realizar una renderización de archivos *".stl"* y editar modelos 3D.

Capítulo IV: Implementación del sistema de control de acceso

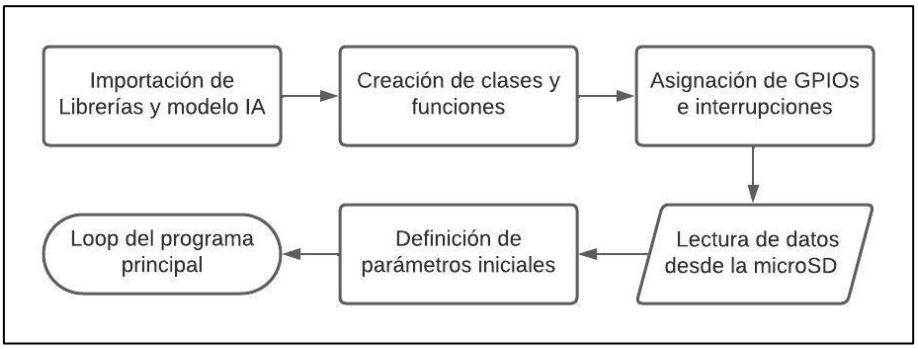
En el presente capítulo se explica cada uno de los procedimientos con los que se diseñó e implementó del sistema. Al final del capítulo se explica el código y funcionamiento del programa de software usado para la comparación con el sistema de la MaixDuino a fin de cumplir con el cuarto objetivo planteado en este trabajo.

Programa usado en MaixDuino

Para obtener un mejor entendimiento del programa, su código se dividió en seis secciones, cada una con diferentes objetivos y funciones. Como se puede observar en la **Figura 9** el proyecto ejecuta cada una de ellas de forma secuencial. A continuación, se detallan las mismas, recalcando que todo el programa fue escrito en lenguaje MicroPython.

Figura 9

Diagrama de flujo del programa



Nota. Se muestra el diagrama de flujo con cada una de las secciones en las que se divide el mismo, para obtener un mejor entendimiento del proyecto.

Sección 1

Esta sección se encarga de la importación de librerías y del modelo IA que se va a ocupar. En primer lugar, se trata de importar solo las librerías necesarias ya que es una práctica recomendada en Python, pues promueve la eficiencia, la legibilidad y la estabilidad del código. Además, ayuda a reducir la sobrecarga de memoria y a crear programas más organizados y fáciles de mantener. Como se observa en la **Figura 10** cada una de estas librerías se usa para gestionar ciertas partes específicas del hardware o software del proyecto.

Figura 10

Sección 1 del código del programa

```
#Sección 1 - Importación de Librerías y modelo IA
import sensor                # Cámara
import lcd                   # Pantalla
import image
import gc
import utime                 # Tiempo
import time
import network               # Módulo ESP32
import usocket  as socket    # MQTT
import ustruct  as struct
import KPU      as kpu       # Carga del modelo IA
from   Maix     import FPIOA, GPIO  # GPIOs
from   fpioa_manager import fm
from   board    import board_info   # Pines de la MaixDUino
from   ubinascii import hexlify
                             # Cargar el modelo IA en memoria xxx.kmodel
task_fd = kpu.load(0x300000)
task_ld = kpu.load(0x400000)
task_fe = kpu.load(0x500000)
                             # Inicio de reloj
clock = time.clock()

#Sección 2 - Creación de Clases y Funciones
                             ##Clase Wifi
```

Nota. Se muestra la importación de todas las librerías necesarias para el funcionamiento del sistema.

Carga del modelo IA. Antes de seguir con los pasos para la implementación del modelo IA, es necesario el KPU de la placa.

KPU. Es un procesador de redes neuronales de propósito general que puede realizar cálculos de redes neuronales convolucionales con bajo consumo de energía. De esta manera puede obtener el tamaño, las coordenadas y los tipos de objetos detectados en momentos específicos.

Pasos para implementación del modelo. Para la implementación del modelo IA a usar se debe ejecutar el método KPU.load(offset, file_path) como se observa en la Figura 8, donde el argumento offset es el tamaño del desplazamiento en memoria flash donde se desea guardar el modelo, ejemplo 0xd00000 significa que el modelo se grabará al inicio de los 13M o 0x300000 significa la ubicación 3M de memoria flash. De la misma manera el argumento file_path representa la ubicación y el nombre del archivo en el sistema de almacenamiento, como "/sd/xxx.kmodel".

Sección 2

El objetivo principal de esta sección es crear la clases y funciones que se van a usar a lo largo del código. En la sección de apéndices se puede observar el código completo del programa en donde se divisan las dos clases creadas y sus respectivos métodos.

Sección 3

El fin de esta sección se centra en poder gestionar de manera correcta las GPIOs de la tarjeta. El detalle de cada una de estas líneas se encuentra en el tema “Gestión de las GPIOs de la MaixDuino" del presente capítulo.

Sección 4

En esta sección se realiza la lectura de datos (nombres y características de las personas enroladas) desde la tarjeta microSD, en la **Figura *11*** se detalla el código a ejecutar para obtener los resultados previstos

Figura 11

Sección 4 del código del programa

```
#Sección 4 - Lectura de Datos
record_ftrs = []                    # Vector de características faciales
names = []                          # Vector de nombres
with open("/sd/data/name1.txt", "w") as f:
    f.write('David')

for i in range(19):                 # Número de características a cargar
    name = "name"+str(i+1)          # Proceso automático de guardado en
    feat = "features"+str(i+1)
    addr="/sd/data/"+name+".txt"
    addr1="/sd/data/"+feat+".txt"
    print(addr)
    with open(addr, "r") as f:
        names.append(f.read())      # Lectura del archivo txt
    with open(addr1, "rb") as f:
        fstr=f.read()
        record_ftrs.append(fstr)    # Guradado de valores en el vector de
        caracterísitcas
        record_ftrs[i]=bytearray(record_ftrs[i])
for i in range(19):
    if names[i]=='':
        names=names[0:i]            # Guradado de nombres en su respectivo
        vector
        break
for i in range(19):
    if record_ftrs[i]==bytearray(b''):
        record_ftrs=record_ftrs[0:i]
        break
lcd.draw_string(0, 0,"Datos Almacenados",2)# Presentación en pantalla física
de nombres
names2 = []
for i in names:
    if i not in names2:
        names2.append(i)
lcdnterminal(names2)
print(record_ftrs)                  # Presentación de datos en monitor serie
rec=len(record_ftrs)
```

Nota. En esta figura se observa el código para la lectura de datos desde la microSD.

Sección 5

Con el objetivo de definir los parámetros iniciales, tal y como se puede observar en la **Figura 12**, antes de iniciar el programa principal, se deben ejecutar varias funciones, las cuales se encargan de inicializar y ajustar los periféricos y variables necesarias para el funcionamiento del sistema en general.

Figura 12

Sección 5 del código del programa

```
#Sección 5 - Definición de parámetros iniciales
lcd.init()                          # Inicialización de la pantalla LCD
sensor.reset()                      # Inicialización de la cámara
sensor.set_pixformat(sensor.RGB565) # Código de colotes
sensor.set_framesize(sensor.QVGA)   # Tamaño de la imagen
sensor.set_hmirror(1)               # Orientación de la cámara
sensor.set_vflip(0)
sensor.run(1)

anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437,
          6.92275, 6.718375, 9.01025)           # Anclas para la detección de
          rostros
dst_point = [(44, 59), (84, 59), (64, 82), (47, 105),
             (81, 105)]                         # Puntos claves del rostro
             humano
a = kpu.init_yolo2(task_fd, 0.5, 0.3, 5, anchor)
img_lcd = image.Image()                         # Creación de objeto imagen
img_face = image.Image(size=(128, 128))         # Redimensionamiento a 128x128
a = img_face.pix_to_ai()
record_ftr = []

ACCURACY = 83.5                                 # Exactitud a evaluar
(porcentaje)
WIFI_SSID = "NETLIFE-FPOGO-HABITACIONES"
WIFI_PASSWORD = "17187150pogo"
state=False                                     # Estado del nodo a gestionar
(Puerta, foco, etc)
#connect_wifi(WIFI_SSID, WIFI_PASSWORD)         # Conexión a
WiFi
lcd.draw_string(60, 100,"Bienvenidos a Smart Home SD - Booty",2)# Excritura
en Pantalla
utime.sleep_ms(1500)
lcd.draw_string(120, 140,"Wifi Conectado",2)
utime.sleep_ms(1500)
```

Nota. En la sección 5 se detalla la creación e inicialización de los parámetros que se ocuparán en el programa.

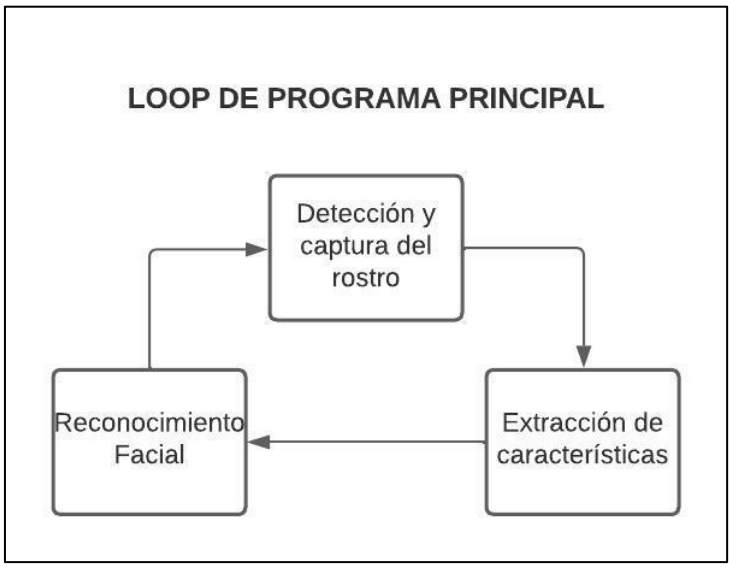
Sección 6

El programa usado para el reconocimiento facial del presente proyecto se basa en el modelo presentado por (SiPeed, n.d.). El cual cumple con las tres etapas esenciales mencionadas en el marco teórico. A continuación, se explica de forma general el funcionamiento del programa principal, el cual es un bucle como se observa en la **Figura 13**. Debido a la extensión del código a continuación se dará una breve explicación de su funcionamiento, sin embargo, si se quiere entrar en detalle del mismo se

recomienda observar el código completo del programa, ubicado en la en la sección apéndices del presente trabajo.

Figura 13

Diagrama de flujo del funcionamiento del programa principal



Nota. Se detalla el bucle en el cual se mantiene el programa principal.

Funcionamiento

Detección y captura del rostro. Para la detección el programa usa el modelo YOLO v2 (You Only Look Once), el cual realiza la detección de objetos en tiempo real con una alta precisión que permite correr en dispositivos que no son muy potentes, por lo cual es conveniente para trabajar con tarjetas de desarrollo con limitadas características de desempeño.

Extracción de características. Una vez detectado el rostro y después de dar la orden de enrolar a una persona, el programa corta la imagen y la redimensiona a un tamaño de 128x128 pixeles, luego identifica cinco puntos principales, los cuales son ambos ojos, la nariz y las comisuras de los labios, estos puntos serán tratados como valores de tipo *bytearray*, es decir representa un arreglo mutable de bytes que se guardarán en la memoria del dispositivos, junto con el nombre de la persona en cuestión, para posteriormente poder trabajar con ellos.

Identificación. Después de guardar los datos en memoria, la próxima vez que el programa detecte un rostro, rastreará automáticamente los puntos claves de este y los comparará con el banco de datos que tiene tras enrolarse varias personas. Luego de una comparación con la información de cada persona guardada se obtiene un resultado de semejanza en porcentaje, si este es mayor a un umbral establecido como mínimo, entonces se habrá identificado correctamente el rostro en cuestión. Después de decidir si el reconocimiento fue correcto, si es así, la placa mandará un mensaje de aprobación usando el protocolo MQTT.

De esta manera los dispositivos que estén escuchando (Home Assistant y NodeMCU en este caso) el tema respectivo podrán usarlo como *trigger* para realizar sus respectivas automatizaciones, sean estas encender un foco o LED, activar un relé o un dispositivo, entre otras. Después, el programa vuelve al punto de detección de rostros y así en forma de bucle.

Conexión de la MaixDuino a Wi-Fi

A continuación, en la **Figura 14** se presenta la función específica que permite la conexión de la tarjeta MaixDuino a una red Wi-Fi, sin embargo, para lograr entender su funcionamiento se debe saber que antes es necesario crear la clase “wifi()”, desarrollada en el GitHub de MaixPy y licenciada de forma no comercial por el MIT(Instituto Tecnológico de Massachusetts, del inglés *Massachusetts Institute of Technology*), pues es la encargada, gracias a sus métodos, de intervenir directamente con el módulo ESP32 de la MaixDuino (MaixPy, s/f). En primer lugar, gracias a la función *reset()* se establece el mapeo y reconexión de las IO del chip ESP32 en caso de que exista un problema. Es importante notar el uso de la función “Try - Except” con el fin de evitar errores al conectarse, pues de lo contrario es muy probable que la tarjeta de desarrollo deje de funcionar hasta su reinicio. Se puede acceder al código de esta clase en la sección 2 del código completo adjunto dentro de la sección apéndices.

Figura 14

Función que permite conectar la tarjeta a Wi-Fi

```
def connect_wifi(WIFI_SSID, WIFI_PASSWORD):
    while not wifi.isconnected():
        print("--- Reset ESP32 ---")
        wifi.reset()
        print("--- Conectando a Wi-Fi ---")
        try:
            wifi.connect(WIFI_SSID, WIFI_PASSWORD)
        except Exception as e:
            print("Conexión fallida: {}".format(e))
    print("Wi-Fi conectado a:", wifi.isconnected(), wifi.ifconfig())
```

Nota. En la Figura se muestra la función *connect_wifi(),* la cual permite la conexión de la placa a una red Wi-Fi

A través del monitor serie de MaixPy se puede comprobar la correcta conexión de la tarjeta como se muestra en la **Figura 15**.

Figura 15

Conexión de la MaixDuino a una red Wi-Fi

```
[bytearray(b'\xe5\xe7\xe1\x80!3>\xef\xc2\x00<\xea\xdd\xfe\xce\x01\xc2\xd0&\x1f\xde
--- Conectando a Wi-Fi ---
Wi-Fi conectado a: True ('192.168.100.172', '255.255.255.0', '192.168.100.1')
MQTT Conectado
```

Nota. En esta figura se aprecia la comprobación, a través del monitor serie de MaixPy, de una correcta conexión entre la MaixDuino y una red Wi-Fi.

Gestión de las GPIOs de la MaixDuino

Pasos para la configuración de GPIOs en la tarjeta

Para la gestión de las entradas y salidas de la MaixDuino se deben seguir los pasos detallados a continuación y escritos en la sección 4 del código completo. Antes de empezar a trabajar con las entradas de la tarjeta se debe realizar un paso importante, este es, configurar el hardware (distribución de pines) correspondiente a la MaixDuino, para ello se descarga el archivo *config_maix_duino.py*, disponible en (Chen, s/f) o adjunto en este documento dentro de la sección apéndices, luego se lo debe cargar en MaixPy y ejecutar por una vez, de esta manera la configuración se almacenará en la configuración flash del dispositivo y este quedará listo para su uso. Una vez

configurado el hardware correspondiente a la placa, se procede a registrar los botones (BOOT o RST), entradas o salidas que se vayan a ocupar y asignarles una constante GPIO que funcionará como identificación para su posterior funcionamiento, esto se logra a través del uso de la función *fm.registrer()*, la cual posee dos argumentos, el primero es el encargado de asignar el botó o pin físico a usar y el segundo es la asignación GPIO que se le dará al mismo.

Después de esta introducción primero se debe crear un objeto perteneciente a la clase GPIO, la cual dispone de cuatro argumentos, como se ve en la **Figura 16**, a continuación, se detalla cada uno de ellos:

Figura 16

Configuración MQTT en ESPHome

```
class GPIO(ID, MODE, PULL, VALUE)
```

Nota. Se detalla los argumentos de la clase GPIO.

- **ID.** Es el pin GPIO que se va a usar, el número se puede obtener del mapeo de pines de la tarjeta que se encuentra dentro de la sección apéndices.
- **MODE.** es el modo con el que se le va a configurar al GPIO, se puede asignar cuatro modos distintos como se muestra en la **Tabla 4**.

Tabla 4

Tipos de valores del argumento MODE

ARGUMENTO	MODO
GPIO.IN	entrada
GPIO.PULL_UP	entrada pull-up
GPIO.PULL_DOWN	entrada pull-down
GPIO.OUT	salida

Nota. Esta tabla muestra los tipos de valores que se pueden asignar al argumento MODE.

- **PULL**. este parámetro establece si se ocupará la entrada como pull.
- **VALUE.** Aquí se establece el valor que va a tomar en caso de ser una salida, puede ser 1 o 0.

Una vez una vez creados los objetos ya se pueden usar en el código, a través de sus métodos, para este trabajo en específico solo se usará "*.value()*", en el cual como argumento se establece el valor de salida que ocupará el pin correspondiente. Finalmente, también se puede establecer interrupciones usando las entradas como *triggers,* para ello se hace uso del método "*.irq(CALLBACK_FUNC, TRIGGER_CONDITION, GPIO.WAKEUP_NOT_SUPPORT,PRORITY)*", en la cual, sus argumentos establecen, en primer lugar la función a ejecutar durante la interrupción, segundo el actuador (botón, entrada o evento) que funcionará como *trigger* de activación, tercero (*GPIO.WAKEUP_NOT_SUPPORT,* por defecto) y por último el grado de prioridad que se dará a la interrupción.

Comunicación entre MaixDuino y NodeMCUv2

Para realizar el envío de información desde la MaixDuino al NodeMCU, además del constante monitoreo en Home Assistant, se optó por el uso del protocolo de comunicación MQTT (Transporte de telemetría de colas de mensajes, del inglés *Message Queing Telemetry Transport*) debido a sus ventajas sobre otras opciones como el protocolo HTTP.

Estructura de mensajes

Para el presente trabajo se desarrolló una estructura de mensajes enfocada en un hogar inteligente, por lo tanto, esta se maneja, en primer lugar, por escenas, es decir lugares del hogar como sala, cocina, dormitorios, etc. Luego le sigue el dispositivo a gestionar y por último la carga útil del mensaje, como se muestra en la **Figura 17**:

Figura 17

Estructura de los mensajes del proyecto

Nota. Se detalla la estructura de los mensajes del presente proyecto dependiendo de los temas que se vayan a usar.

Configuración MQTT en Home Assistant

Para poder gestionar y monitorear los mensajes MQTT publicados en Home Assistant, se debe descargar el complemento “Mosquito Broker” el cual tomará la función de mediador entre los dispositivos que interactúen mediante el protocolo MQTT. Para ello se lo buscará en la tienda de complementos de HA (Home Assistant), una vez seleccionado, se procede a instalarlo y además marcar la opción “Vigilancia”, esta función permite reiniciar este *add-on* en caso de que exista alguna caída o fallo en la conexión.

Por lo tanto, se usará el Broker público de la empresa EMQX, se accede a él a través de su página web, en la que se debe registrar como nuevo usuario y automáticamente podrá acceder a los datos necesarios para realizar la conexión, los mismos se muestran en la **Figura 18**.

Figura 18

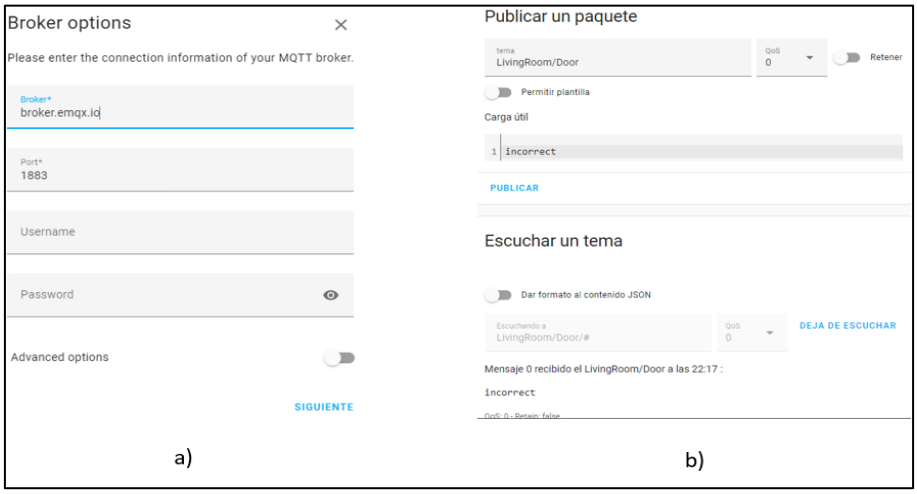
Datos del Broker EMQX

Nota. Se presenta la información detallada de los puertos y datos del Broker EMQX, para su uso dentro del proyecto. Tomado de (EMQX, s/f)

Una vez conocidos los datos se procede, en la pestaña “Configuración”, a configurar el Broker en Home Assistant, para ello se deben ingresar los mismos datos (de Broker y puertos) como se muestra en la **Figura 19** a), mostrados anteriormente, y de esta manera se establecerá la conexión y se podrá publicar y escuchar los mensajes que pasen a través del Broker configurado como se muestra en la **Figura 19** b).

Figura 19

Configuración MQTT en Home Assistant



Nota. En la parte a) se muestra la configuración del Broker MQTT en Home Assistant y en b) la comprobación si se está suscrito/publicando en un tema en específico.

Configuración MQTT para la MaixDuino

Para establecer la conexión con el Broker MQTT deseado, al igual que la conexión a Wi-Fi, se crea una clase, en esta ocasión se trata de *MQTTClient,* misma que fue programada por (Motl, s/f). Debido a la extensión de su código, si se quiere entrar en detalle, se puede acceder a él en la sección 2 del código completo, el cual se encuentra en apéndices de este documento.

Esta clase cuenta con varias funciones que nos permiten la correcta manipulación de mensajes MQTT, sea para su publicación o suscripción. Específicamente hablando en el presente proyecto se usan los métodos *“.connect()”* para la conexión a los datos de Broker establecidos y también *“.publish()”* para la publicación de mensajes al momento de reconocer un rostro.

Configuración MQTT para el NodeMCU desde ESPHome

Como se indicó en el capítulo anterior, la forma más fácil de programar el dispositivo NodeMCU ESP8266 es a través de ESPHome, con archivos YAML.

Pasos para la configuración. En primer lugar, se necesita configurar la red Wi-Fi a la cual se conectarán los dispositivos ESP y así tener una interconexión con Home Assistant. Una vez reconocido el tipo de dispositivo ESP que se tiene (NodeMCU ESP8266 v2, en este caso), se introduce una configuración inicial. Luego se configura el dispositivo dependiendo de su función, en este caso por ejemplo para una configuración de GPIOs se usa el código de la **Figura 20.** Además, se recuerda que dependiendo de los pines que se vayan a usar se comprueba su identificación en *pinout* de la placa, el cual también se encuentra en la sección apéndices**.**

Figura 20

Configuración de GPIOs en ESPHome

```
# Example configuration entry
esphome:
  name: livingroom

esp8266:
  board: nodemcuv2

binary_sensor:
  - platform: gpio
    name: "Pin GPIO17"
    pin: GPIO17
```

Nota. Aquí se detalla el código a implementar para configurar las GPIOs en el NodeMCU ESP8266 v2.

Se recuerda que el uso de archivos YAML como configuración con ESPHome también puede servir para el uso de protocolos HTTP o MQTT, como se observa en la **Figura 21**.

Figura 21

Configuración MQTT en ESPHome

```
# Simple:
some_option: topic/to/send/to

# Disable:
some_option:

# Advanced:
some_option:
  topic: topic/to/send/to
  payload: online
  qos: 0
  retain: true
```

Nota. Se detalla el código a implementar para configurar el uso del protocolo MQTT en el NodeMCU ESP8266 v2.

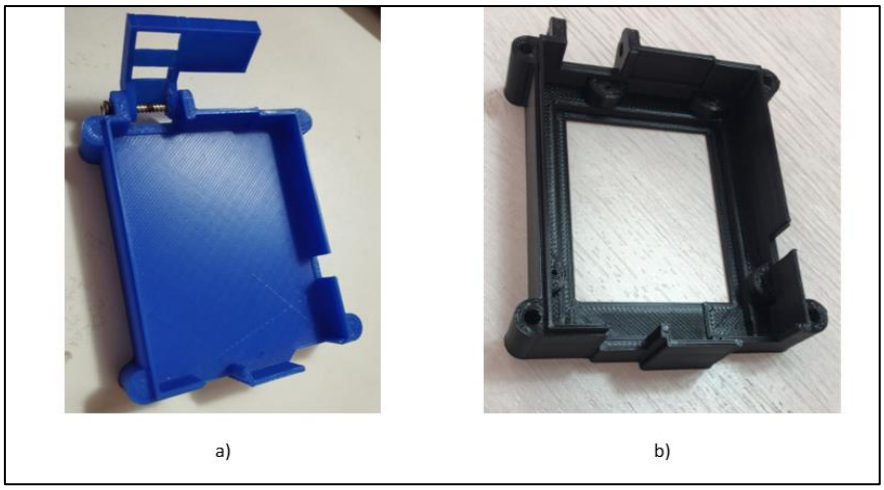
Diseño del armazón

Con el objetivo de optimizar y preservar la manipulación de la tarjeta, se recurrió al diseño de un prototipo de carcasa protectora para la tarjeta, la cual tiene una parte plástica impresa en 3D y otra diseñada en material de buena resistencia, además se estableció la ubicación de la cámara, dos diodos LEDs que indicarán si existe un reconocimiento correcto o no, un botón para el enrolamiento de las personas y un sensor de movimiento, el cual nos permitirá encender un flash o led de alta intensidad, que ayudará a mejorar las condiciones, cuando estas no sean óptimas, para un correcto funcionamiento del programa.

El armazón plástico impreso en 3D está basado en el modelo de la página (Ultimaker Thingiverse, s/f), además, para la edición de este modelo se usó la plataforma Blender con la cual se logró editar el modelo original mostrado en la **Figura 22** a) y obtener el de la **Figura 22** b)

Figura 22

Comparación de los modelos original y editado

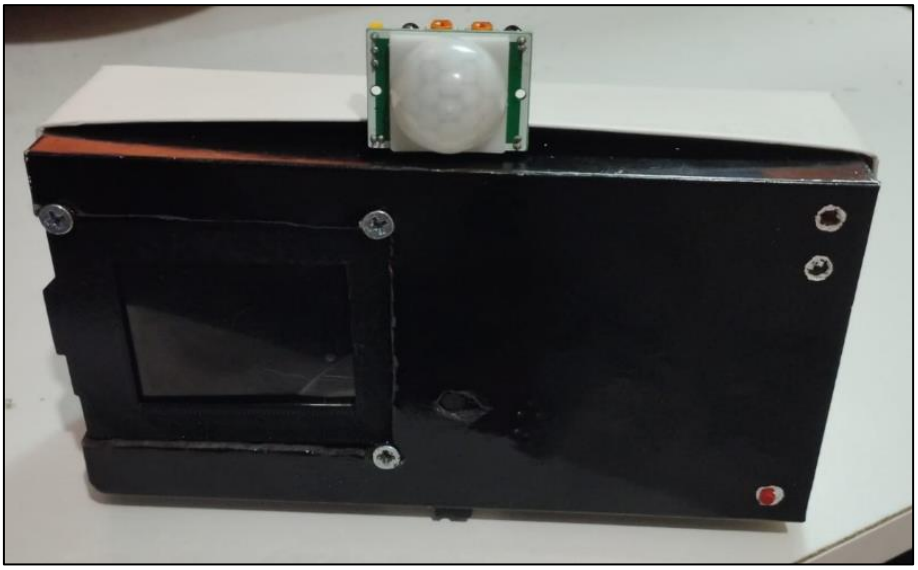


Nota. Comparativa entre el modelo original a) y el editado b)

Por su parte el modelo exterior se diseñó de manera tal que permita facilitar la manipulación de las GPIOs (Entradas y salidas de propósito general, del inglés g*eneral purpose inputs outputs*), y que además ayude a la estabilidad y protección de los componentes, como pantalla, cámara, LEDs, botones y sensor de movimiento. El diseño final se presenta en la **Figura 23**.

Figura 23

Diseño final del prototipo



Nota. Diseño final del prototipo usado

Descripción general del programa OpenCV

El programa que se usó para realizar un reconocimiento facial en software, usando la librería OpenCV, está basado en el trabajo de Solano (2020), de igual manera que el programa del computador embebido, este también cuenta con las 3 etapas esenciales del reconocimiento facial, descritas en el marco teórico, las cuales se dividen en 3 scripts distintos programados en lenguaje Python y descritos a continuación.

Detección y captura del rostro

Para la fase de detección y captura el programa que se usa es *CapturaDeRostros.py*, el cual tiene por objetivo capturar y preparar las imágenes con las cuales se van a entrenar el o los modelos a usar. Este programa usa uno de los tantos clasificadores con los que cuenta la librería OpenCV, específicamente el de detección de rostros humanos, llamado "*haarcascade_frontalface_default.xml".*

Para cumplir con el objetivo de comparar ambos sistemas de detección, hardware y software y tratando de establecer las mismas condiciones, se realizaron varias modificaciones al sistema hardware. Por ejemplo, se dispuso que el clasificador debía trabajar con un número pequeño de imágenes de entrenamiento, por lo tanto, se redujo de 300 a 8 el número de capturas. De igual forma el redimensionamiento pasaría de 150x150 a 128x128 como en el programa de la MaixDuino. Todo el código detallado se puede observar en este documento dentro de la en la sección apéndices.

Extracción de características

Una vez que se tiene el banco de datos de las personas enroladas, se procede a trabajar en el entrenamiento del programa. El *script* usado, *EntrenamientoRF.py*, busca las imágenes almacenadas, como ya están redimensionadas entonces se prosigue directamente con el entrenamiento, haciendo uso del método *face.EigenFaceRecognizer_create()* y la función *face_recognizer.train()* con los datos, imágenes y nombres, almacenados previamente. Finalmente se guarda el modelo en un archivo

"*.xml*" con el cual nos evitamos volver a realizar un entrenamiento nuevo a menos que se enrole una persona nueva. Todo el código se encuentra detallado en la dentro de los apéndices.

Identificación

Como etapa final se tiene el reconocimiento facial en vivo. Para ello, en primer lugar, se debe leer el modelo entrenado previamente, esto se logra con la función *face_recognizer,read().* Después comienza un bucle en el cual se identificará y capturará el o los rostros presentes en un video o en una webcam, en caso de trabajar con imágenes en vivo, del ordenador, gracias a la función *cv2.VideoCapture(0,cv2.CAP_DSHOW)*. Posteriormente, con la ayuda de la función *face_recognizer,predict()*, se obtendrá una etiqueta y su respectivo valor de confianza asociado (la semejanza existente entre las imágenes capturadas y las guardadas por el modelo). Posteriormente el programa comparará este valor de confianza con respecto a un umbral, al igual que el programa de la MaixDuino y de esta manera establecerá si se trata de una persona enrolada o no. El código completo de este script se encuentra en la en la sección apéndices.

Capítulo V: Experimentación

En este capítulo se detallan las condiciones, pruebas y experimentos realizados para obtener los resultados.

Condiciones de detección

Para la realización de las pruebas primero se definieron las condiciones óptimas y mínimas con las cuales el modelo podría trabajar correctamente. Para lo cual se validó los siguientes parámetros:

Distancia

La distancia máxima a la cual el rostro puede estar alejado de la cámara para que exista un buen reconocimiento se definió en función del tamaño de imagen con el cual trabaja la etapa de extracción de características (128 x 128 pixeles). Esta distancia es aproximadamente 1.2 metros. Por otro lado, para determinar la distancia mínima se realizaron pruebas y se estableció que la distancia mínima es de aproximadamente 40 centímetros, pues al acercarse más la fase de detección comienza a fallar, estableciendo los puntos clave del rostro en otros lugares erróneos. En la **Figura 24** se muestra aproximadamente la distancia mínima y máxima para que exista una correcta detección

Figura 24

Distancia mínima y máxima para que exista detección del rostro

Nota. En la figura se puede observar a la izquierda la distancia mínima y a la derecha la distancia máxima establecida

Intensidad de luz

Para establecer la cantidad de luz mínima a la cual existe una detección de rostro se trabajó con una aplicación móvil de luxómetro en un smartphone, específicamente Light Meter – Lux Meter de la empresa Coolexp. Debido a las características del programa y el hardware usado (una cámara VGA), es importante señalar que entre mayor sea la cantidad de luz el reconocimiento trabaja de mejor manera.

Ambientes Cerrados. Se requieren al menos 12 luxes para que exista una detección, por consiguiente y para obtener unas condiciones óptimas se necesita estar a más de 120 luxes, tal y como se puede ver en la **Figura 25**.

Figura 25

Condiciones de iluminación en ambientes cerrados

Nota. Se detallan los valores óptimos (izquierda) y mínimos (derecha) de iluminación para que exista una detección.

Ambientes exteriores. La gran parte de ambientes exteriores poseen al menos 1000 luxes en el día, de manera que no existe problema alguno con la detección al tratarse de ambientes con condiciones óptimas. En la noche, estos ambientes se tratarán como ambientes internos, y en ese caso será necesario la ayuda de un foco o flash, es decir iluminación artificial, para que exista una detección.

Ángulos de visión

Debido al funcionamiento de la etapa de reconocimiento del programa, los ángulos a los cuales el rostro debe estar se definen al momento de enrolar a la persona, puesto que con un ligero

cambio de ángulo del rostro ya no existe un buen reconocimiento, por lo cual se concluye que la persona debe estar perfectamente alineada al frente de la cámara a la hora de ser enrolada y reconocida.

Primer experimento

Condiciones

El primer experimento se realizó con 12 personas, de las cuales únicamente 8 fueron enroladas en el sistema. Se estableció que las condiciones tienen que ser óptimas: todas las personas debían encontrarse a una distancia de 50cm aproximadamente y el ambiente es exterior (luz natural), con una luminosidad de más de 1500 luxes. Las personas se presentaron frente a la cámara de la misma manera en la que fueron enroladas, es decir sin realizar gestos o muecas que puedan alterar sus datos biométricos. Finalmente, en un caso real el sistema de control de acceso de un hogar permanecerá en el mismo lugar durante todo el experimento.

Para dar como válido un reconocimiento, después del respectivo enrolamiento, se estableció que todas las personas pasen 3 veces por el sistema de forma aleatoria y permanecer frente a la cámara por al menos 5 segundos, acercándose o alejándose dentro de los parámetros permitidos.

Pruebas

En la **Figura 26** se visualiza la realización del primer experimento, como se puede observar las condiciones son las establecidas anteriormente. Por esta razón los resultados fueron óptimos, existiendo un reconocimiento correcto de las 8 personas que habían sido enroladas. En el siguiente capítulo se tratarán con mayor amplitud los resultados obtenidos.

Figura 26

Primer experimento

Nota. En la Figura se pueden observar las condiciones descritas para la realización del primer experimento.

Segundo experimento

Condiciones

Con el objetivo de simular un escenario de control de acceso de un hogar inteligente, en el segundo experimento se trabajó con 22 personas de las cuales se enroló solo a 6 que serían los miembros de una familia. Al igual que en el primer experimento el sistema siempre permaneció en el mismo lugar y se procuró que las personas no realicen gestos que puedan alterar los rasgos biométricos de sus rostros. En esta ocasión el enrolamiento de una persona se lo realizó 3 veces con lo que se obtuvo 3 imágenes de la persona con distintos niveles de iluminación. Se trabajó con luz natural, luz natural atenuada y con luz artificial, para ello, las pruebas se realizaron dentro de una casa usando distintos tipos de cortinas e iluminación.

Finalmente, para dar como válido un reconocimiento, se estableció que todas las personas pasen 4 veces por el sistema de forma aleatoria, y permanecer ahí por al menos 5 segundos, acercándose o alejándose dentro de los parámetros permitidos.

Pruebas

En la **Figura 27** se observa el momento y las condiciones bajo las cuales se realizaron las pruebas de este segundo experimento. Por medio de un enrolamiento bajo diferentes condiciones de iluminación, se obtuvo un buen resultado, pues al igual que el primer experimento existió un buen reconocimiento de las personas enroladas, como se podrá observar en el capítulo VI.

Figura 27

Segundo Experimento

Nota. En la Figura se pueden observar las condiciones descritas para la realización del segundo experimento, es decir, diferentes tipos de iluminación.

Tercer experimento

Condiciones

El tercer experimento responde al cuarto objetivo específico del presente proyecto, que se trata de la comparación del reconocimiento facial hardware vs software. Para llevarlo a cabo se estableció una iluminación natural o artificial que se encuentre por arriba de los 100 luxes y una distancia máxima de 1.50m.

Se contó con la participación de 5 personas de las cuales se enroló a 3. Para tratar de equiparar las circunstancias de los modelos, en ambos casos se trabajó con 8 capturas de rostro por persona en la etapa de extracción de características, este número de capturas permitió mejorar la robustez del sistema de la MaixDuino, la iluminación fue variante durante el enrolamiento, procurando establecer diferentes condiciones de luz que ayudarán a tener un banco de datos variado con el que se puedan entrenar los modelos.

De manera análoga a los dos experimentos previos, para validar un reconocimiento se estableció que todas las personas deberían presentarse 5 veces de forma aleatoria por un máximo de 5 segundos, acercándose o alejándose del sistema.

Pruebas

Durante las pruebas se trabajó con variantes condiciones de iluminación, procurando un ambiente real. De esta manera, en ambos casos, el reconocimiento fue perfecto y por lo tanto se puede decir que los resultados son equiparables en cuanto a efectividad. Sin embargo, un dato adicional a tomar en cuenta es que el programa de software puede trabajar con diferentes rostros al mismo tiempo algo que en el modelo de la tarjeta MaixDuino no se pudo hacer, como se puede observar en la **Figura 28**.

Figura 28

Tercer Experimento

Nota. En la Figura se pueden observar el funcionamiento del programa en a) software y b) hardware con tres rostros al mismo tiempo

Capítulo VI: Resultados

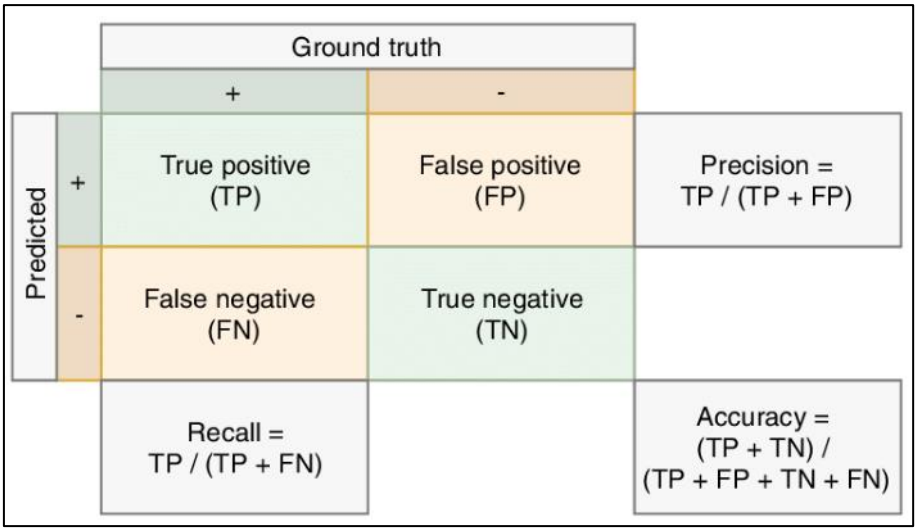
En el presente capítulo se detallan los resultados obtenidos luego de las pruebas realizadas con su análisis correspondiente. Para ello primero se visualiza la forma de evaluar los resultados, los cuales se darán a través de la presentación de matrices de confusión.

Matriz de confusión

Una matriz de confusión es una herramienta utilizada en el campo de la estadística y el aprendizaje automático para evaluar el rendimiento de un modelo de clasificación. Se utiliza para visualizar el rendimiento de un algoritmo de clasificación comparando las predicciones del modelo con los valores reales de las etiquetas de clase. Es una manera estándar de mostrar la cantidad de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN) de manera más visual. Para una clasificación de dos clases (también conocida como clasificación binaria), la matriz de confusión es una tabla de 2 × 2 que tiene la forma mostrada en la **Figura 29** (Skansi, 2018).

Figura 29

Parámetros de la matriz de confusión



Nota. La figura muestra la forma que posee una matriz de confusión y sus respectivos parámetros y métricas. Tomado de (Jeppesen et al., 2019)

Parámetros

Según Kulkarni et al. (2020). Los parámetros de una matriz de confusión se pueden observar en la Figura 26, estos son:

- **Verdaderos Positivos (True Positives - TP).** Son los casos en los que el modelo predice correctamente que una instancia pertenece a una clase específica, y esa instancia realmente pertenece a esa clase.
- **Falsos Positivos (False Positives - FP).** Son los casos en los que el modelo predice incorrectamente que una instancia pertenece a una clase específica, pero en realidad esa instancia no pertenece a esa clase.
- **Verdaderos Negativos (True Negatives - TN).** Son los casos en los que el modelo predice correctamente que una instancia no pertenece a una clase específica, y esa instancia realmente no pertenece a esa clase.
- **Falsos Negativos (False Negatives - FN).** Son los casos en los que el modelo predice incorrectamente que una instancia no pertenece a una clase específica, pero en realidad esa instancia sí pertenece a esa clase.

Métricas

Las métricas que se derivan de una matriz de confusión son útiles para evaluar el rendimiento de un modelo de clasificación (Skansi, 2018). En la **Figura 29** se puede obtener una mejor visualización de las métricas que se describirán a continuación:

- **Precisión (Accuracy).** Mide la proporción de predicciones correctas en relación con todas las predicciones. Se calcula como: (TP + TN) / (TP + TN + FP + FN).
- **Sensibilidad o Recall0**. Mide la capacidad del modelo para identificar correctamente las instancias positivas. Se calcula como: TP / (TP + FN).

- **Valor Predictivo Positivo (Positive Predictive Value, PPV) o Precisión.** Mide la proporción de instancias positivas predichas correctamente con respecto a todas las instancias predichas como positivas. Se calcula como: TP / (TP + FP).

Uso en el presente proyecto

La razón principal para el uso de las matrices de confusión en el presente trabajo es que es una manera práctica de visualizar los resultados de un clasificador binario. De esta manera se puede hacer un símil entre un sistema clasificador de Machine Learning con el sistema de reconocimiento facial que se ha diseñado. Es decir, se tratarán los resultados del reconocimiento (correcto o incorrecto) como si fueran los de un clasificador binario. Por lo tanto, se podrá evaluar a través de su parámetros y métricas la efectividad del sistema diseñado.

Primer experimento

Figura 30

Matriz de confusión de los resultados del primer experimento

		Valor Real		
		+	-	
Valor estimado	+	23 (TP)	0 (FP)	100% 0%
	-	1 (FN)	11 (TN)	92% 8%
		96% 4%	100% 0%	97% 3%

Nota. Esta figura muestra los resultados obtenidos después de realizar las primeras pruebas.

Análisis

Como se puede observar en la **Figura 30**, para el primer experimento se obtuvo un rendimiento general del sistema del 97%, existiendo un reconocimiento muy bueno (reconociendo

las 8 personas enroladas al menos dos de los tres intentos) y solo un falso negativo en el experimento. Además, cuenta con una sensibilidad del 96%. Para aplicaciones de seguridad, uno de los parámetros más importantes que se tiene que reducir en lo posible es la cantidad de falsos positivos, es decir, en la práctica se trata de evitar que el sistema reconozca a personas desconocidas como pertenecientes al hogar.

Segundo experimento

Figura 31

Matriz de confusión de los resultados del segundo experimento

		Valor Real		
		+	-	
Valor estimado	+	19 (TP)	0 (FP)	100% 0%
	-	5 (FN)	64 (TN)	93% 7%
		79% 21%	100% 0%	94% 6%

Nota. Esta figura muestra los resultados obtenidos después de realizar el segundo experimento pruebas.

Análisis

En la **Figura 31** se puede observar las métricas de evaluación del segundo experimento, en este caso el rendimiento general del sistema fue del 94%. La sensibilidad cayó a un 79% pero manteniendo el número de falsos positivos en cero. Por lo cual el reconocimiento de las 6 personas enroladas fue bueno, aunque no óptimo, recordando que constantemente se cambiaron las condiciones de iluminación en el experimento.

Tercer experimento

Figura 32

Matriz de confusión de los resultados del tercer experimento

Nota. Esta figura muestra los resultados obtenidos después de realizar el tercer experimento a) Hardware y b) Software.

Análisis

Durante la comparación de ambos mecanismos se pudo comprobar la robustez que tienen, lo cual depende mucho de la cantidad de imágenes que se usen para entrenar los modelos individuales de cada uno, como se observa en la **Figura 32** con 8 imágenes de entrenamiento en ambos sistemas se obtuvo un reconocimiento perfecto.

Capítulo VII: Conclusiones y Trabajos Futuros

Conclusiones

MaixDuino es una placa de desarrollo basada en el chip Kendryte K210, que es conocido por su capacidad de ejecutar modelos de aprendizaje profundo y realizar tareas relacionadas con la IA en el dispositivo. Es una herramienta versátil que, por las características descritas en este texto y su precio, es una de las mejores opciones para el desarrollo de este proyecto.

Se realizó el diseño e implementación de un programa enfocado al reconocimiento facial, aprovechando las características de hardware y software que posee la MaixDuino. Por ejemplo, el uso del módulo embebido ESP32 para su conexión a una red Wi-Fi, el chip Kendryte 210 para trabajar con el modelo de inteligencia artificial encargado del reconocimiento, entre otros que se detallan en el desarrollo de este trabajo. El sistema aprovecha la integración con un software de hogar inteligente de código abierto como Home Assistant, permitiendo su comunicación y gestión a través del protocolo MQTT, una de las mejores opciones para los trabajos de IoT.

El sistema desarrollado en el computador embebido posee una efectividad de más del 90%, cumpliendo con su cometido. Además, uno de los factores más importantes es que tiene una cantidad nula de falsos positivos de acuerdo con las pruebas realizadas, lo que le proporciona una buena confiabilidad, fundamental en el campo de la seguridad. En relación con el programa desarrollado en Python, procurando medir sus respuestas bajo las mismas condiciones y parámetros, se obtuvieron resultados similares en cuanto a su efectividad. Como se vio a lo largo de este trabajo existen diferencias muy puntuales entre ellos, pero finalmente ambos cumplen a cabalidad con sus tareas.

El aporte de la computación de borde, al permitir que el hardware realice tareas de reconocimiento en la propia placa, es muy importante ya que ofrece beneficios clave en términos de velocidad, eficiencia y capacidad de respuesta frente a proyectos similares. Todo esto convierte a este sistema en una solución valiosa para una amplia gama de aplicaciones en la era del IoT.

Desarrollos Futuros

Para obtener un mejor rendimiento del sistema en general, una mejora primordial sería la implementación de un sensor de profundidad o a su vez trabajar un sistema de múltiples cámaras, como lo realizan los mejores sistemas de reconocimiento facial que existen en el mercado en la actualidad. Sin embargo, para ello se deben plantear nuevamente los recursos de hardware que se vayan a usar, incluyendo la posible elección de una nueva placa de desarrollo

De igual manera, una gran ventaja sería el poder contar un módulo de administración de usuarios con una interfaz amigable para el público en general, que permita contar con varias funcionalidades, como el registro y gestión de nuevos usuarios, dar de baja aquellos que ya no se requieran en el sistema, entre otras funciones.

Referencias

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 international conference on engineering and technology (ICET)*, 1–6.

Amazon. (s/f). *¿Qué es MQTT?* Recuperado el 11 de agosto de 2023, de https://aws.amazon.com/es/what-is/mqtt/

Atik, S. O., & Ipbuker, C. (2021). Integrating convolutional neural network and multiresolution segmentation for land cover and land use mapping using satellite imagery. *Applied Sciences*, *11*(12), 5551.

Basilio, J. K., Maniacup, J., & Martinez, J. (2021). Face Mask and Face Shield Detection Using Image Processing with Deep Learning and Thermal Scanning for Logging System. *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 1–6. https://doi.org/10.1109/HNICEM54116.2021.9731972

Bouaouda, S., & others. (2022). *Desarrollo de un sistema de Visión Artificial para ayuda a personas con dificultad motora en las extremidades superiores*.

Bradski, G., Kaehler, A., & others. (2000). OpenCV. *Dr. Dobb's journal of software tools*, *3*(2).

Bravo, C. J., Ramírez, P. E., & Arenas, J. (2018). Aceptación del Reconocimiento Facial Como Medida de Vigilancia y Seguridad: Un Estudio Empírico en Chile. *Información tecnológica*, *29*(2), 115–122. https://doi.org/10.4067/S0718-07642018000200115

Cabello Pardos, E. (2004). *Técnicas de reconocimiento facial mediante redes neuronales*. Informatica.

Canaan. (2018). *Kendryte K210*. https://www.canaan.io/product/kendryteai

Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An Overview on Edge Computing Research. *IEEE Access*, *8*, 85714–85728. https://doi.org/10.1109/ACCESS.2020.2991734

Chen, J. (s/f). *config_maix_duino*. Recuperado el 12 de agosto de 2023, de https://github.com/sipeed/MaixPy_scripts/blob/master/board/config_maix_duino.py

EMQX. (s/f). *Public MQTT Broker for IoT Testing*. Recuperado el 13 de agosto de 2023, de https://www.emqx.com/en/mqtt/public-mqtt5-broker

Fiscalía General del Estado. (2022, diciembre 31). *Estadísticas FGE - Robos*. https://www.fiscalia.gob.ec/estadisticas-de-robos/.

Flavell, L. (2011). *Beginning blender: open source 3d modeling, animation, and game design*. Apress.

George, D. (2014). *The MicroPython Documentation*. https://micropython.org/

Home Assistant. (s/f). *Installation*. Recuperado el 7 de agosto de 2023, de https://www.home-assistant.io/installation/

Huidobro, J. M., & Tejedor, R. J. M. (2010). *Manual de domótica*. Creaciones Copyright SL.

Jaramillo, C. D. (2021). Utilización del sistema de reconocimiento facial para preservar la seguridad ciudadana. *El Criminalista Digital. Papeles de Criminología*, *9*, 20–37.

Jararweh, Y., Doulat, A., AlQudah, O., Ahmed, E., Al-Ayyoub, M., & Benkhelifa, E. (2016). The future of mobile cloud computing: integrating cloudlets and mobile edge computing. *2016 23rd International conference on telecommunications (ICT)*, 1–5.

Jeppesen, J., Jacobsen, R., Inceoglu, F., & Toftegaard, T. (2019). A cloud detection algorithm for satellite imagery based on deep learning. *Remote Sensing of Environment*, *229*, 247–259. https://doi.org/10.1016/j.rse.2019.03.039

Khan, M. Z., Harous, S., Hassan, S. U., Ghani Khan, M. U., Iqbal, R., & Mumtaz, S. (2019). Deep Unified Model For Face Recognition Based on Convolution Neural Network and Edge Computing. *IEEE Access*, *7*, 72622–72633. https://doi.org/10.1109/ACCESS.2019.2918275

MaixPy. (s/f). *GitHub - sipeed/MaixPy_scripts: micropython scripts for MaixPy*. Recuperado el 17 de septiembre de 2023, de https://github.com/sipeed/MaixPy_scripts

Motl, A. (s/f). *Umqtt Example*. Recuperado el 12 de agosto de 2023, de https://github.com/daq-tools/umqtt-example/blob/main/umqtt/simple.py

No, D. (2016, junio 30). *Comparación de las Placas NodeMCU*. https://www.esploradores.com/comparacion-de-placas-nodemcu_/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

RISC-V. (2019). *Specifications*. https://riscv.org/technical/specifications/

Rozada Raneros, S., & others. (2021). *Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning*.

Scarel, G., & Müller, O. (2010). Sistema de reconocimiento facial. *Universidad Nacional del Litoral*.

SiPeed 1. (s/f). *Face Recognition*. Recuperado el 14 de agosto de 2023, de https://wiki.sipeed.com/soft/maixpy/en/course/ai/image/face_recognization.html

SiPeed 2. (s/f). *Introduction to MaixPy*. Recuperado el 11 de agosto de 2023, de https://wiki.sipeed.com/soft/maixpy/en/index.html

SiPeed 3. (s/f). *MaixDuino*. Recuperado el 12 de agosto de 2023, de https://wiki.sipeed.com/soft/maixpy/en/develop_kit_board/maix_duino.html

Sipeed 4. (2023). *Sipeed-Maixduino*. https://dl.sipeed.com/shareURL/MAIX/HDK/Sipeed-Maixduino/Maixduino_2832

Skansi, S. (2018). *Introduction to Deep Learning: from logical calculus to artificial intelligence*. Springer.

Solano, G. (2020). *Reconocimiento Facial*. https://github.com/GabySol/OmesTutorials2020/tree/master/6%20RECONOCIMIENTO%20FACIAL

Stallings, W. (2014). *Comunicaciones y Redes de Computadores* (6th ed.). Prentice Hall.

Ultimaker Thingiverse. (s/f). *MaixDuino Case*. Recuperado el 11 de agosto de 2023, de https://www.thingiverse.com/thing:4113837

W, Z., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). ACM Computing Surveys. En *Face recognition: A literature survey.* (pp. 399–458).

Zhang, H., Jolfaei, A., & Alazab, M. (2019). A Face Emotion Recognition Method Using Convolutional Neural Network and Image Edge Computing. *IEEE Access*, *7*, 159081–159089. https://doi.org/10.1109/ACCESS.2019.2949741

Zhao, Y., Wang, W., Li, Y., Colman Meixner, C., Tornatore, M., & Zhang, J. (2019). Edge Computing and Networking: A Survey on Infrastructures and Applications. *IEEE Access*, *7*, 101213–101230. https://doi.org/10.1109/ACCESS.2019.2927538

Apéndices