UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SISTEMA DOMÓTICO CON RECONOCIMIENTO FACIAL Y CHATBOT PARA GESTIÓN Y CONTROL DE SEGURIDAD DE UNA RESIDENCIA.

ERREYES GALVEZ JUAN JOSE
INGENIERO DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SISTEMA DOMÓTICO CON RECONOCIMIENTO FACIAL Y CHATBOT PARA GESTIÓN Y CONTROL DE SEGURIDAD DE UNA RESIDENCIA.

ERREYES GALVEZ JUAN JOSE
INGENIERO DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

DESARROLLO DE UN SISTEMA DOMÓTICO CON RECONOCIMIENTO FACIAL Y CHATBOT PARA GESTIÓN Y CONTROL DE SEGURIDAD DE UNA RESIDENCIA.

ERREYES GALVEZ JUAN JOSE
INGENIERO DE SISTEMAS

NOVILLO VICUÑA JOHNNY PAUL

MACHALA, 23 DE FEBRERO DE 2022

MACHALA
2022

TTIL_J-Erreyes

INFORME DE ORIGINALIDAD

1%	1%	0%	0%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	www.coursehero.com Fuente de Internet	<1%
2	Submitted to Universidad Cesar Vallejo Trabajo del estudiante	<1%
3	infohackers.org Fuente de Internet	<1%
4	www.depeca.uah.es Fuente de Internet	<1%
5	www.idemsoft.com Fuente de Internet	<1%
6	www.proz.com Fuente de Internet	<1%
7	www.tid.es Fuente de Internet	<1%
8	forums.openbravo.com Fuente de Internet	<1%
9	katherine.reilly.net Fuente de Internet	<1%

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

El que suscribe, ERREYES GALVEZ JUAN JOSE, en calidad de autor del siguiente trabajo escrito titulado DESARROLLO DE UN SISTEMA DOMÓTICO CON RECONOCIMIENTO FACIAL Y CHATBOT PARA GESTIÓN Y CONTROL DE SEGURIDAD DE UNA RESIDENCIA., otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.

El autor declara que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

El autor como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 23 de febrero de 2022

ERREYES GALVEZ JUAN JOSE
0706557998

DEDICATORIA

Dedico este trabajo a Dios, a mis padres, mis hermanos, a mis seres queridos, amigos quienes me han apoyado en cada uno de mis pasos y quienes me han impulsado permanentemente en la realización de mis objetivos en la vida y así formar los cimientos para ser un profesional.

Juan José Erreyes Gálvez

AGRADECIMIENTO

Doy primeramente gracias a Dios por bendecirme y dirigir mis pasos en mi vida. Agradezco principalmente a mis padres por inculcarme valores como la perseverancia que me han servido en cada cosa que he realizado.

A mi querida Universidad Técnica de Machala, las autoridades y a mis profesores que con sus grandes enseñanzas me han permitido consolidar conocimientos que han permitido desarrollar este trabajo.

A mi tutor de titulación, Ing. Novillo Vicuña Johnny Paul, por su orientación principal en el desarrollo de este trabajo. Gracias a todos.

Juan José Erreyes Gálvez

RESUMEN

En el mundo moderno de hoy, la seguridad del hogar se ha convertido en uno de los más importantes inconvenientes de resolver adecuadamente, por lo que la iniciativa primordial detrás de IoT es facilitar la conexión de varios aparatos eléctricos del hogar a una red local y ofrecer una identidad exclusiva para que de esta forma se permita el intercambio de datos y llegar a establecer funciones que permitan dar seguridad a la residencia.

Este trabajo se presenta un sistema que permite el control y la gestión de seguridad en una residencia por medio del reconocimiento facial, utilizando aprendizaje profundo (Deep Learning) en un dispositivo embebido denominado “Jetson Nano”, en la que se incorpora una cámara USB para la creación de las imágenes, reconocimiento facial para la vigilancia, también de una chapa eléctrica para el acceso a la residencia.

La arquitectura para la construcción de este proyecto se ha fundamentado en el software libre haciendo uso del sistema operativo Linux en la distribución Ubuntu, el lenguaje de programación de alto nivel Python que permitió la construcción del sistema, implementando librerías para la visión por computador como OpenCV y para la detección y reconocimiento facial Dlib con la api de face_recongnition que incorpora modelos pre-entrenados con redes neuronales convolucionales para la detección y algoritmo de clasificación “knn” mejorado con una precisión del 99.38%. Para el acceso se utilizó la biblioteca de manejador de señales digitales Jetson.Gpio. Para controlar la vigilancia y gestionar el acceso para evitar que cualquiera entre a la residencia se hizo uso de un chatbot genérico a base de lenguaje natural.

Luego de desarrollar el sistema y ensamblado de los dispositivos se efectuaron las pruebas necesarias para medir la eficiencia del sistema, tanto del reconocimiento facial para la vigilancia y acceso. Los resultados presentan que el algoritmo de detección y reconocimiento funcionan de forma efectiva en un sistema integrado de bajo coste.

Palabras clave: Seguridad, Reconocimiento facial, Aprendizaje profundo, Dlib, OpenCV, face_recognition, domótica.

ABSTRACT

In today's modern world, home security has become one of the most important issues to solve properly, so the primary initiative behind IoT is to facilitate the connection of various electrical appliances in the home to a local network and provide a unique identity to enable the exchange of data and get to establish functions to provide security to the residence.

This work presents a system that allows the control and management of security in a residence through facial recognition, using deep learning (Deep Learning) in an embedded device called "Jetson Nano", which incorporates a USB camera for the creation of images, facial recognition for surveillance, also an electric plate for access to the residence.

The architecture for the construction of this project was based on free software using the Linux operating system in the Ubuntu distribution, the high-level programming language Python that allowed the construction of the system, implementing libraries for computer vision such as OpenCV and for facial detection and recognition Dlib with the face_recongnition api that incorporates pre-trained models with convolutional neural networks for detection and classification algorithm "knn" improved with an accuracy of 99.38%. The Jetson.Gpio digital signal handler library was used for access. A generic natural language-based chatbot was used to control surveillance and manage access to prevent anyone from entering the residence.

After developing the system and assembling the devices, the necessary tests were carried out to measure the efficiency of the system, both in terms of facial recognition for surveillance and access. The results show that the detection and recognition algorithm works effectively in a low-cost integrated system.

Keywords: Security, Face recognition, Deep learning, Dlib, OpenCV, face_recognition, home automation.

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

La seguridad del hogar constituye un factor primordial durante años en la vida cotidiana de cada persona. Actualmente debido a la existencia de nuevas tecnologías en lo que respecta a la seguridad física ha coadyuvado a automatizarla en varios requerimientos para que sea más efectiva y eficiente.

La construcción de software y hardware que integran los sistemas de seguridad han tenido una gran evolución exhaustiva, los sistemas biométricos es una muestra de ello, tanto que se han venido utilizando por mucho tiempo por su efectividad, como es el caso de la clásica autenticación con claves, luego con la llegada de la huella digital, acceso por tarjeta, el monitoreo constante con la video vigilancia, redes de sensores.

En los últimos 10 años, la biometría se ha convertido en un área de indagación bastante conocida en la visión por computador y una de las aplicaciones más famosas del estudio de la comprensión de imágenes por medio del ordenador [1].

En la actualidad, la (inteligencia artificial) (IA) se ha venido incorporando de manera más natural a la vida diaria de la población y las operaciones de las organizaciones. La inteligencia artificial está evolucionando para brindar incontables beneficios a todos los sectores productivos como la salud, retail, manufactura, policial y educación.

Recientemente, los avances en los sistemas de vigilancia se han producido gracias a los diferentes dispositivos del (IoT) internet de las cosas [2]. La domótica emplea tecnologías IoT para dotar a los a los pobladores del hogar un más grande bienestar, eficiencia energética y mayor seguridad [3]. Por ello la domótica posibilita ofrecer contestación a los requerimientos para hacerle frente a cambios sociales y las tendencias recientes de nuestra forma de vida, haciendo más fácil el diseño de viviendas y domicilios más humanos, más particulares, polifuncionales y flexibles.

La automatización en estos años está triunfando trascendentalmente en el campo del diseño de interiores, debido a que ayuda a mejorar los espacios interiores usando tecnología inteligente. Los sistemas de automatización del hogar se fundamentan en la utilización de la tecnología inteligente programada para satisfacer necesidades [4] .

El reconocimiento facial es un procedimiento mediante el cual se puede identificar a las personas comparando las características del rostro, por lo cual esta técnica es aplicada últimamente para el acceso en la que las áreas son controladas y la video vigilancia [5].

La iniciativa tecnológica de este trabajo tiene como fin el diseño y desarrollo para hogares pequeños un prototipo de sistema video vigilancia y de acceso con reconocimiento facial y la utilización de algoritmos de aprendizaje profundo (deep Learning) en un dispositivo embebido “Jetson Nano” programado con Python para luego hacer uso de un chatbot donde el cliente consulte los individuos que han accedido o visitado al hogar.

El presente documento se divide en tres capítulos descritos a continuación:

En el c**apítulo 1** se presenta la propuesta tecnológica, donde se describe el diagnóstico de necesidades y requerimientos, detallando la justificación para la consecución de este trabajo.

En el c**apítulo 2** se define la propuesta tecnológica y arquitectura, continuando con la fundamentación teórica de los recursos de software y hardware utilizados, luego se establecen los objetivos y por último el proceso de diseño, ensamblaje del hardware y la ejecución del software.

En el c**apítulo 3** se define el proceso de evaluación del prototipo con el uso de la matriz de confusión para posteriormente mostrar los resultados del rendimiento de la técnica de reconocimiento facial mencionada para este proyecto.

1. CAPÍTULO I: DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS

1.1. Ámbito de Aplicación: descripción del contexto y hechos de interés

La seguridad del hogar es un factor importante en el mundo actual, se hace necesaria hoy en día, ya que las posibilidades de intrusión aumentan día a día.

Según [3] , la domótica emplea tecnologías IoT para dotar a los habitantes del hogar un mayor confort, eficiencia energética, seguridad e interacción con sus electrodomésticos incluso cuando no hay nadie en casa.

La seguridad y la privacidad en casa son relevantes para cada persona, por ello tener una base de datos local de individuos identificados ayuda tanto con la seguridad como con la privacidad del hogar [6] .

El reconocimiento facial es una identificación personal biométrica basada en las características físicas humanas que es un rostro [7]. Esta consiste en una aplicación informática capaz de detectar, seguir, identificar o verificar rostros humanos a partir de una imagen o un vídeo capturado con una cámara digital [8].

La tecnología de reconocimiento facial se ha venido utilizando en la empresa privada como en el de la pública para una pluralidad de usos, que van desde la seguridad física, hasta las experiencias de compra personalizadas.

El Chatbot se conceptualiza como un programa que ayuda a los humanos a hacer un dialogo coherente con la máquina usando el lenguaje natural como inglés, etcétera y la plática podría ser llamativa algunas veces con monumentales vocabularios y extensa gama de temas de plática, recientemente la utilización del aprendizaje profundo ha incrementado en la industria y Chatbot es una de sus aplicaciones [9].

Este trabajo tiene como objetivo diseñar e implementar para hogares pequeños un prototipo de sistema de video vigilancia y de acceso con reconocimiento facial con el uso de algoritmos de aprendizaje profundo (deep Learning) para luego integrarlo en un sistema domótico controlado por un chatbot para que el usuario consulte las personas que han entrado o han visitado al hogar, sean conocidos o desconocidos. La finalidad de este proyecto es que exista un sistema que permita el control de seguridad de una residencia usando Inteligencia Artificial (IA).

1.2. Establecimiento de Requerimientos

Para el prototipo se espera portabilidad por lo tanto en cuanto a hardware se lo desarrollará en tarjetas de bajo coste: NVIDIA Jetson Nano 2GB con GPU integrada elaborada por el fabricante NVIDIA, junto a una cámara digital USB con resolución Full HD (1080X1920) a 30 FPS para capturas de rostros del dataset, vigilancia y acceso, chapa eléctrica a 12v para abrir y cerrar puerta, un relé A 5V que controle el encendido y apagado. En la parte de software el sistema se lo desarrollará en el lenguaje de programación python que trae librerías propias de inteligencia artificial (IA) en lo que respecta a visión artificial se utilizará Opencv y la biblioteca face_recognition de DLIB que posee modelos pre-entrenados de detector de caras y algoritmos para reconocer y manipular rostros a base del aprendizaje profundo (deep learning). Se agregará la capacidad de que el sistema almacene las capturas en formato de imagen jpg en una carpeta con fecha y las imágenes con la hora, esta carpeta se denominará historial y esta va estar almacenada en la dirección principal del proyecto de software. También se establecerá una base de datos y almacenará información de quienes visitan y acceden al hogar sean conocidos o desconocidos y presentar en el chatbot un historial con la información del individuo cuando haya visitado o accedido; si el propietario está dentro de la residencia mostrarlo en un monitor, siempre y cuando detecte una nueva persona.

1.3. Justificación del requerimiento a satisfacer

Según [1] el mundo moderno de hoy la seguridad del hogar se convirtió en uno de los más importantes inconvenientes que se debe abordar correctamente. La iniciativa primordial detrás de IoT es conectar diferentes objetos mecánicos y eléctricos que nos rodean a una red informática y ofrecer a todos ellos una identidad exclusiva para que de esta forma permita la comunicación de datos en conjunto.

Para el propietario de la residencia le nace el deseo de saber quiénes visitan a su puerta con su sistema básico de video vigilancia en la puerta mientras no esté presente, ya sea un individuo conocido o desconocido, en el caso de que quiera dar acceso a una persona conocida del hogar cuando él esté presente requiere movilizarse a la puerta para permitir el acceso. Con la realización de este prototipo busca satisfacer esos requerimientos mencionados, por ello existirá un sistema que permita el control de seguridad de una residencia usando reconocimiento facial y chatbot, con una cámara de seguridad en la puerta del hogar donde reconozca el rostro de las personas conocidas de la residencia y a su vez detectar a desconocidas

para que la información de estas sean almacenadas en una base de datos, y las captura en tiempo real guardarlas en el disco duro principal del dispositivo. Tendrá la capacidad de permitir el acceso por la puerta a los individuos conocidos con el consentimiento del propietario.

2. CAPÍTULO II: DESARROLLO DEL PROTOTIPO

2.1. Definición del prototipo tecnológico

Para base de este proyecto se ha tomado de referencia lo que explican los autores [10] en su libro de técnicas modernas de aprendizaje profundo (Deep Learning) sobre etapas que conlleva comúnmente una solución para el reconocimiento facial moderno. En la figura 1 se evidencia este proceso descrito en un diagrama.

Figura 1. Etapas del reconocimiento facial

Fuente: Elaboración Propia

Fundamentado con lo anteriormente mencionado, se ha hecho posible desarrollar un prototipo de sistema y hardware para el reconocimiento facial enfocado a la seguridad de una residencia.

En la figura 2, se puede visualizar de una forma resumida y específica un esquema de la arquitectura del prototipo realizado, fundamentada por 4 etapas, de la siguiente manera:

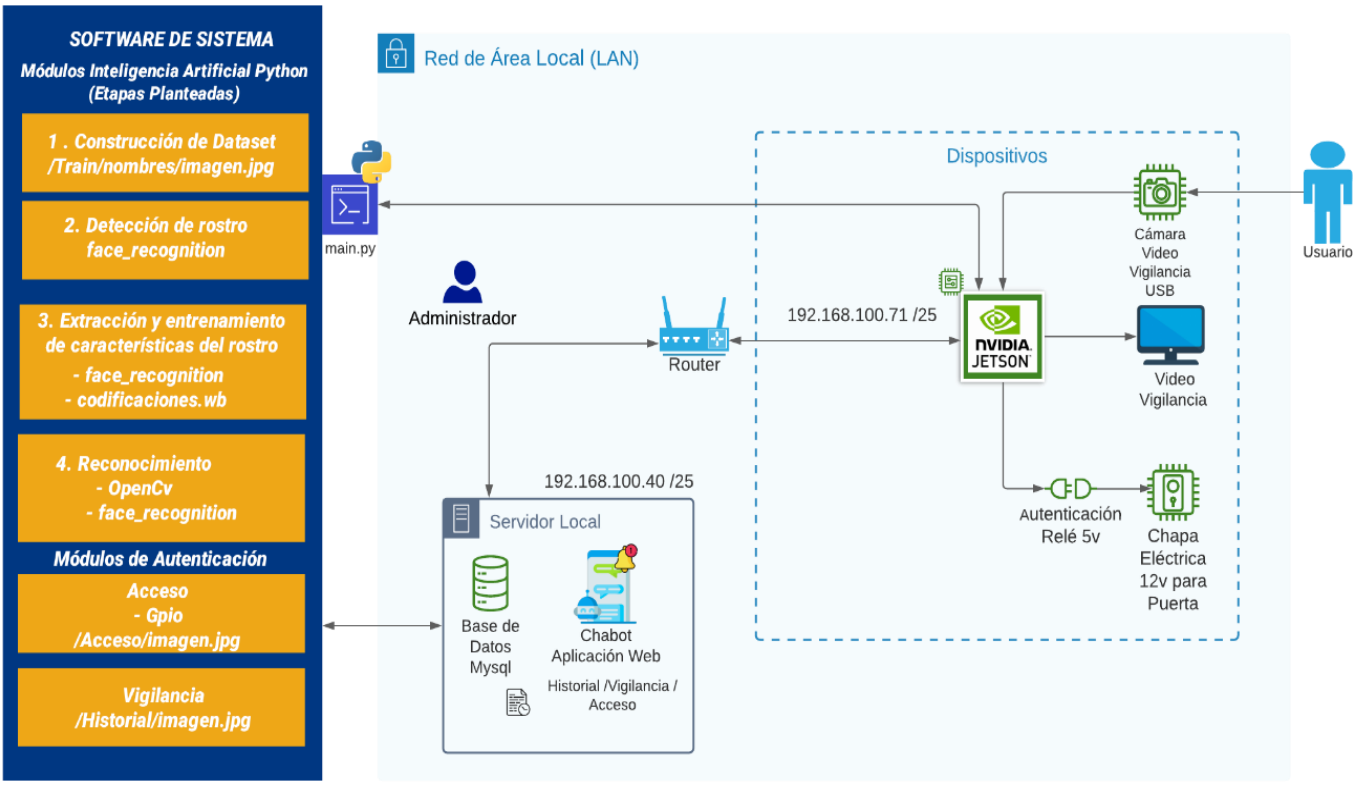


Figura 2. Arquitectura del sistema domótico con reconocimiento facial y chatbot para gestión y control de seguridad de una residencia.

Fuente: Elaboración Propia

Etapa 1: Construcción de Dataset

Módulo para la construcción de dataset, este servirá para el proceso de entrenamiento, en esta se captura fotos del rostro de la persona a ser almacenada en la base de datos como conocido haciendo uso del siguiente módulo de detección de rostro, además se pedirá la información general, como sus nombres completos, su identificación que es un número que no se pueda repetir con el resto preferible el número de cédula, y el parentesco que tenga con el propietario, ya sea esposa, hijo, primo, etc. Luego se etiquetará con los nombres completos a un directorio individual para el entrenamiento.

Etapa 2: Detección del rostro

Este módulo permitirá la detección de rostros, haciendo uso de la librería face_recognition con el método face_locations que posee modelos de detección facial pre-entrenados; luego se procede almacenar en el directorio de entrenamiento como archivo de imagen; en el próximo punto teórico de este capítulo se explicará con más detalle su funcionamiento e implementación.

Etapa 3: Extracción y entrenamiento de características o codificaciones de rostro.

Luego de haber construido el dataset de entrenamiento, en este módulo se procede en cada imagen, extraer las codificaciones faciales de las personas de manera individual. Se usa nuevamente la librería face_recognition con el método face_encodings que permite la codificación de la cara en 128 dimensiones con entrenamiento para luego ser almacenada en un archivo binario.

Etapa 4: Reconocimiento

Para este módulo se utiliza otro de los métodos de face_recognition el de compare_faces que dada una lista de codificaciones de caras compara con una codificación candidata para ver si coinciden. Para iniciar el video se usa la librería WebcamVideoStream, y para dibujar el cuadro limitador de rostro en frame se usa OpenCv.

A continuación, finalizada cada etapa, se hace uso de los módulos de autenticación, en el de acceso, hace uso de las librerías de Python que permite manipular señales digitales en la Jetson Nano en este caso GPIO, que, al identificar un rostro conocido en la cámara, envía señal en alto a un pin configurado para que encienda el Relé y proceder abrir la puerta con la chapa eléctrica para luego almacenar la información de acceso en la base de datos establecida (MySQL), como fecha, hora, identificación, imagen del rostro y del momento exacto de ingreso.

El módulo de vigilancia en cambio almacenará la información en la base de datos, campos como de fecha, hora, identificación, estado, imagen del rostro y del momento en tiempo real de los individuos que aparezcan en cámara de video sean conocidos o desconocidos cada minuto.

Al final el propietario puede hacer preguntas al chatbot para verificar quien accedió a su casa o quienes la visitaron el día de hoy, tales como conocidos o desconocidos.

2.2. Fundamentación teórica del prototipo

2.2.1. Biometría

De acuerdo con la biometría se la identifica como la ciencia que permite detectar y comprobar la identidad de una persona en funcionalidad de sus propiedades físicas y de comportamiento, sin embargo, comúnmente los términos identificar y verificar acostumbran ser confusas, sin embargo, resultan muy variadas [11].

En la última década, la biometría se ha venido convirtiendo en un área de investigación bastante exitosa como parte de la visión por computador y una de las aplicaciones más famosas del análisis y comprensión de imágenes [1].

La cara es un dato biométrico fundamental para la identificación de individuos que, gracias a las satisfactorias prestaciones de reconocimiento, los sistemas de identificación basados en la biometría de la cara se están convirtiendo en omnipresentes.

2.2.2. Sistema de video Vigilancia

La vigilancia del hogar ha venido siendo un elemento fundamental a lo largo de años en nuestras propias ocupaciones diarias. Las técnicas clásicas de vigilancia han usado redes de sensores inalámbricos o circuitos cerrados de televisión como lo son los sistemas de video vigilancia que exigen que las personas supervisen las transmisiones de las cámaras en directo.

Según [2] plantea que recientemente, los adelantos en los sistemas de vigilancia de los sistemas de vigilancia se han producido debido a los dispositivos (IoT) internet de las cosas

2.2.3. Inteligencia Artificial

La inteligencia artificial (IA) es la simulación de procesos de sabiduría humana a causa de máquinas, en especial sistemas informáticos. Dichos procesos integran el aprendizaje la adquisición de información y normas para la utilización de la misma, el entendimiento emplea pautas que permiten llegar a conclusiones aproximadas o categóricas y como autocorrección del resultado final del proceso [12].

Según [13] la IA se ha desarrollado velozmente en los últimos años, conocida popular como inteligencia artificial para acortar, es una especialidad en el campo de la informática, y es una disciplina que estudia cómo hacer que las PCs simulen el raciocinio y la conducta inteligente de los humanos .

2.2.4. Deep Learning (Aprendizaje Profundo)

Lo que respecta al aprendizaje profundo es un subcampo del machine learning en español aprendizaje automático que utiliza arquitecturas jerárquicas para aprender abstracciones de alto nivel en datos. Ante la existente aparición de big data y el exitoso triunfo que ha venido teniendo el aprendizaje profundo, se han planteado

diversos algoritmos de aprendizaje de representación multivista basados en redes neuronales profundas (DNN).
[14].

Según [15] menciona que el aprendizaje automático ha adquirido interés general en los últimos años gracias a su elevado rendimiento en la clasificación y el dominio de aprendizaje representacional desde datos en bruto.

2.2.5. Visión por computador

La visión por ordenador es considerada un método muy utilizado para el análisis de imágenes y vídeos ya que su uso mayormente está enfocado en aplicaciones de automatización [2].

En un contexto como en la detección, rastreo de objetos, reconocimiento, la función de las personas no es explícitamente tan precisa y a la vez afectiva, debido a que se ve reducida a las habilidades visuales de cada individuo, por consiguiente, usar tácticas relacionadas a la visión por computador, posibilita facilitar el rastreo de objetos por medio de su representación desde patrones [16].

Puede decirse que el algoritmo de visión por ordenador es un modelo matemático para el procesamiento de imágenes, es más común en la (IA) inteligencia artificial y tiene fuertes ventajas técnicas ya que recibe información desde imágenes o datos [17].

2.2.6. Reconocimiento Facial

Se define como una identificación personal biométrica basada en las propiedades físicas humanas en este caso el rostro, ya que el rostro humano es exclusivo y simple de ver, este procedimiento está triunfando mucha atención por investigadores y se lleva a cabo en aplicaciones comerciales [7].

El reconocimiento facial también es denominado como sistema de detección facial, que es la función de identificar la localización de la cara en cualquier imagen con cuadro delimitador de salida con las coordenadas de los rostros detectados, sin embargo, el reconocimiento facial es un proceso que compara diversos rostros ligados para detectar, qué rostros son los que pertenecen a la misma persona. Esto se realiza comparando los vectores de incrustación faciales [18].

El reconocimiento facial emplea muchos procedimientos supervisados en los cuales se otorgan instancias con etiquetas conocidas para las salidas necesarias que corresponden y el aprendizaje en el cual las instancias no permanecen etiquetadas [19].

2.2.7. Red neuronal artificial (ANN)

IBM en su página de información manifiesta que las redes neuronales son modelos básicos que imitan el funcionamiento del sistema cerebro humano que se manejan son por medio de neuronas, que por lo general están organizadas en capas [20].

Representada por sus siglas ANN en inglés Artificial Neuronal Network que traducido es una red neuronal artificial, tomando lo manifestado anteriormente se la puede denominar como un modelo de computación que viene de una inspiración biológica del cerebro humano, compuesto por centenares de unidades individuales que la componen, como son las neuronas artificiales que están conectadas con coeficientes llamados pesos que conforman la composición neural. En la figura 3 se presenta la arquitectura de una red neuronal artificial.

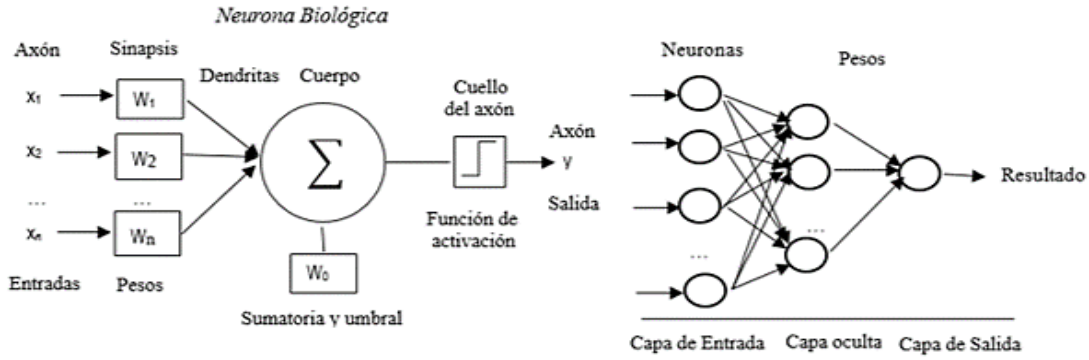


Figura 3. Red Neuronal Artificial ANN

Fuente: [21]

Para el procesamiento estas se estructuran en capas, por lo que existe 3 segmentos comúnmente en una red neuronal:

- **Capa de entrada:** son unidades que representan los campos que se ingresan.
- **Capa oculta:** una o varias.
- **Capa de salida:** una unidad o varias unidades de campos de salida.

Estas se enlazan con nervios de conexión que van cambiando, donde los datos que se ingresan se muestran en la primera capa, y los valores se expanden a partir de cada neurona hasta la capa siguiente con sus neuronas y finalmente se presenta un resultado desde la capa de salida.

2.2.8. Red neuronal convolucional CNN

Sus siglas (CNN) es un tipo de red artificial con enfoque de aprendizaje profundo (Deep Learning), en el cual se entrenan diversas capas de manera robusta [14].

Tipo de red artificial que utiliza la convolución como herramientas en el procesamiento y más enfocado en imágenes utilizada para la clasificación y agrupamiento de imágenes para realizar el reconocimiento de objetos y de rostros en conjunto [22].

En la figura 4 se muestra una arquitectura general de CNN para la clasificación de imágenes capa por capa.

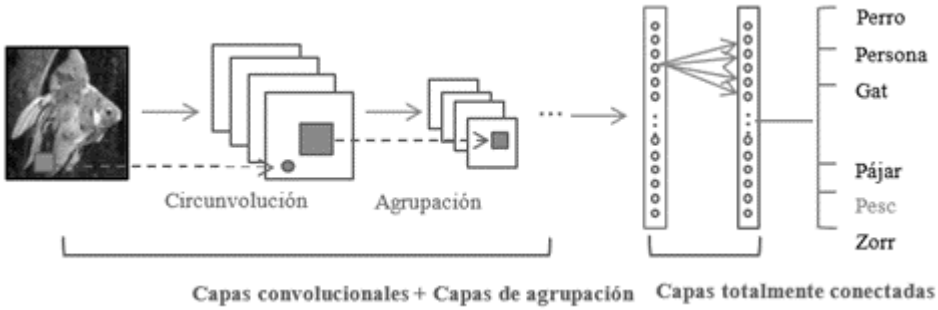


Figura 4. Arquitectura General de CNN.

Fuente: [14]

Las cnnn redes neuronales convolucionales o también denominadas convnets esta toma en cuenta que en su entrada consiste en imágenes y esta limitan la arquitectura de la red como se puede visualizar en la Figura 5, se diferencia una red neuronal artificial a la normal por donde las capas que tiene un poseen neuronas colocadas en 3 dimensiones de alto, ancho y profundidad.

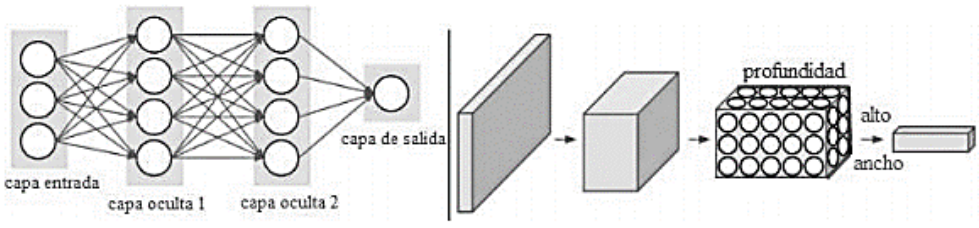


Figura 5. Vista general de red neuronal convolucional vs la normal.

Fuente: [23]

2.2.8.1. CNN (Resnet-34)

La red neuronal Resnet también denominado Red Network traducido una red residual fue realizada por un equipo científico en investigación de IA de Microsoft, realizaron una publicación donde manifiestan que las redes neuronales más profundas son complejas de realizar por ello proponen un marco de aprendizaje residual que facilite el entrenamiento de redes que son principalmente más profundas que las usadas normalmente, por lo que obtuvieron una tasa de error del 3.57 % [24].

La solución primordial que lleva es cómo hacer que la red sea más profunda para que aprenda mejor es decir que la exactitud no se reducirá ya que cuanto más profundamente sea la profundidad de la red, mejora las propiedades extraídas.

Esta red fue planteada en inspiración en las redes neuronales VGG (VGG-16, VGG-19), en cuanto a las redes convolucionales que tienen filtros de 3×3. Mencionan que la degradación lo que conlleva a la exactitud del entrenamiento sugiere que no todos los sistemas son igual de simples de optimizar y considerando una arquitectura más superficial y por otra parte más profunda que incorpora más capas, hay una solución por creación para el modelo más profundo: las capas anexadas son mapeos de identidad y las capas restantes se copian del modelo superficial aprendido [24] .

Los bloques que quedan excedentes se los consideran el componente principal para ResNet, está el bloque residual es decir que la entrada (x) se agrega llanamente a la salida de la red dada por la función (F (x) + x) esta ruta se la denomina la conexión de acceso directo o salto.

El funcionamiento general de esta red convolucional residual se la muestra en un esquema elaborado por los autores está representada en la figura 6.

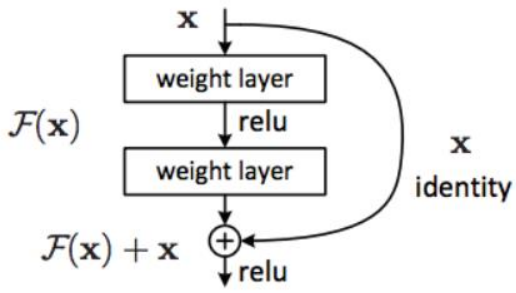


Figura 6. Aprendizaje residual, un bloque de construcción.

Fuente: [24]

La función de salida es: H (x) = F (x) + x, por lo que F (x) = 0, entonces H (x) = x, lo que se denomina como mapeo de identidad, lo que quiere decir cuando la entrada de la red es similar a su salida. Para realizar la suma F (x) + x, la forma en conjunto debe ser exactamente la misma, ya que si la forma no es la misma se la multiplica la entrada x con una matriz Wi, y= F(x ,{w_i } + x. En la figura 7 se representa una red residual de 34 capas.

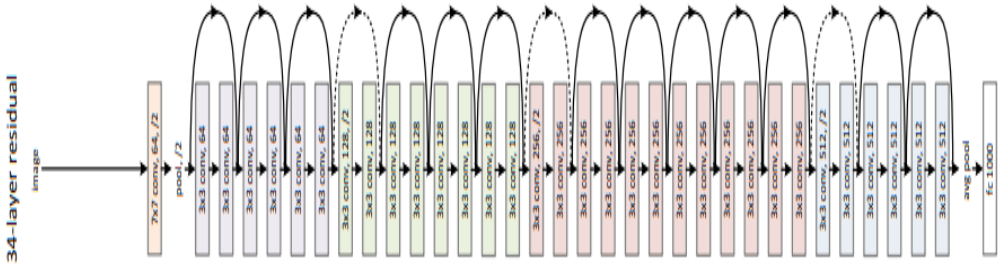


Figura 7. Red residual con 34 capas.

Fuente : [24]

Se hace nominación de esta red neuronal convolucional ya que la biblioteca face_recognition está entrenada con una red neuronal convolucional de este tipo ResNet, pero con 29 capas lo que genera una mayor precisión para el reconocimiento facial.

2.2.9. Software y Hardware

2.2.9.1. Ubuntu LTS 18.04

La distribución GNU/Linux es un sistema operativo orientado a servidores, esta es una basada en Debian sus propiedades son la facilidad que tiene para el funcionamiento, dirección y actualizaciones comunes.

Para este proyecto que se usa una tarjeta Jetson Nano integrada con una GPU, es factible la utilización de este sistema operativo Ubuntu ya que añade controladores para el primordial fabricante de tarjetas gráficas NVIDIA, lo cual incrementa el rendimiento de los equipos donde se instala y a la vez que disminuye el tiempo de configuración del sistema luego de haber hecho su instalación [25].

2.2.9.2. Python

Python es un lenguaje de programación para construcción de sistemas en corto tiempo con una manera más efectiva [26]. Es de simple aprendizaje, la cual tiene un enfoque sencillo, pero realmente efectivo en la programación orientada a objetos.

Integra librerías propias de la Inteligencia artificial accesibles de manera gratuita en el portal web oficial de python.

2.2.9.3. Opencv

Sus siglas Visión (OpenCV), Open Source Computer es una librería que integramás 500 algoritmos que se encuentran optimizados para el procesamiento de imágenes y video incluyendo una librería de MLL para reconocimiento de patrones y clustering [27]. Para la mayoría de las plataformas como Linux, Windows, OS X, y más se encuentra disponible.

Ha sido construida con el fin de proveer una infraestructura común para las aplicaciones de visión por ordenador [28]. Muchos autores [29] [30] hacen uso de esta librería debido al gran número de funciones que están incluidas en ella que facilitan el procesamiento de imágenes. La arquitectura se la describe en la figura 8.

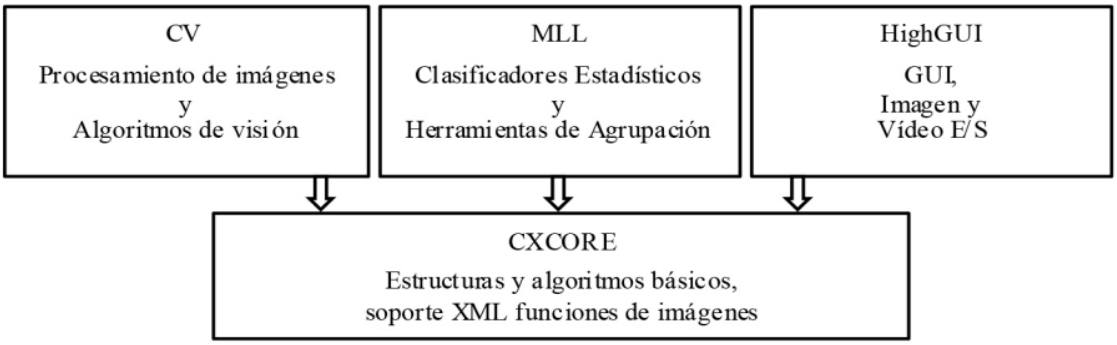


Figura 8. Arquitectura OpenCV

Fuente: [31]

OpenCV involucra varias clases entre ellos trae la clase abstracta denominada face_recognition que está sumamente enfocada al reconocimiento facial.

En el presente proyecto para el reconocimiento facial se necesita de dos bibliotecas anexadas como dlib y face_recognition.

2.2.9.4. Dlib

Dlib trae consigo un kit de herramientas elaborado con c++ que integra algoritmos de aprendizaje automático (ML) para construir software y solucionar inconvenientes de todo el mundo real. Se la ha venido usando en la industria y en la parte académica en una extensa gama de tópicos, como en la robótica, los dispositivos embebidos, teléfonos inteligentes y equipos informáticos de elevado rendimiento y algo importante

que Dlib posibilita usarlo en cualquier aplicación ya que tiene licencia de código abierto totalmente de manera gratuita [32].

La biblioteca dlib, mantenida por Davis King, contiene para este proyecto la implementación de aprendizaje métrico profundo (Deep Learning) que se utiliza con la librería face_recognition para la construcción de las incrustaciones faciales que se usan luego para el proceso de reconocimiento facial en tiempo real.

2.2.9.5. Face_recognition

Es una biblioteca que fue desarrollada por Adam Geitgey, para el reconocimiento facial, esta librería es de código abierto, está construida con Deep Learning dentro de Dlib suministrada de algoritmos como aprendizaje automático (Machine Learning), aprendizaje profundo para el procesamiento de imágenes además de redes neuronales convolucionales [33].

La biblioteca fue entrenada con una red neuronal convolucional tipo ResNet con 29 capas en el dataset LFW (Labelled Faces in the Wild) demostrando una precisión de acierto del 99.38% [33].

Detección de rostro (Face_locations)

Comúnmente al usar OpenCv para la detección de rostros muchos autores lo realizan con una modelo llamada Haar Cascades, pero según [34] menciona que existen muchos algoritmos más precisos que haar, cómo: (HOG + Linear SVM, SSD, Faster R-CNN, YOLO, por nombrar otros más.

Es de preferencia usar modelos que son pre-entrenados ya que los tiempos de inferencia como el costo computacional se reducen significativamente[35].

En este proyecto para llevar a cabo la detección del rostro se emplea una (CNN) que es suministrada por Dlib con el nombre “mmod_human_face_detector.dat” es un modelo que está pre-entrenado*.* Esta red fue entrenada completamente desde cero en un set de datos que poseía cerca de 3 millones de rostros, dando un modelo resultante que tiene un error medio de 0,993833 con una desviación estándar de 0,00272732 desde el benchmark LFW [36].

En el paquete de ***face_recognition*** explica qué modelo de detección de rostros utilizar y mencionan que: "hog" tiene una precisión menor y consumen menos CPU. "cnn" es más preciso ya que está acelerado por la GPU/CUDA [37]. El valor por defecto es "hog", pero para este proyecto se usa “cnn”.

Por este fin la CNN realiza el siguiente procedimiento:

Esta procesa la imagen y la pasa por 3 capas para la disminución de muestras y en estas capas la reducen en una imagen de 8 pixeles por lo que generan un mapa de 32 dimensiones en cuanto a las características, en la figura 9 se puede este procedimiento [38].

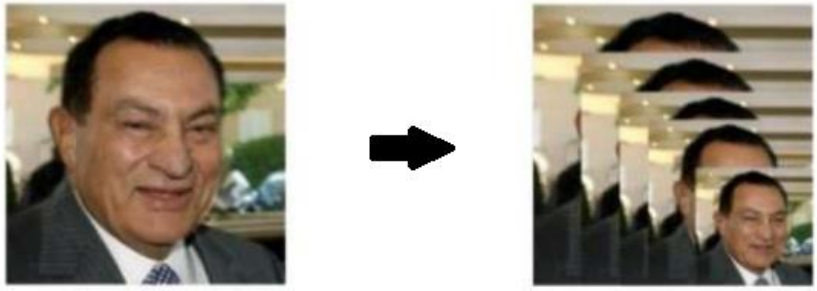


Figura 9. Detección y alineación conjunta de rostros mediante redes convolucionales multitarea en cascada

Fuente: [39]

Luego utiliza 4 capas convolucionales en la que pasa la imagen que leen todos los valores obtenidos en el mapa, seleccionado zonas que son de interés y embotellando estas en cuadro delimitadores. En la figura 10 se puede visualizar este proceso.

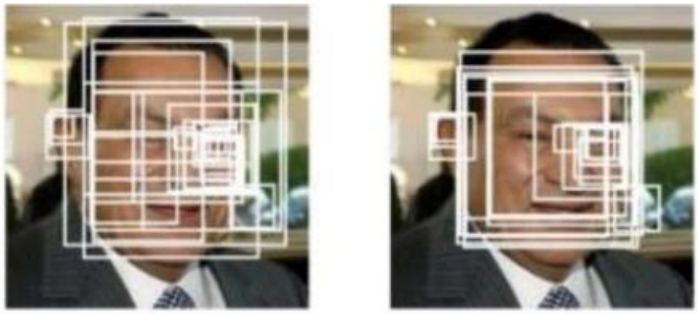


Figura 10. Localización de regiones de interés (ROI).

Fuente: [39]

Esta selección del ROI es aplicada con un puntaje de confiabilidad en cada cuadro delimitador; por lo que este si es alto quiere decir que es una región de interés y se ubica en el cuadro delimitado.

En la última capa indica como capa de pérdida, y calcula el valor que muestra qué cual está funcionando mejor la red neuronal convolucional CNN en el proceso de

detección. La figura 11 se visualiza en el cuadro delimitador, como resultado final la imagen que pasa por la red neuronal CNN.

Detección del rostro con CNN.

Figura 11. Detección de rostro mediante CNN.

Fuente: [39]

En la figura 12 se visualiza la estructurada de una red neuronal convolucional CNN para el proceso de detección de rostro.

Estructura de la red neuronal convolucional CNN

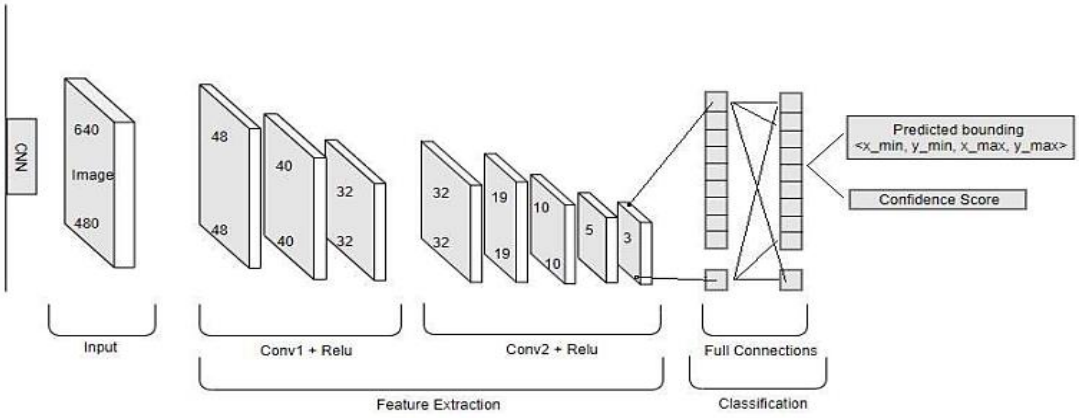


Figura 12. Estructura de red neuronal convolucional CNN para detección de rostro.

Fuente: [36]

Esta red neuronal para la detección de rostro CNN fue entrenada en un data set de 3 millones de imágenes, construida a partir de diferentes grupos de datos como VGG, VOC y más por lo que logra un porcentaje de precisión en la detección del 99.38% [36].

Para trabajar con esta red neuronal CNN, se necesita que los valores de cada imagen de entrada estén ajustados, a continuación, se muestran en la Tabla 1.

Tabla 1. Parámetros estandarizados del modelo empleado

Modelo	Escala	Tamaño pixeles (Largo x Ancho)	Resta Media	Orden de los canales
mmod_human_face_detector.dat	1	640x480	104,177,123	RGB

Fuente: Elaboración propia.

Se utiliza este modelo ya que en la Jetson Nano posee una GPU o Tarjeta Gráfica Integrada por lo que tiene altas prestaciones para usarlo y poder sacar provecho de las ventajas mencionadas anteriormente, en el caso de utilizar CNN esta ofrece una mejor precisión en el proceso de reconocimiento del rostro. En la figura 13 podemos visualizar un collage de pruebas de la precisión de detección de rostro incluso en ángulos complejos.

Precisión de la detección de rostro en el sistema.

Figura 13.Precisión de detección de rostro con CNN en GPU Jetson Nano.

Fuente: Elaboración propia.

Codificaciones y/o Incrustaciones de rostro (Face_encodings)

El proceso de codificar los rostros que maneja esta librería es la de un aprendizaje métrico profundo muy diferente al del aprendizaje profundo que se capacita a una red para aceptar solamente una imagen de entrada y como salida realice una clasificación/etiqueta de la imagen. En lugar que genere una sola etiqueta, se va a emitir un vector de característica con valores.

El método predictor de forma de 68 puntos viene incorporado en los modelos del api de face_recognition con dlib, que tiene el nombre de "***shape_predictor_68_face_landmarks.dat***", fue pre-entrenado, haciendo que determine puntos específicos en las regiones de la cara como boca, cejas, nariz, ojos y mandíbula. Estos índices de 68 puntos se presentan en la figura 14.

Figura 14. Puntos para referencia de rostro detectado - esquema

Fuente: [40]

La figura 15, muestra una prueba realizada con el predictor de forma de 68 puntos,

Figura 15. Rostro con bordes identificados y landmarks dibujados.

Fuente: Elaboración propia.

Antes de que se pueda reconocer rostros en imágenes o en videos, primero se debe cuantificar las caras del dataset de entrenamiento. Según [35] manifiesta que no se está entrenado la red, si no que esta ya ha sido entrenada para crear incrustaciones 128-d. Esta formación se realiza mediante trillizos. En la figura 16 se visualiza el procedimiento:

Entrenamiento mediante tripletes.

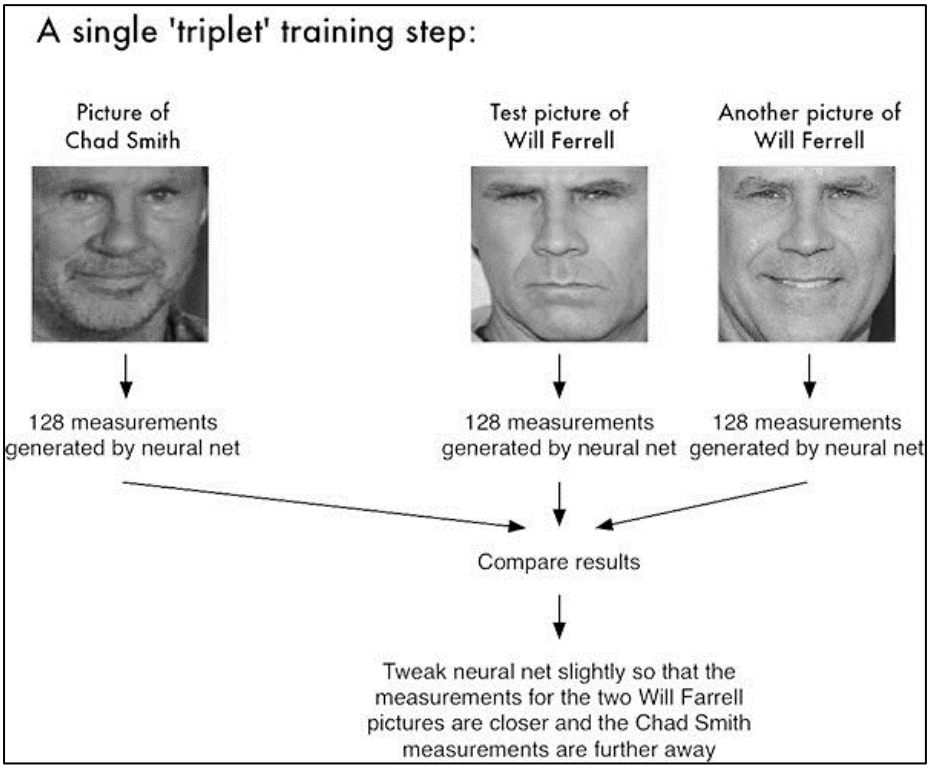


Figura 16. Entrenamiento mediante tripletes.

Fuente: [40]

En la representación de la figura 16 el autor manifiesta que se proporcionan 3 imágenes de red, donde dos de estas imágenes son el rostro de la misma persona y la tercera imagen es una cara que es seleccionada aleatoriamente y no deben ser de la misma persona que se tiene las 2 imágenes. La red cuantifica los rostros, construyendo una incrustación que es la cuantificación 128-d para cada una tal como se visualiza en la figura 17, para que luego se pueda acordar los pesos de la red para que esas mediciones de las dos imágenes estén cerca entre ellas y a mayor distancia las mediciones de la aleatoria.

Descriptor 128D a una imagen del dataset de entrenamiento.

Imagen de entrada

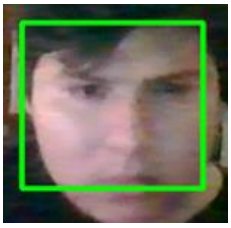


Imagen codificada a 128D

```
[array([-0.14102907,  0.1218354 ,  0.0751892 , -0.00645518, -0.12173542,
         0.02305251, -0.05429415, -0.11522955,  0.20934038, -0.11493328,
         0.26588392,  0.0026087 , -0.27224889, -0.04433503, -0.07100123,
         0.13537443, -0.10835259, -0.11851592, -0.03312575,  0.00873024,
         0.05240052, -0.02250515,  0.08595543,  0.02243967, -0.17975312,
        -0.37293863, -0.11317863, -0.10128213, -0.04467701, -0.06118239,
        -0.05100561, -0.07718913, -0.16485539, -0.06552239, -0.01622237,
         0.05210717, -0.07904266, -0.05637395,  0.2253736 , -0.04772786,
        -0.21742913, -0.03821171,  0.03449536,  0.20532858,  0.1285657 ,
         0.07940934,  0.05931298, -0.13216379,  0.07670875, -0.20934454,
         0.12976371,  0.11271225,  0.0996123 ,  0.00324515,  0.08203301,
        -0.12746936, -0.02688375,  0.14795165, -0.16238259,  0.01433081,
         0.06244088, -0.00574952, -0.04863946, -0.05526987,  0.24281617,
         0.16852531, -0.09132722, -0.12931097,  0.21726927, -0.13662763,
        -0.03909035,  0.05005407, -0.10465588, -0.24280486, -0.2735236 ,
         0.03862845,  0.45757541,  0.15386863, -0.13363656,  0.01041816,
        -0.06734404, -0.01930927,  0.04418375,  0.07379039, -0.07276778,
         0.0200979 , -0.09691442,  0.04933634,  0.18192419, -0.05266997,
        -0.03905399,  0.18436827,  0.02965726,  0.03922205,  0.05650592,
         0.03725109, -0.1144084 ,  0.05174316, -0.13884039,  0.01259667,
         0.07645657,  0.00586876,  0.04584204,  0.08964485, -0.14732021,
         0.10082877, -0.01285281, -0.06113401,  0.02678572,  0.0255925 ,
        -0.12544405, -0.08447115,  0.14104123, -0.24952233,  0.17711404,
         0.09760315,  0.00649869,  0.17723347,  0.10303544,  0.08564837,
         0.04208636, -0.00089474, -0.20092249, -0.06145699,  0.07153322,
        -0.05500717,  0.14160316,  0.05161323])]
```

Figura 17. Aplicación del descriptor 128D a una imagen del dataset.

Fuente: Elaboración propia

Face_compares

El procedimiento para la detección que aplica Dlib lo realiza con el algoritmo euclidiano que mide la distancia que existe entre cada codificación del rostro para luego comparar estas mediciones con otro, si es que existen similitudes entre esas instancias el sistema va a predecir de que es una persona conocida, pero si esos valores no coinciden se la presenta como una persona que es desconocida. La red convolucional está constituida por 29 capas que previamente ha sido entrenada con 13 millones de rostros lo que garantiza que existe una precisión del 99.38% en el momento de detectar e identificar a un rostro humano [36].

El método que usa dlib junto a face_recognition es el de compare_faces, en la cual se puede manipular los argumentos siguientes:

known_face_encodings: se ubica una lista de códigos faciales que ya se hayan almacenado.

face_encoding_to_check: El código de un solo rostro que se quiera comparar con la lista conocida de rostros, ya sea desde un video en tiempo real, o imagen.

tolerance: La distancia entre rostro y rostro se deben tomar en cuenta una coincidencia, cuanto más bajo es más estricto según la documentación en 0,6 es el mejor rendimiento típico que se haya logrado, esta puede ir variando, reducirlo cuando haya rostros similares.

Al final retorna un arreglo con valores de verdaderos y falsos las cuales muestran que el código facial que se haya almacenado y al que se desea verificar coincide con la codificación facial.

Este algoritmo maneja como clasificador a vecinos más cercanos (knn***)*** un algoritmo de (ML) machine learning (aprendizaje automático) supervisado, el cual se fundamenta en que las instancias en un grupo de datos existirán cerca de otras instancias que poseen características semejantes y dará como resultado en que tenga más votos [41].

La distancia euclidiana es medida por la siguiente fórmula: $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ y el resultado es la raíz cuadrada de la suma del cuadrado de la resta de entre dos puntos.

```
[0.48591186 0.46299603 0.50595269 0.58865697 0.50374742 0.57377515
 0.60940132 0.59484976 0.53868434 0.44956091 0.55281318 0.5579266
 0.51417914 0.45668711 0.54227672 0.43885154]  <=0.55
[0.5907324  0.57871005 0.49655626 0.60505887 0.5084872  0.56088811
 0.56044934 0.57735566 0.54027935 0.52886398 0.57798064 0.59915044
 0.5385974  0.50332902 0.51385496 0.51396735]  <=0.55
```

Figura 18. Aplicación de clasificador knn para cálculo de distancia euclidiana en una imagen de rostro con una tolerancia de 0.55 %.

Fuente: Elaboración propia

En el caso se no utilizar este método se puede optar por desarrollar una implementación de soporte vectorial en maquina (SVM) (Support Vector Machine) que representa un procedimiento de aprendizaje supervisado que crea funcionalidades para la clasificación desde un grupo de datos que han sido entrenados [42].

2.2.9.6. ChatBot

Un Chatbot es un representante inteligente capaz de interactuar con un cliente para contestar a una secuencia de cuestiones y dar la contestación idónea [43].

El chatbot utilizado en este proyecto usa (PLN) procesamiento de lenguaje natural campo destacado en la inteligencia artificial (IA), en la que ayuda al ordenador a comprender el escrito como lo realizan los humanos. [30].

2.2.9.7. Mysql

MySQL se lo denomina como un gestor de bases de datos relacional que es usado actualmente y es de mucha popularidad [44]. Este es un sistema que permite la gestión de las bases de datos y posee una licencia código abierto y en otra de una versión para compra de compañía de Oracle.

2.2.9.8. Jetson Nano 2 GB

Los Jetsons Nano poseen el sistema operativo Linux, lo cual posibilita la utilización de muchas bibliotecas útiles para el procesamiento de imágenes, entre ellas las más usadas es OpenCV una biblioteca de visión por computador multiplataforma [25].

Tarjeta desarrollada por NVIDIA que posibilita llevar a cabo ejecuciones de software de IA (inteligencia artificial) como redes neuronales, aprendizaje automático (ML), aprendizaje profundo para la clasificación de imágenes, detección de objetos, procesamiento voz, reconocimiento facial y varias.

Para este proyecto se optó por la selección del modelo de Jetson Nano 2Gb Developer Kit que según Nvidia es primordial para aprender, construir inteligencia artificial (IA) y para la robótica, elaborado para desarrolladores con un entorno de Linux, dando un rendimiento de inteligencia artificial impresionante a un costo bajo para generar productos de inteligencia artificial innovadores en cada una de las industrias [45].

Se la puede comparar con una Raspberry Pi ya que esta tiene exactamente la misma iniciativa, excepto que el Jetson Nano tiene una GPU Nvidia integrada. Puede llevar a cabo aplicaciones aceleradas por GPU como modelos de aprendizaje profundo muchísimo más acelerado que una placa como Raspberry Pi, que no posee una GPU integrada que sea compatible con la mayor parte de los marcos de aprendizaje profundo (Deep Learning).

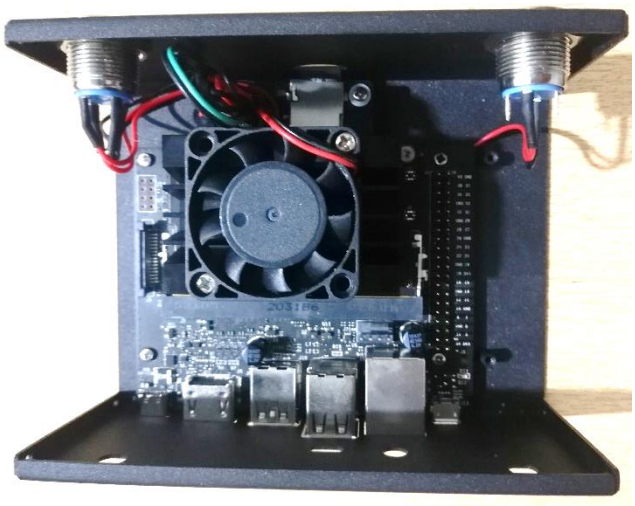


Figura 19. Nvidia Jetson Nano 2Gb.

Fuente: Elaboración propia

A continuación, se presentan sus características técnicas.

Tabla 2. Especificación técnica Jetson Nano 2 Gb

Especificación Técnica NVIDIA Jetson Nano 2gb Developer Kit	
GPU	128-core Nvidia Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memoria	2 GB 64-bit LPDDR4 25.6 GB/s
Almacenamiento	microSD (Tarjeta no incluida)
Codificación de video	4kp30 | 4x 1080p30 | 9x720p30 (H.264/H.265)
Decodificación de video	4Kp60 | 2x 4kp30 | 8x 1080p30 | 18x 720p30 (H.264/H.265)
Conectividad	Gigabit Ethernet 802.11ac wireless[t]
Cámara	1x MIPI CSI-2 connector
Mostrar	HDMI
USB	1xUSB 3.0 Type A, 2x USB 2.0 Type A, 1xUSB 2.0 Micro-B
Otros	12-pin Header (Power and related signals, UART) 40-pin header (GPIO, I²C, I²S, SPI, UART) 4-pin Fan Header[t]
Mecánica	100mm x 80 mm x 29 mm

Fuente: Elaboración propia

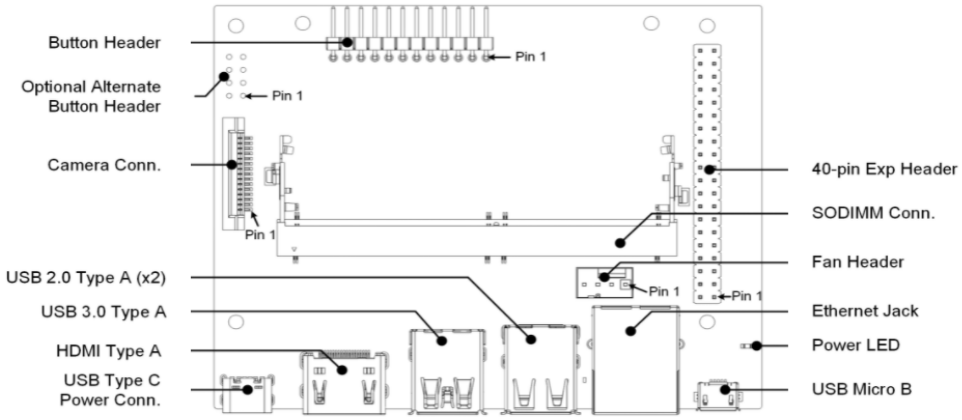


Figura 20. Diseño de placa Jetson nano

Fuente: [45]

2.2.9.9. Cámara de video

Cámara Web o Webcam se la denomina como una cámara digital conectada a un dispositivo electrónico, como PC, laptop o dispositivos embebidos, la cual puede capturar video, imágenes y transmitirlas por medio de una red privada, externa o a Internet.

Figura 21. Cámara Full HD USB
Fuente: Elaboración propia

Especificaciones Técnicas

- Resolución: Full HD (1080p a 30 fps)
- Micrófono cancela ruido.

2.2.9.10. Cerradura eléctrica

Una cerradura o chapa eléctrica es un sistema electromecánico que mediante la electricidad y el magnetismo en sus bobinas permite la apertura y cierre de una puerta. Estas pueden se aprovecharlas remotamente, acostumbran tener un costo más alto que las convencionales sin embargo ofrecen una mayor seguridad y bienestar [46].

Su funcionamiento simple para abrir puertas es dado a que cuando se encuentra en posición de reposo esta permanece cerrado dejando a la puerta bloqueada, por lo cual al recibir un impulso eléctrico el mecanismo se desbloquea y permite la apertura. En su interior posee una doble bobina, un pestillo y el expulsor.

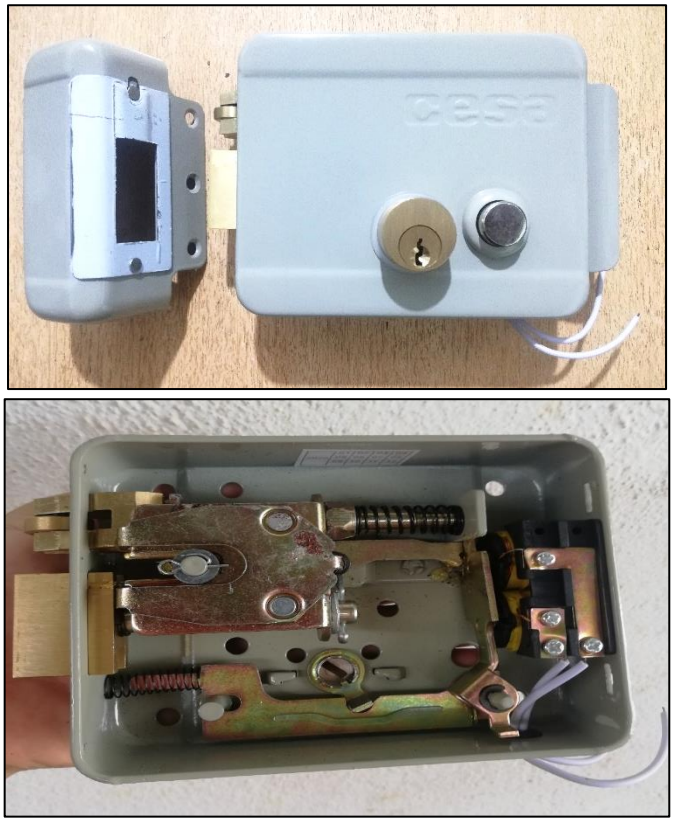


Figura 22. Cerradura Eléctrica.

Fuente: Elaboración propia

Especificaciones Técnica

- Voltaje de entrada 12 V
- Encendido normal.

2.2.9.11. Relay

El relé posibilita el paso de la corriente eléctrica más bien funciona como un interruptor en estado abierto y cerrado, sin embargo es accionado eléctricamente, no manualmente [47].

Está compuesto de una bobina conectada a una corriente, cuando la bobina se activa produce un campo electromagnético que hace que el contacto del relé que está usualmente abierto se cierre y permita el paso de la corriente por un circuito para lograr encender una lámpara o cualquier otro dispositivo.

Para implementar como interruptor de acceso para este proyecto se usará un módulo de relé de 1 canal, que posee placa de protección de interfaz para PIC AVR DSP ARM MCU Arduino y disparador de bajo nivel de 5V, modelo JQC-3FF-S-Z.

Figura 23. Relé

Fuente: Elaboración propia

Especificaciones Técnicas

- Voltaje para entrada: 5 Volts
- Voltaje para control: 3 a 9 Volts
- Voltaje para la salida: 250 VCA / 30 VDC máx.
- Corriente de salida: 10 A.
- Disparador de bajo nivel.
- Dimensiones: 38x12x12mm
- Peso: 13 g

2.3. Objetivos del prototipo

2.3.1. Objetivo General

- Desarrollar un sistema con reconocimiento facial y chatbot para la gestión y control de seguridad de una residencia, mediante aprendizaje profundo en un dispositivo jetson nano programado en Python.

2.3.2. Objetivos Específicos

- Desarrollar una aplicación de consola que permita registrar y realizar capturas de rostros de las personas implementando algoritmos de visión e inteligencia artificial.
- Desarrollar un sistema de video vigilancia en tiempo real para visualizar las visitas en la puerta de una residencia.
- Desarrollar un sistema que permita el acceso a personas conocidas previamente registradas utilizando reconocimiento facial.
- Registrar la información de vigilancia y acceso en un gestor de base de datos conectada en una red de área local.
- Utilizar un chatbot que permita gestionar la entrada, visualizar el historial de acceso y vigilancia de una residencia.

2.4. Diseño del prototipo

2.4.1 Componentes y dispositivos

Los dispositivos electrónicos utilizados en el desarrollo del prototipo de sistema domótico con reconocimiento facial y chatbot para la gestión de la seguridad de una residencia, están detallados en la Tabla 3.

Tabla 3. Dispositivos para prototipo.

Dispositivos	
NVIDIA Jetson Nano 2gb	2 GB 64-bit LPDDR4 25.6 GB/s / Quad-core ARM A57 @ 1.43 GHz / 128-core Nvidia Maxwell / Ubuntu LTS 18.04
microSD	32Gb
Servidor Local (Laptop HP)	Pc 64-bit i7-7500u / 16 Gb DDR4 / Windows 10 Pro.
Router Huawei EG8141A5	Puertos: 1 telefonía + 1 GE y 3 FE + 1 USB + Wi-FI.
Webcam Video Uokier	Full HD (1080p a 30 fps)
Monitor LG	Full HD (1080p a 75 Hz)
Adaptador Wifi Usb – Tp-link	150 Mbs
Cables Hembra	3 líneas flexibles tipo bus de datos
Relay	JQC-3FF-S-Z. 5v
Chapa Eléctrica	Voltaje de entrada a 12v
Cargador alimentador para Jetson Nano	Fuente de alimentación de 5.1V 3A DC
Cargador alimentador para Chapa eléctrica	Voltaje de salida a 12v

Fuente: Elaboración propia

Para la parte de la implementación del sistema se toma como base el diagrama que está definido en la Figura 1, donde se explican las 4 etapas para el reconocimiento facial.

En el avance de cada capítulo, se va presentando el esquema de cada módulo del sistema, la tecnología usada, técnica de reconocimiento facial. La figura 25 presenta el diseño usual del sistema de reconocimiento facial y chatbot para la gestión de la seguridad de una residencia.

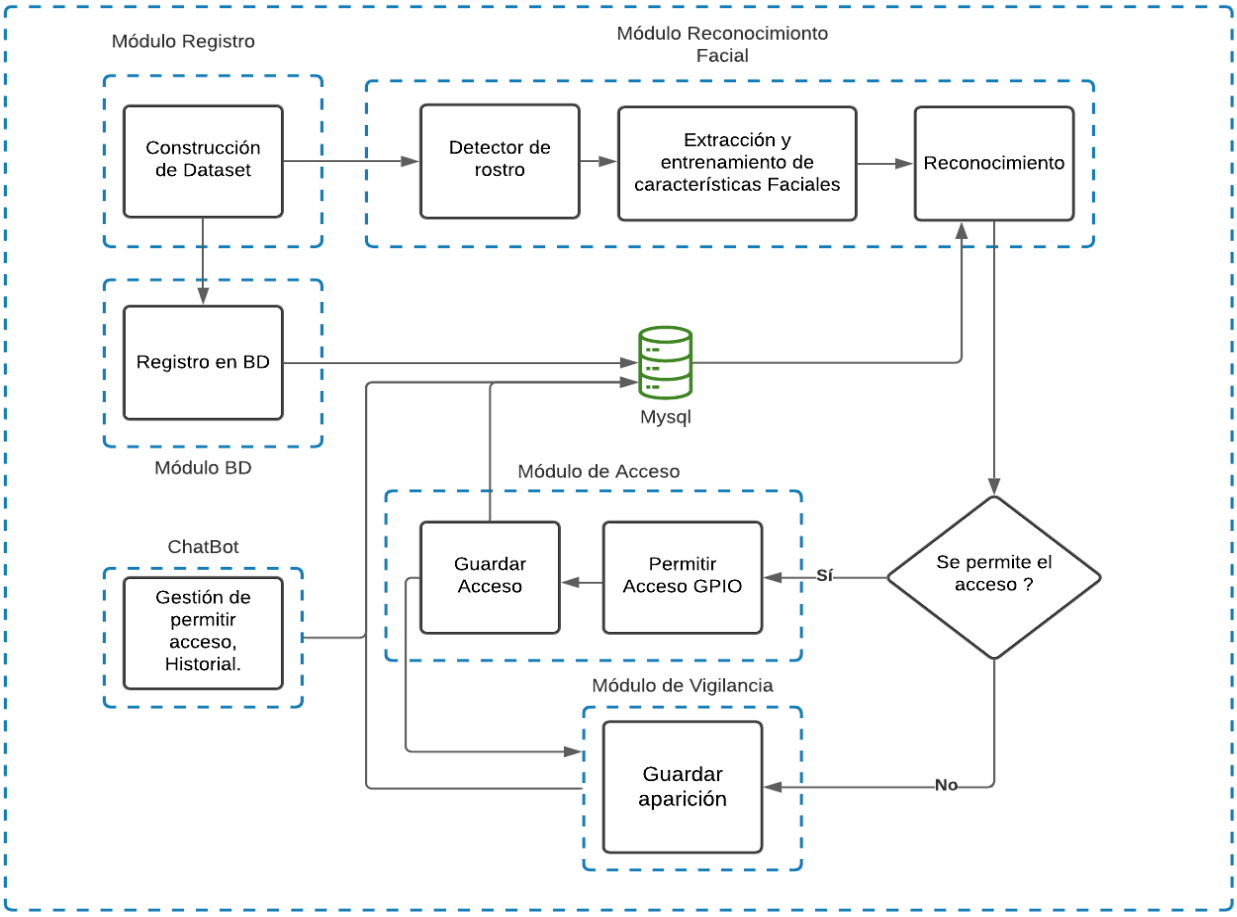


Figura 25. Diseño general del prototipo para la gestión de la seguridad de una residencia con reconocimiento facial y chatbot.

Fuente: Elaboración propia

2.4.2 Diseño de Base de datos

El planteamiento de la arquitectura de este proyecto en la figura 1, la base de datos será gestionada en un servidor local, con el fin de no consumir los recursos de la jetson nano, en la tabla 4 se presentan las herramientas a usar para administrar de la base de datos.

Tabla 4. Herramientas para la gestión de la base de datos.

Herramientas	Descripción
Servidor de base de datos	MySQL 10.4.8
Entorno de diseño de bases de datos	MySQL Workbench 8.0.7

Fuente: Elaboración propia

Para poder almacenar en nuestra base de datos la información de quienes ingresan por la puerta del hogar, llevar un registro de quienes visitan el hogar sean o no conocidos, se utilizará 4 tablas denominadas, registro, acceso, vigilancia, gestión.

A continuación, se describen los argumentos que llevan cada tabla y su función en el mismo.

Tabla 5. Argumentos de tabla registro en MySQL.

Argumentos	Descripción
Identificación	Campo de caracteres, generada como clave única para cada fila, la cual servirá como número de identificación de personas registradas de la residencia para la construcción del dataset y en el reconocimiento facial.
nombre	Campo de caracteres, que recepta los nombres de la persona registrada.
apellido	Campo de caracteres, que recepta los apellidos de la persona registrada.
parentesco	Campo de caracteres, que recepta el parentesco que se tenga con los dueños de la residencia.
captura_rostro	Campo binario, que recepta la foto del rostro de la persona registrada.

Fuente: Elaboración propia

Tabla 6. Argumentos de tabla vigilancia en MySQL

Argumentos	Descripción
Id	Campo numérico, que servirá como clave única para cada fila, esta se incrementa automáticamente en cada nuevo registro.
Fecha_hora	Campo DateTime, donde se almacenará la fecha y hora de la vigilancia en la puerta, indicando el año, mes y día en el siguiente formato "YY / mm / dd", y el tiempo: hora, minutos, segundos en el formato "H:M:S"
estado	Campo de caracteres, que indica si la persona es "CONOCIDA" o "DESCONOCIDA"
identificación	Campo de caracteres, como clave foránea, que recepta el número de identificación de la persona registrada para guardarla en vigilancia.

Fuente: Elaboración propia

Tabla 7. Argumentos de tabla acceso en MySQL.

Argumentos	Descripción
Id	Campo numérico, que servirá como clave única para cada fila, esta se incrementa automáticamente en cada nuevo registro.
identificación	Campo de caracteres, como clave foránea, que recepta el número de identificación de la persona registrada para guardarla en acceso.
Fecha_hora	Campo DateTime, donde se almacenará la fecha y hora en que se otorgó el acceso por la puerta en la residencia, indicando el año, mes y día en el siguiente formato "YY / mm / dd", y el tiempo: hora, minutos, segundos en el formato "H:M:S"

Fuente: Elaboración propia

Tabla 8. Argumentos de tabla gestión en MySQL.

Argumentos	Descripción
Id	Campo numérico, que servirá como clave única para cada fila, esta se incrementa automáticamente en cada nuevo registro.
acceso	Campo de caracteres, donde se almacenará el valor "SI" o "NO" para permitir el acceso a la residencia.

Fuente: Elaboración propia

La relación que existe entre las tablas se las puede visualizar en la figura 26, en la cual existe una relación de 1: N entre las tablas registro y acceso, haciendo referencia a que una persona conocida tiene de uno a muchos accesos diarios en la residencia. Relación de 1: N entre las tablas registro y vigilancia, cuando la persona conocida

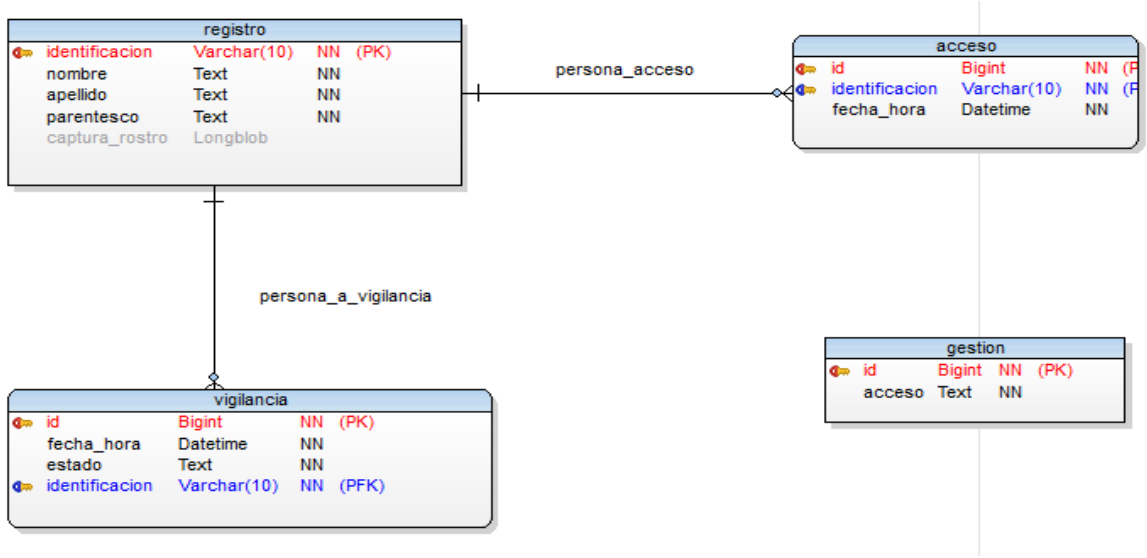


Figura 26. Diagrama Entidad Relación de base de datos del sistema.

Fuente: Elaboración propia

En el servidor local se realiza la instalación de MySQL conjuntamente con la herramienta visual de diseño de bases de datos Workbench, se crea la base de datos denominado "sistema_domotica_bd" y conjuntamente la creación de las tablas, a continuación, se presentan las sentencias DDL para cada tabla.

Creación de tabla registro

```
DROP TABLE IF EXISTS `registro`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `registro` (
  `identificacion` varchar(10) NOT NULL,
  `nombre` text NOT NULL,
  `apellido` text NOT NULL,
  `parentesco` text NOT NULL,
  `captura_rostro` longblob,
  PRIMARY KEY (`identificacion`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Creación de tabla vigilancia

```
DROP TABLE IF EXISTS `vigilancia`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `vigilancia` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `fecha_hora` datetime NOT NULL,
  `estado` text NOT NULL,
  `identificacion` varchar(10) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `identificacion_fk_idx` (`identificacion`),
  CONSTRAINT `identificacion_fk_v` FOREIGN KEY (`identificacion`)
  REFERENCES `registro` (`identificacion`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Creación de tabla acceso

```
DROP TABLE IF EXISTS `acceso`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `acceso` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `identificacion` varchar(10) NOT NULL,
  `fecha_hora` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `identificacion_fk_idx` (`identificacion`),
  CONSTRAINT `identificacion_fk` FOREIGN KEY (`identificacion`)
  REFERENCES `registro` (`identificacion`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Creación de tabla gestión.

```
DROP TABLE IF EXISTS `gestion`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `gestion` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `acceso` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

Para consultar al chatbot de acuerdo a las preguntas que se le haga como de saber el historial de acceso, vigilancia de conocidos y desconocidos, para ello se necesita de 3 vistas, A continuación, se presentan las sentencias DDL que se utilizaron para la creación de las vistas.

Vista de acceso

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `sistema_domotica_bd`.`vista_acceso` AS
    SELECT
        `r`.`identificacion` AS `identificacion`,
        CONCAT_WS(' ', `r`.`nombre`, `r`.`apellido`) AS `nombres`,
        `r`.`parentesco` AS `parentesco`,
        CAST(`a`.`fecha_hora` AS DATE) AS `fecha`,
        CAST(`a`.`fecha_hora` AS TIME) AS `hora`
    FROM
        (`sistema_domotica_bd`.`registro` `r`
        JOIN `sistema_domotica_bd`.`acceso` `a`)
    WHERE
        ((`r`.`identificacion` = `a`.`identificacion`)
            AND (CAST(`a`.`fecha_hora` AS DATE) = (SELECT CURDATE())))
    ORDER BY CAST(`a`.`fecha_hora` AS TIME) DESC
```

Vista de conocidos

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `sistema_domotica_bd`.`vista_acceso` AS
    SELECT
        `r`.`identificacion` AS `identificacion`,
        CONCAT_WS(' ', `r`.`nombre`, `r`.`apellido`) AS `nombres`,
        `r`.`parentesco` AS `parentesco`,
        CAST(`a`.`fecha_hora` AS DATE) AS `fecha`,
        CAST(`a`.`fecha_hora` AS TIME) AS `hora`
    FROM
        (`sistema_domotica_bd`.`registro` `r`
        JOIN `sistema_domotica_bd`.`acceso` `a`)
    WHERE
        ((`r`.`identificacion` = `a`.`identificacion`)
            AND (CAST(`a`.`fecha_hora` AS DATE) = (SELECT CURDATE())))
    ORDER BY CAST(`a`.`fecha_hora` AS TIME) DESC
```

Vista de desconocidos

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `sistema_domotica_bd`.`vista_acceso` AS
    SELECT
        `r`.`identificacion` AS `identificacion`,
        CONCAT_WS(' ', `r`.`nombre`, `r`.`apellido`) AS `nombres`,
        `r`.`parentesco` AS `parentesco`,
        CAST(`a`.`fecha_hora` AS DATE) AS `fecha`,
        CAST(`a`.`fecha_hora` AS TIME) AS `hora`
    FROM
        (`sistema_domotica_bd`.`registro` `r`
        JOIN `sistema_domotica_bd`.`acceso` `a`)
    WHERE
        ((`r`.`identificacion` = `a`.`identificacion`)
            AND (CAST(`a`.`fecha_hora` AS DATE) = (SELECT CURDATE())))
    ORDER BY CAST(`a`.`fecha_hora` AS TIME) DESC
```

En la figura 27 se visualizan las tablas y vistas creadas luego de la ejecución de las sentencias DDL en MySQL.

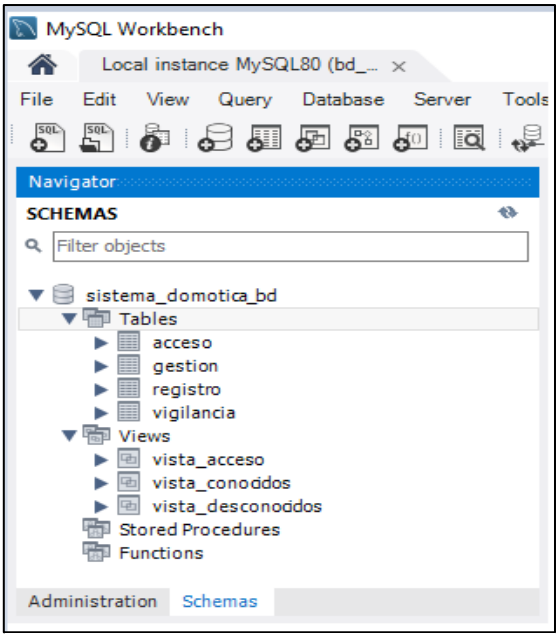


Figura 27. Tablas y vistas en MySQL

Fuente: Elaboración propia

2.4.3 Backend del Sistema

El diseño del sistema se lo desarrolló para modo consola, ya que por la limitada RAM de 2 GB que posee la Jetson Nano, durante el procesamiento de entrenamiento de las codificaciones del rostro y en el reconocimiento consumen memoria en su totalidad pasando al espacio de intercambio (memoria SWAP) de la tarjeta microSD por lo que el tiempo de lectura es muy lenta, por lo que desarrollar un aplicativo web con animaciones consumen exhaustivamente memoria en el contexto anterior.

En el aspecto del chatbot se lo desarrolló como aplicativo web ya que al finalizar el sistema de reconocimiento facial la memoria RAM reduce su consumo por lo que abrir en un navegador web con la dirección IP, el sistema si es factible, y su procesamiento lo hace desde el servidor local de donde este está alojado.

Para la programación del sistema se usó el editor de código fuente visual code, una laptop que es la misma que sirve como servidor local, las características se las menciona en la tabla 3, ya que tiene mayor capacidad de RAM que la jetson nano.

Previo a esto se hizo la activación y configuración del uso compartido de archivos e impresoras desde ambos sistemas operativos para que la jetson nano y el servidor local puedan enlazarse en una red local doméstica con el grupo de trabajo denominado “CASA”, haciendo uso de SAMBA y con ello se puedan compartir los archivos de código fuente que se tiene en el servidor local al sistema de archivos de la Jetson Nano. En el Anexo figura 87, se presenta la conexión con samba desde

Windows a Ubuntu y en el punto 2.5.2 del capítulo 3 se describe e instala las tecnologías de desarrollo como son las librerías y el entorno en la que se ejecuta el sistema.

La estructura del sistema se la presenta en la figura 28:

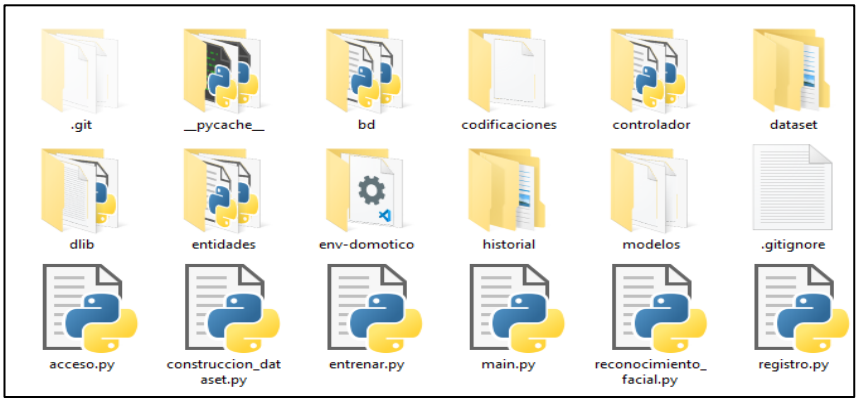


Figura 28. Estructura de código fuente del sistema.
Fuente: Elaboración propia

2.4.3.1 Conexión a Base de datos

El módulo de conexión para la base de datos MySQL al servidor local, el archivo se encuentra dentro del directorio bd del directorio principal del sistema, en este se define la dirección ip, el nombre de usuario, la contraseña, nombre de la base de datos y el número de puerto.

A continuación, se presenta un extracto del código fuente:

```
class BD:
    def __init__(self):
        # Servidor Local
        self.host ="192.168.100.40"
        self.usuario="bd_sistema_domotico"
        self.contrasenia="6Wa8!S7VqAk4"
        self.nombre_bd="sistema_domotica_bd"
        self.puerto = 3306

    def conectar(self):

        self.db = pymysql.connect(
            host= self.host,
            user= self.usuario,
            password= self.contrasenia,
            database= self.nombre_bd,
            port= self.puerto)

        print("Conección establecida: "+str(self.db))

        return self.db

    def desconectar(self):
        self.db.close
```

2.4.3.2 Registro

En esta sección se explica el módulo de registro, el cual servirá para almacenar la información de las personas conocidas del hogar.

Las bibliotecas necesarias para realizar el script del registro de las personas y almacenarlas en la base de datos se presentan en la Tabla 9.

Tabla 9. Librerías para el módulo registro.

Librería	**Función**
pymysql	Conexión MySQL.

A continuación, en la figura 29, se presenta el diagrama de flujo que integran el módulo de registro.

Diagrama de fujo del algoritmo registro.py

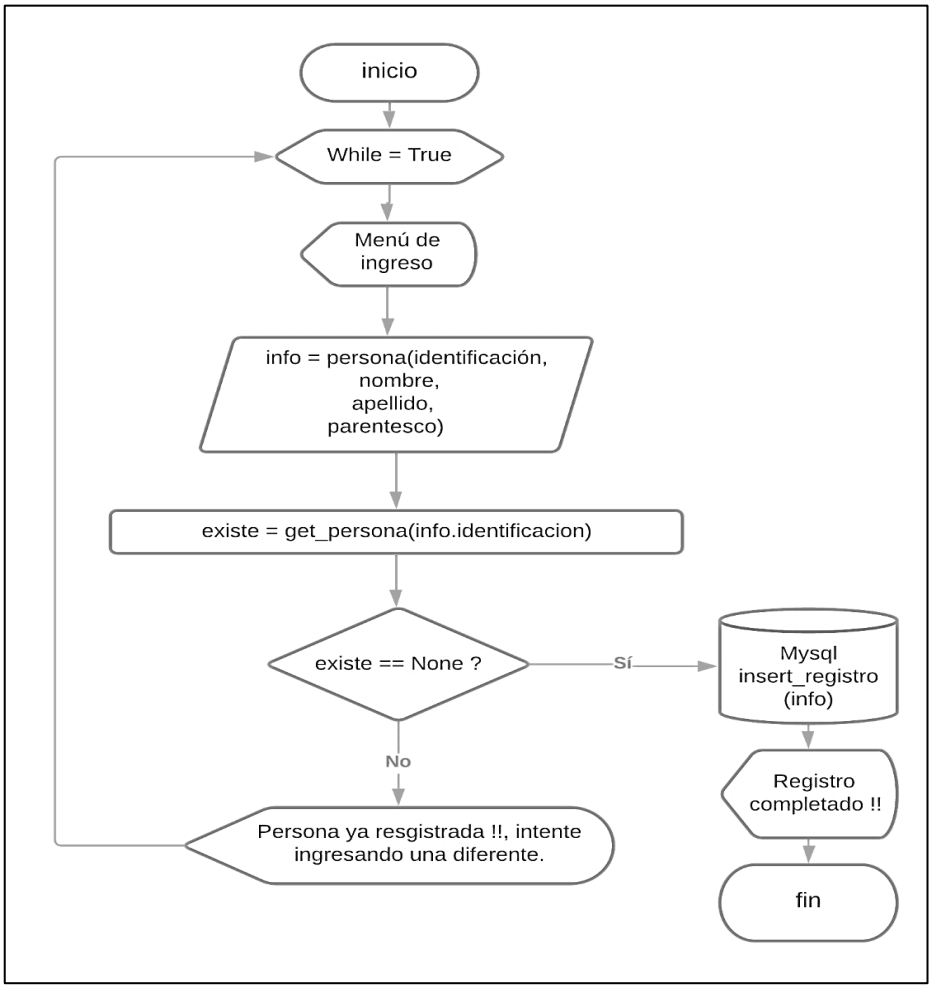


Figura 29. Diagrama de flujo módulo registro.

Fuente: Elaboración propia

El registro se lo realiza con la ayuda de una interfaz por medio de consola programada en Python, en ella se le pide al usuario el ingreso de la información en este caso el número de identificación, nombres y apellidos de la persona y el parentesco que tenga con los propietarios de la residencia.

A continuación, una parte del código

```
    def registrar(self):
        print("[INFO]  *************** Registro de personas del
hogar y conocidos.. **************")
        print(" --- Registro de nueva persona..")
        identificacion = input('---► Ingrese identificacion: ')
        nombre = input('---► Nombres: ')
        apellido = input('---► Apellidos: ')
        parentesco = input('---► Parentesco: ')

        persona = Persona(identificacion, nombre, apellido,
parentesco, NULL)
        return persona
```

Luego se almacena la información en la base de datos del sistema en el modelo de registro donde esta las sentencias DML para la inserción del registro, a continuación, un fragmento del código sql desde Python.

```
        self.db = coneccion.BD().conectar()
        cursor = self.db.cursor()
        sql = "INSERT INTO registro(identificacion, nombre,
apellido, parentesco, captura_rostro) \
        VALUES (%s,%s,%s,%s, NULL);"
        args = (persona.identificacion, persona.nombre,
persona.apellido, persona.parentesco)
        try:
            cursor.execute(sql, args)
            self.db.commit()
            self.db.close()
        except:
            print("Error al insertar")
            self.db.rollback()
```

2.4.3.3 Construcción de dataset.

En esta sección se explica el módulo de construcción del dataset necesario para el procesamiento de las codificaciones del rostro y/o entrenamiento.

Par esto, los requerimientos tecnológicos a usar, son los dispositivos mencionados anteriormente Jetson nano y la cámara Full Hd de conexión USB que se encargará de recopilar las imágenes del rostro.

Las bibliotecas necesarias para realizar las capturas del rostro y almacenarlas en disco y en una base de datos se muestran en laTabla 10.

Tabla 10. Librerías para la construccion_dataset

Librería	Función
imutils	Para procesamiento de imágenes.
face_recognition	Cargar modelo de detección de rostro
cv2	Para analizar y procesar imágenes con el cuadro delimitador del rostro, almacenamiento en disco
os	Para realizar operaciones en el sistema operativo, creación y ubicación de carpetas.
WebcamVideoStream	Permite iniciar video stream de la cámara usb.
pymysql	Conexión para la base de datos MySQL.

Fuente: Elaboración propia

En Python se procede a la importación de las bibliotecas mencionadas, a continuación, se visualiza el procedimiento:

```
import imutils
import face_recognition
import cv2
import os
from imutils.video import WebcamVideoStream
from pymysql import NULL
```

Luego de haber importado se realiza la codificación de los procedimientos, previamente la figura 30 explica el diagrama de flujo de los diferentes procedimientos del módulo de construcción de dataset.

Diagrama de fujo del algoritmo construcción_dataset.py

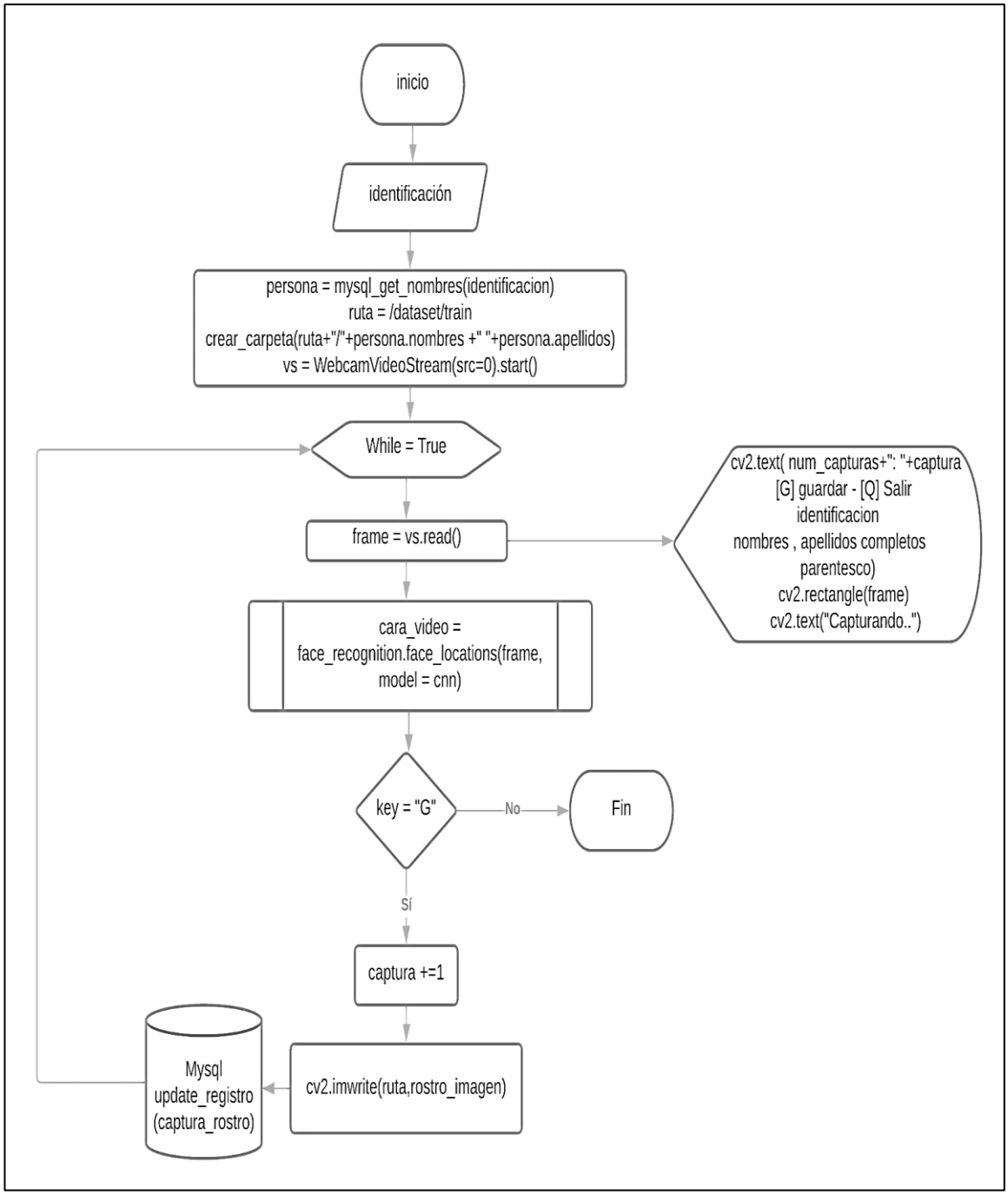


Figura 30. Diagrama de flujo módulo construccion_dataset.py

Fuente: Elaboración propia

La recopilación de las imágenes se las realiza con la ayuda de una interfaz GUI programada en Python, en ella se presentará el número de capturas realizadas en ese momento, las opciones de teclado: [G] para guardar la captura, [Q] para salir del frame, la información de la persona: su identificación, nombres completos, parentesco, y el cuadro delimitador de la detección del rostro con un etiquetado “Capturando” en la parte superior.

El directorio de salida de las imágenes, por organización y entendimiento, es conveniente separar las imágenes en subdirectorios etiquetadas con el nombre de las personas.

```
      self.BASE_DIR = os.getcwd()
        # Ruta de almacenamiento de las captura de rostros en
carpeta para luego entrenar
        self.ruta_train = self.BASE_DIR+'/dataset/train'

        # Etiqueta de carpeta
        self.persona_nombre = 'Desconocido'
        # Ruta de carpeta con la etiqueta creada de la persona a
registrar
        self.persona_ruta = self.ruta_train + '/' +
self.persona_nombre
```

Antes de almacenar las imágenes de rostro primero se necesita detectarlas. En la librería de face_recognition, trae dos métodos de detección de rostros incluyen uno llamado “HOG” con menos precisión, y la “CNN” con mayor precisión ya que esta hace uso de la GPU.

Para este proyecto se hace uso del modelo pre-entrenado MMOD, “CNN” ya que la Jetson Nano integra una GPU. En el api ya viene preconfigurado el modelo, al momento de llamar el método face_locations este inicia la importación con el uso de la función cnn_face_detector_model_location() que contiene la ruta de archivo indicando la carpeta y el nombre del modelo: “models/mmod_human_face_detector.dat”, por lo que la importación lo hace automáticamente.

```
frame = vs.read()
frame = imutils.resize(frame,width=640)
frame_auxiliar = frame.copy()
# Redimensionar el fotograma del vídeo a 1/4 de tamaño para un
procesamiento más rápido del reconocimiento facial
reducido_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
rgb_reducido_frame = reducido_frame[:, :, ::-1]
# detectar las coordenadas (x, y) de los cuadros delimitadores
# correspondientes a cada rostro en el cuadro de entrada, y luego
calcular
# detección de rostro
#. El número 2 en el segundo argumento indica que se muestrea la
imagen 2 veces.
cara_video = face_recognition.face_locations(rgb_reducido_frame,
number_of_times_to_upsample= 2, model="cnn")
```

La variable cara_video devuelve un arreglo de objetos que es el cuadro delimitador o área rectangular del rostro en el video en la que contienen funciones (top, right, bottom, left) que retornan las coordenadas de todas las ubicaciones.

```
# bucle sobre las caras reconocidas
for (top, right, bottom, left) in cara_video:
     # reescalar las coordenadas de la cara
      top *= 4
      right *= 4
      bottom *= 4
      left *= 4

      cv2.rectangle(frame, (left, top), (right, bottom),(0, 255, 0),
2)

      y = top - 15 if top - 15 > 15 else top + 15
                cv2.putText(frame, "Capturando..", (left, y),
      cv2.FONT_HERSHEY_PLAIN,1, (0, 255, 0), 2)
      cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 0), 1, cv2.LINE_AA, False)
```

Para el proceso de guardar en disco y en Base de datos, se configura la ruta de acceso al directorio de salida. Las imágenes de las caras se almacenarán en este directorio. Luego se inicializa un total que representa el número de imágenes faciales almacenadas en el disco.

```
if key == ord("G") or key == ord("g"):
    ruta = os.path.sep.join([
    self.persona_ruta,"{}.jpg".format(str(captura).zfill(5))])
    cv2.imwrite(ruta, face_image)
    captura += 1
```

2.4.3.4 Extracción y entrenamiento de características o codificaciones de rostro.

En esta sección se explica el módulo de extracción y entrenamiento de características o codificaciones del rostro, como se indicaba en el capítulo 1, en realidad no se hace un entrenamiento, sino el modelo que integra face_recognition denominado face_encondings ya está pre-entrenado para devolver un vector de 128 dimensiones con valores reales de las codificaciones del rostro.

La extracción se desarrolla con la ayuda de una interfaz por medio de consola programada en Python, en ella se le pide al usuario seleccionar la opción de entrenar, para luego realizar el proceso.

Durante el proceso se va presentando en consola, la cantidad de imágenes procesadas por cada persona registrada. Al finalizar se guardará las codificaciones en la carpeta “codificaciones” en esta se lo serializa en el disco como “codificaciones.pickle”

Las bibliotecas necesarias para realizar la extracción de las características faciales y almacenarlas en disco se muestran en la Tabla 11.

Tabla 11. Librerías para extracción de incrustaciones del rostro.

Librería	**Función**
imutils	Para procesamiento en imágenes.
face_recognition	Cargar modelo de detección de rostro
cv2	Para analizar y procesar imágenes con el cuadro delimitador del rostro, almacenamiento en disco
os	Para realizar operaciones en el sistema operativo, creación y ubicación de carpetas.
pickle	Para convertir las codificaciones en una secuencia de bytes y almacenarlas en disco.
pymysql	Conexión para la base de datos MySQL.

Fuente: Elaboración propia

En Python se procede a la importación de las bibliotecas mencionadas, a continuación, se visualiza el procedimiento:

```
from imutils import paths
import face_recognition
import pickle
import cv2
import os
from pymysql import NULL
```

A continuación, se presenta en la figura 3 el diagrama de flujo de los diferentes procedimientos del módulo de extracción de codificaciones del rostro.

Diagrama de fujo del algoritmo entrenamiento.py

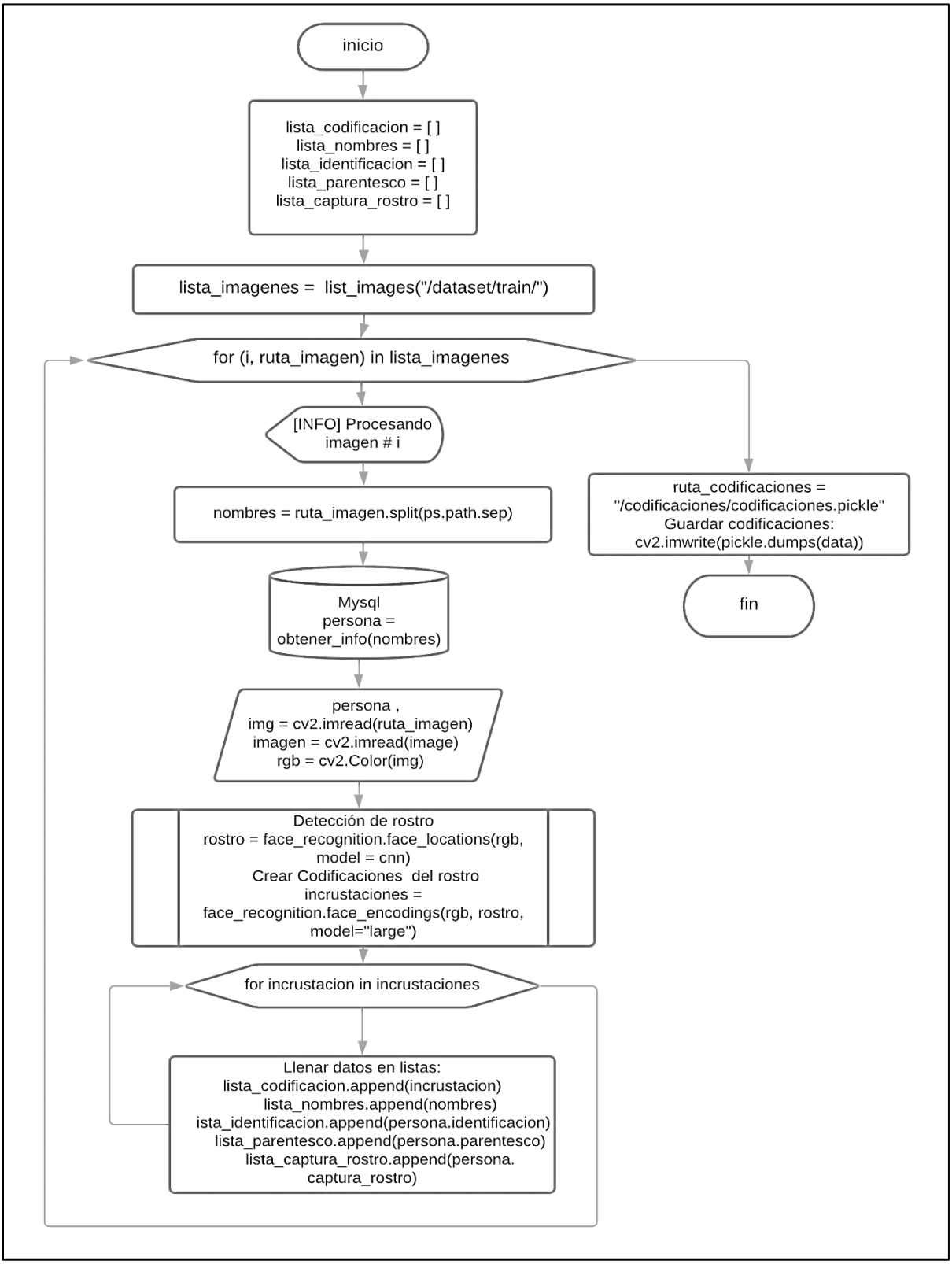


Figura 31. Diagrama de fujo del algoritmo entrenamiento.py

Fuente: Elaboración propia

El proceso de entrenamiento inicia luego que se haya creado el dataset de los distintos rostros de las personas registradas.

Se utiliza la ruta a este directorio del conjunto de datos de entrada para poder crear una lista de todos los contenidos de él. También se necesita inicializar 5 listas, estas contendrán las codificaciones, los nombres, el número de identificación, parentesco y captura del rostro, correspondientes a cada persona del dataset.

```
print("[INFO]  Cuantificando rostros ...")
    lista_imagenes =
list(paths.list_images(BASE_DIR+"/dataset/train/"))
```

El bucle ciclará las veces del número de imágenes almacenadas de caras en el conjunto de datos. Es decir, se está recorriendo los caminos de cada una de las imágenes, a partir de allí se extrae el nombre completo de cada persona que es el mismo del nombre del directorio.

Para cada iteración, se va ir detectando cada cara, por ello se utiliza nuevamente el método face_locations, donde encuentra y/o localiza las caras en ella, lo que resulta en una lista de caras, como argumento se le especifica la imagen anticipadamente procesada con el número de muestreo por defecto que indica que se muestrea la imagen 1 sola vez para evitar excesivamente el consumo de memoria de GPU y de RAM.

Luego, cada rostro es convertido en una lista de 128 números reales, esto lo que se está haciendo es la codificación de los rostros y el método que maneja dentro de los métodos de face_recognition es el llamado face_encodings. Esto se va ir agregando en la lista de codificaciones. Previo se realiza en la base de datos una consulta para que devuelva la información de cada persona y almacenarlas en las listas restantes mencionadas anteriormente, para posteriormente poderla serializar en disco.

```
def entrenar_rostro():
    # Se toma la ruta de las imágenes de entrada en nuestro conjunto
de datos
    print("[INFO]  Cuantificando rostros ...")
    lista_imagenes =
list(paths.list_images(BASE_DIR+"/dataset/train/"))


    # bucle sobre las rutas de las imágenes
    for (i, ruta_imagen) in enumerate(lista_imagenes):

        # extraer  nombres de la persona de la ruta de la imagen
        print("[INFO] Procesando imagen {}/{}".format(i +
1,len(lista_imagenes)))
```

```
        nombres = ntpath.basename(ruta_imagen.split(os.path.sep)[-2])

        # Hace una consulta para poder recibir los datos de la persona
por medio de nombres
        persona =
Crud_registro().get_persona_x_nombres_apellidos(nombres)


        # cargar la imagen de entrada y convertirla de BGR (ordenación
OpenCV)
        # a la ordenación dlib (RGB)
        img = cv2.imread(ruta_imagen)
        rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        # Detectar las coordenadas (x, y) del rostro delimitador
        # correspondientes a cada rostro de la imagen de entrada

        # Detector de rostros con dlib con modelo cnn MMDO (Max-Margin
Object Detection). Usa la GPU por lo que mejora la detección
        rostro = face_recognition.face_locations(rgb,
number_of_times_to_upsample=1, model="cnn")


        # Envia para crear las incrustaciones de la cara
        codificaciones(rgb,rostro,nombres, persona = persona)
```

En la salida, se tendrá un archivo denominado “codificaciones.pickle”, este contendrá las incrustaciones de rostros 128-d, para cada de las personas del conjunto de datos dataset. Los tiempos de procesamiento de la Jetson Nvidia toma tiempo dependiendo de la cantidad de imágenes, pero gracias a la GPU que viene integrada los tiempos de espera disminuyen.

```
def codificaciones(rgb, rostro, nombres, persona: Persona):

    # calcular la incrustación facial para la cara
    incrustaciones = face_recognition.face_encodings(rgb, rostro,
model="large")

    #print(nombres)
    # bucle sobre las codificaciones
    for incrustacion in incrustaciones:
        # añadir cada codificación +  nombres de conocidos,
        # la identificacion, el parentesco y captura de rostro a nuestro
conjunto de datos
        lista_codificacion.append(incrustacion)
        lista_nombres.append(nombres)
        lista_identificacion.append(persona.identificacion)
        lista_parentesco.append(persona.parentesco)
        lista_captura_rostro.append(persona. captura_rostr

# Guarda todas las codificaciones faciales + y datos al disco
    print("[INFO] Serializando encodings...")
    data = {
            "codificacion": lista_codificacion,
            "nombres": lista_nombres,
            "identificacion": lista_identificacion,
            "parentesco": lista_parentesco,
            "captura_rostro" : lista_captura_rostro
            }


    f = open(BASE_DIR+"/codificaciones/codificaciones.pickle", "wb")
    f.write(pickle.dumps(data))
    f.close()
```

2.4.3.5 Reconocimiento Facial

En esta sección se explica el módulo de reconocimiento facial, el proceso final se presentará con la ayuda de una interfaz de ventana programada en Python y OpenCv. En el proceso se hace coincidir cada codificación de rostro en la codificación de la imagen de entrada. En la figura se muestra el proceso previo para posteriormente realizar el reconocimiento facial

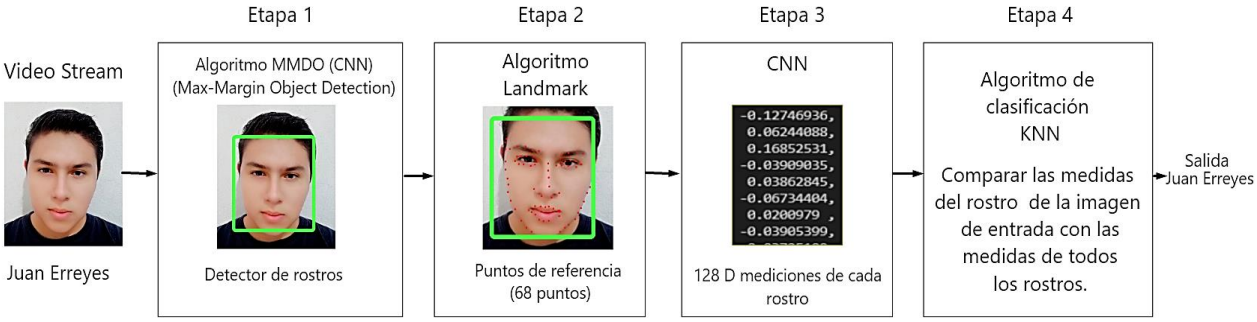


Figura 32. Etapas previas y posteriores para el reconocimiento facial.

Fuente: Elaboración propia

En este proceso se involucran los módulos de vigilancia y acceso, luego de haber detectado a una persona, esta sección se explicará más adelante.

Las bibliotecas necesarias para realizar la extracción de las características faciales y almacenarlas en disco se muestran en la Tabla 12.

Tabla 12. Librerías para el proceso de reconocimiento facial.

Librerías	**Función**
imutils	Para procesamiento de imágenes.
WebcamVideoStream	Permite iniciar video stream de la cámara usb.
face_recognition	Cargar modelo de detección de rostro
cv2	Para analizar y procesar imágenes con el cuadro delimitador del rostro, almacenamiento en disco
datetime	Para capturar la fecha y hora de hoy
time	Para capturar el número de segundos que han transcurrido.
os	Para realizar operaciones en el sistema operativo, creación y ubicación de carpetas.

Pil	Para abrir, manipular y guardar muchos formatos de archivo de imagen diferentes
pymysql	Conexión para la base de datos MySQL.

Fuente: Elaboración propia

A continuación, en la figura 33 se presenta el diagrama de flujo de los diferentes procedimientos del módulo de reconocimiento facial.

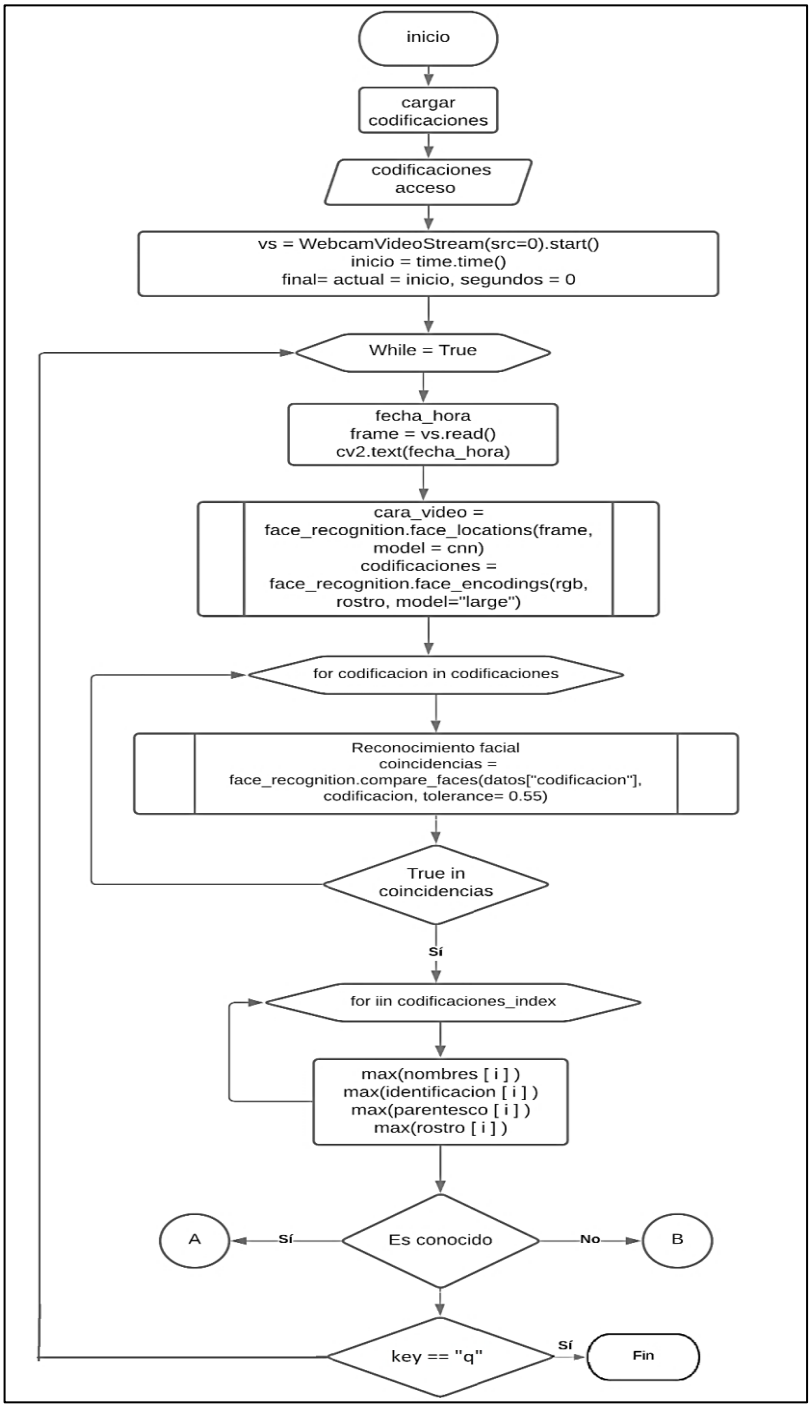


Figura 33. Diagrama de fujo del algoritmo reconocimiento_facial.py

Fuente: Elaboración propia.

Python se procede a la importación de las bibliotecas mencionadas, a continuación, se visualiza el procedimiento:

```
import cv2
from datetime import datetime, timedelta
import face_recognition
import time
import os
import imutils
from imutils.video import WebcamVideoStream
import io
from PIL import Image
from pymysql import NULL
```

Primeramente, se cargan las codificaciones generadas más la información de cada persona registrada. Estos datos se necesitarán más adelante durante el proceso de reconocimiento facial.

```
try:
        print("[INFO] Cargando codificaciones + detector facial
espere....")
        archivo = BASE_DIR+"/codificaciones/codificaciones.pickle"
        return  pickle.loads(open(archivo, "rb").read())
    except:
        print("[INFO] Modelo entrenado no existe en disco..")
```

En este proceso se hace coincidir la codificación de rostro que se obtiene en la imagen de entrada en este caso del video stream con las codificaciones que se serializaron en disco.

```
# detectar las coordenadas (x, y) de la cara delimitadora
# correspondientes a cada rostro en el cuadro de entrada,
rostro_video = face_recognition.face_locations(rgb_reducido_frame,
number_of_times_to_upsample=2, model="cnn")
# calculo de las incrustaciones faciales de cada rostro
codificaciones = face_recognition.face_encodings(rgb_reducido_frame,
rostro_video, model="large")
```

Se hace uso del método que trae consigo face_recognition denominado compare_faces, tal como se explica en el marco teórico, que esta usa como clasificador a modelo K-NN, dada de la lista se calcula el número de votos para cada nombre, es decir los valores que asociados a cada nombre, contar los votos y seleccionar el nombre de la persona con el mayor número de votos. Se determina los índices de donde están esos valores.

```
for codificacion in codificaciones:
# intentar hacer coincidir cada cara de la imagen de entrada con las
# codificaciones conocidas
# Tolerancia del 0.55
coincidencias = face_recognition.compare_faces(datos["codificacion"],
codificacion, tolerance= 0.55)

nombres = "Desconocido"
identificacion =""
parentesco = ""
rostro =""
# comprueba si se ha encontrado una coincidencia
if True in coincidencias:
# encontrar los índices de todas las caras coincidentes y luego
inicializar un
# diccionario para contar el número total de veces que cada cara fue
emparejado
coincidencias_index = [i for (i, b) in enumerate(coincidencias) if b]
conteo_nombres = {}
conteo_identificacion = {}
conteo_parentesco = {}
conteo_rostro = {}

# determinar la cara reconocida con el mayor número
# de votos (nota: en caso de un improbable empate Python
# seleccionará la primera entrada del diccionario)
nombres = max(conteo_nombres, key= conteo_nombres.get)
identificacion = max(conteo_identificacion,
key=conteo_identificacion.get)
parentesco = max(conteo_parentesco, key=conteo_parentesco.get)
rostro = max(conteo_rostro, key=conteo_rostro.get)
```

En el frame se presentará la fecha y hora en tiempo real para la vigilancia, también aparecerá la foto de la persona identificada, tomada desde la base de datos y un cuadro delimitador del rostro que muestra la información de la persona conocida de la que se haya almacenado en la base de datos. En el caso de una persona desconocida se presentará el cuadro en rostro con la etiqueta de desconocido.

```
# bucle sobre las caras reconocidas
for (top, right, bottom, left), nombres, identificacion_ , parentesco_ ,
rostro_ in zip(rostro_video, lista_nombres, lista_identificacion,
lista_parentesco, lista_rostro):
# reescalar las coordenadas de la cara
top *= 4
right *= 4
bottom *= 4
left *= 4
# Si la persona es conocida
if(nombres !="Desconocido"):
cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
                    y0, dy, texto =  80,16, "\nPersona: "+nombres+" "+"
"+"\nIdentificacion: "+identificacion_+"\nParentesco: "+parentesco_
cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 0), 1, cv2.LINE_AA, False)
es_conocido = True
else:
#Cuando la persona es desconocida
cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
cv2.rectangle(frame, (left, bottom + 35), (right+130, bottom), (0, 0, 255),
cv2.FILLED)
font = cv2.FONT_HERSHEY_PLAIN
cv2.putText(frame, nombres, (left + 6, bottom+28), font, 1, (255, 255, 255),
1)
```

2.4.3.6 Vigilancia

En esta sección está integrada en el módulo del reconocimiento facial, se realiza la vigilancia de las personas conocidas y desconocidas, en la que cada persona que aparezca en cámara y al detectar su rostro se captura todo el contexto de la persona y se lo almacena en disco en formato de imagen. A continuación, en la figura 34, se explica el diagrama de flujo de los diferentes procedimientos para la vigilancia.

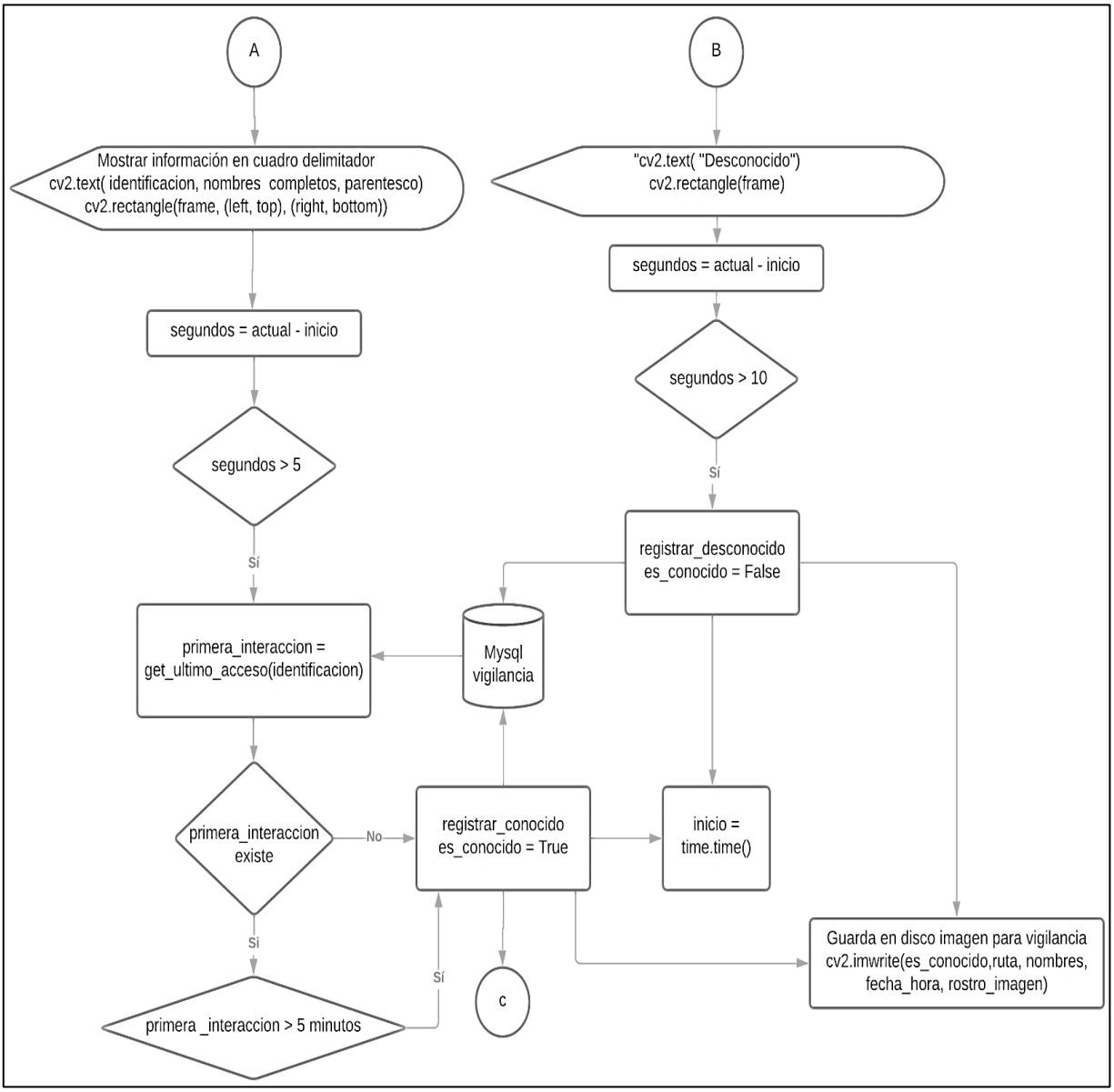


Figura 34. Diagrama de flujo para la vigilancia.

Fuente: Elaboración propia

Durante el reconocimiento, pasado los 5 segundos, se determina si la persona conocida en su última aparición en vigilancia haya transcurrido más de 5 minutos en ese caso se registra como nueva aparición en la base de datos y luego almacena las capturas en el disco.

```
if(segundos > 5):

primera_iteraccion =
control_vigilancia().get_primera_interaccion(identificacion)

    # Si existe una interacción el día de hoy
   if(primera_iteraccion is not None):
 # Si la persona ha vuelta luego de 5 minutos se registra nueva visita
       if datetime.now() - primera_iteraccion > timedelta(minutes=5):

registro_vigilancia_conocido(fecha, es_conocido ,hora, nombres,
captura,fecha_hora,identificacion_)

     else:
# Si no existiera registro de la persona el dia de hoy se registra
como nuevo
registro_vigilancia_conocido(fecha, es_conocido, hora,nombres,
captura,fecha_hora, identificacion_)
inicio = time.time()
```

Ante una aparición de un desconocido se almacenará la información, luego de cada 10 segundos de lo que es detectado el rostro.

```
#Se va guardar persona desconocida cada 10 segundos
segundos = int(actual - inicio)

  if(segundos > 10):

    print("Desconocido")
    registro_vigilancia_desconocido(fecha, hora, nombres, captura,
fecha_hora)
   inicio = time.time()
```

Se captura la fecha y hora, ruta, nombres, rostro de imagen, para la creación de la carpeta a la fecha de la aparición, y luego almacenar con nombre de la persona y la hora.

```
if es_conocido:
        formato =
os.path.sep.join([BASE_DIR+"/historial/vigilancia/conocidos/"+fecha,
"{}.jpg".format(str(hora +" "+nombres))])
    else:
        formato =
os.path.sep.join([BASE_DIR+"/historial/vigilancia/desconocidos/"+fecha
, "{}.jpg".format(str(hora))])
    cv2.imwrite(formato, captura)
```

2.4.3.7 Acceso

En esta sección también está integrada en el módulo del reconocimiento facial, el acceso se procederá a realizar si el usuario administrador, proporciona el acceso ya sea mediante la interfaz de consola o del chatbot.

La biblioteca necesaria para realizar el acceso se muestra en la Tabla 13.

Tabla 13. Librería para el sistema de acceso.

Librería	**Función**
Jetson.GPIO	Manejador de señales digitales en Jetson Nano.

Fuente: Elaboración propia.

A continuación, la figura 35 explica el diagrama de flujo de los diferentes procedimientos para la vigilancia.

Diagrama de flujo del algoritmo de acceso

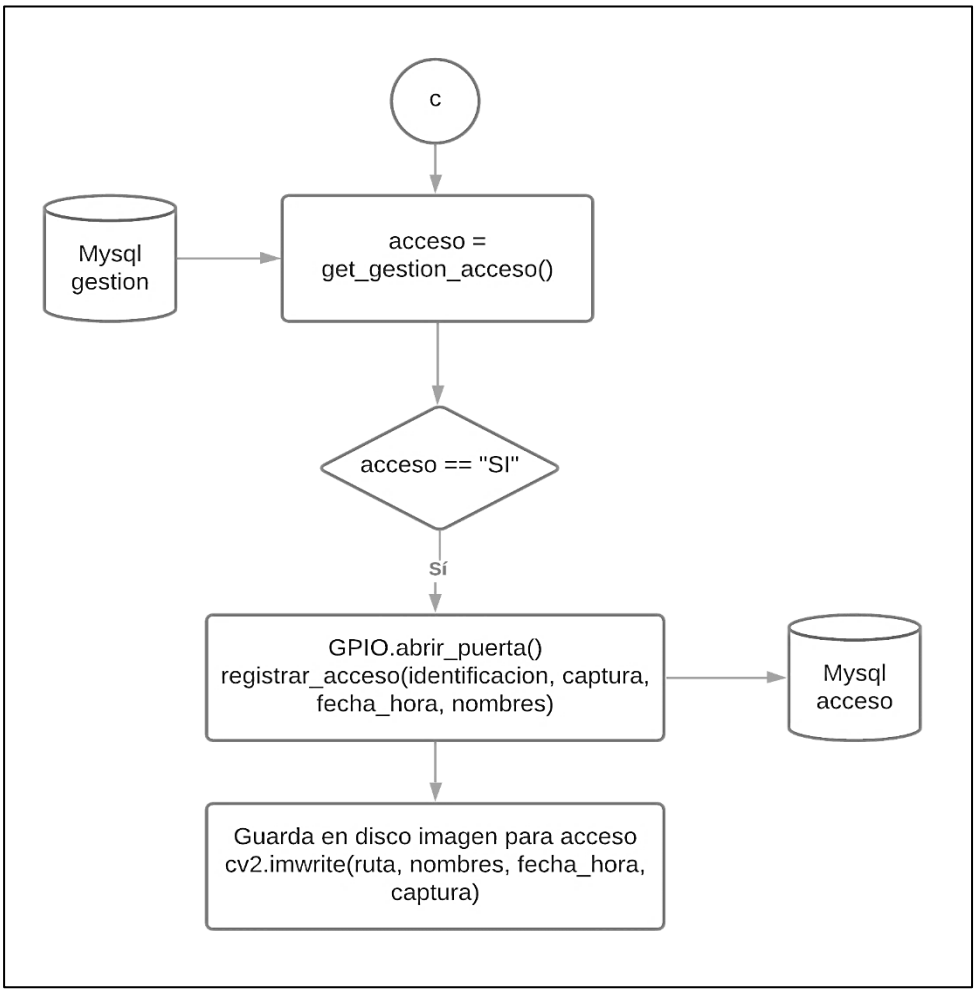


Figura 35. Diagrama de fujo del algoritmo de acceso.

Fuente: Elaboración propia

Durante el reconocimiento para personas conocidas al pasar lo 5 segundos se procede al acceso por la puerta si el usuario administrador lo concede ya sea por la interfaz de consola o por el chatbot, por ello se procede consultar de la tabla gestión si el valor de acceso es igual a SI o NO, en el caso que sea afirmativo se procederá a llamar a la librería GPIO para activar el acceso a puerta por medio de los pines de la Jetson Nano que en su cabecera posee 40 pines, en conjunto con un relay y chapa eléctrica. Luego continúa con el registro en la base de datos en la tabla acceso y serializar la captura del contexto en disco.

```
ACCESO = control_gestion().get_acceso()

if(ACCESO == "SI"):
  print("Acceso permitido")
  gpio.abrir()
  print("Abriendo Puerta")
  time.sleep(3)

  gpio.cerrar()
  print("Cerrando")

registro_acceso(identificacion_,captura, fecha, hora,
nombres,fecha_hora)
print("Acceso Guardado")

formato = os.path.sep.join([BASE_DIR+"/historial/acceso/"+fecha,
"{}.jpg".format(str(hora +" "+nombres))])
    cv2.imwrite(formato, captura)
```

Se instancia la clase del módulo acceso.py, en la que la función GPIO.out permite el control del estado (High/Low), para este caso True o False para el pin 12, en ella se especifica el pin de salida y el estado que se desea que esté.

```
# Para un relay con encendido en bajo voltaje
def abrir(self):
   GPIO.output(self.pin_acceso, False)
def cerrar(self):
    GPIO.output(self.pin_acceso, True)
```

Como se indicaba anteriormente, el relay es de encendido en bajo voltaje, el tiempo que toma el control en bajo es de 3 segundos lo que permite el paso de la corriente a la chapa eléctrica y posteriormente apertura la puerta. Si se realiza el control en alto, el relay interrumpirá la corriente a la chapa eléctrica por lo que la puerta se bloquea hasta una nueva orden de llegada.

2.4.4 Frontend del sistema

El desarrollo de las interfaces se ha empleado Python y OpenCV, a continuación, se presenta los diseños GUI principales del sistema:

Construcción de dataset

La interfaz del dataset tal como se presenta en la figura, presenta el número de capturas realizadas en ese momento, las opciones de teclado: [G] para guardar y [Q] para salir del frame, la información de la persona: su identificación, nombres completos, parentesco, y el cuadro delimitador de la detección del rostro con un etiquetado “Capturando” en la parte superior.

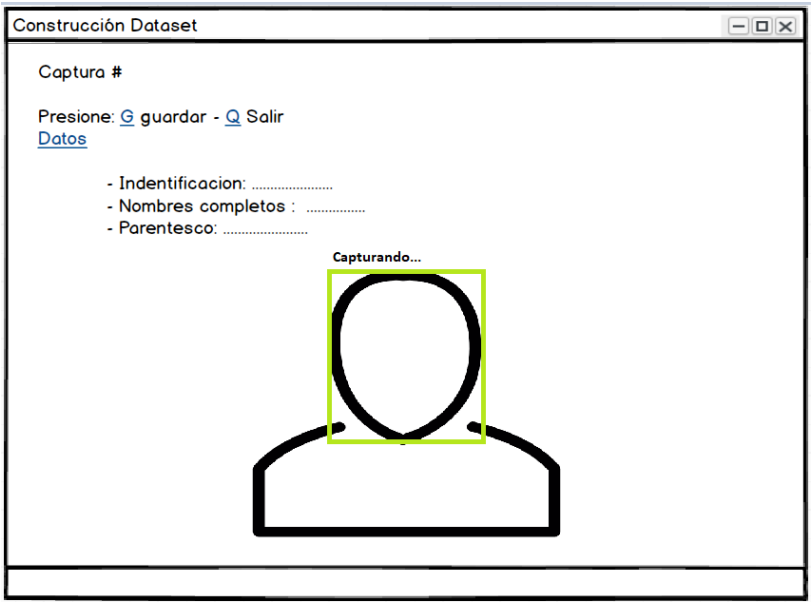


Figura 36. Diseño Interfaz de construcción de dataset.
Fuente: Elaboración propia.

Reconocimiento facial

En una ventana se presenta el proceso de reconocimiento del rostro como se lo muestra en la figura, en esta aparece en la parte superior la fecha y hora en tiempo real para la vigilancia, también aparecerá la foto de la persona identificada, tomada desde la base de datos y un cuadro delimitador del rostro con la información de la persona conocida que se haya registrado en la base de datos, como los nombres completos, la identificación y el parentesco. Se mostrará el mensaje de acceso cuando se lo permita.

En el caso de que la persona sea desconocida se presenta el cuadro delimitador de la detección del rostro con un color rojo y etiquetado “Desconocido” en la parte inferior, como en la figura.

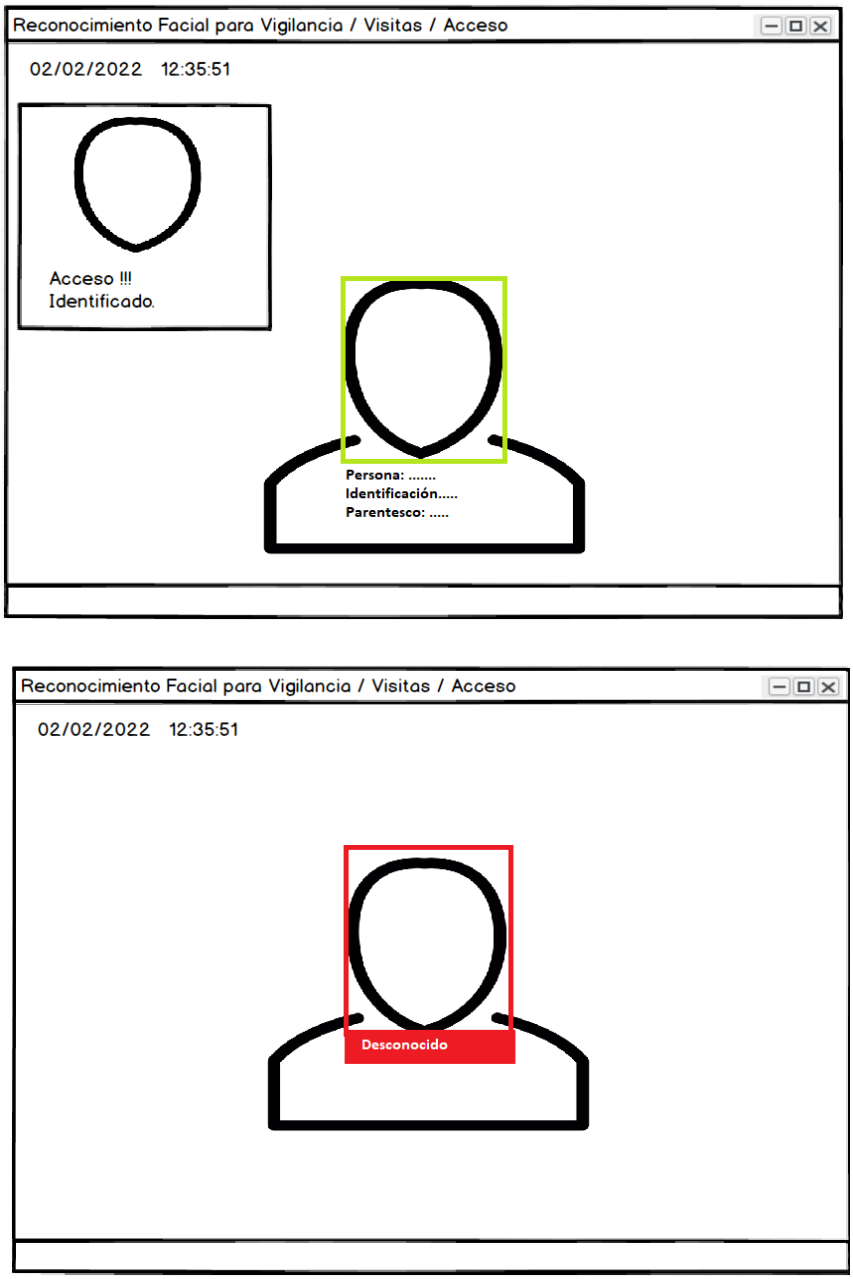


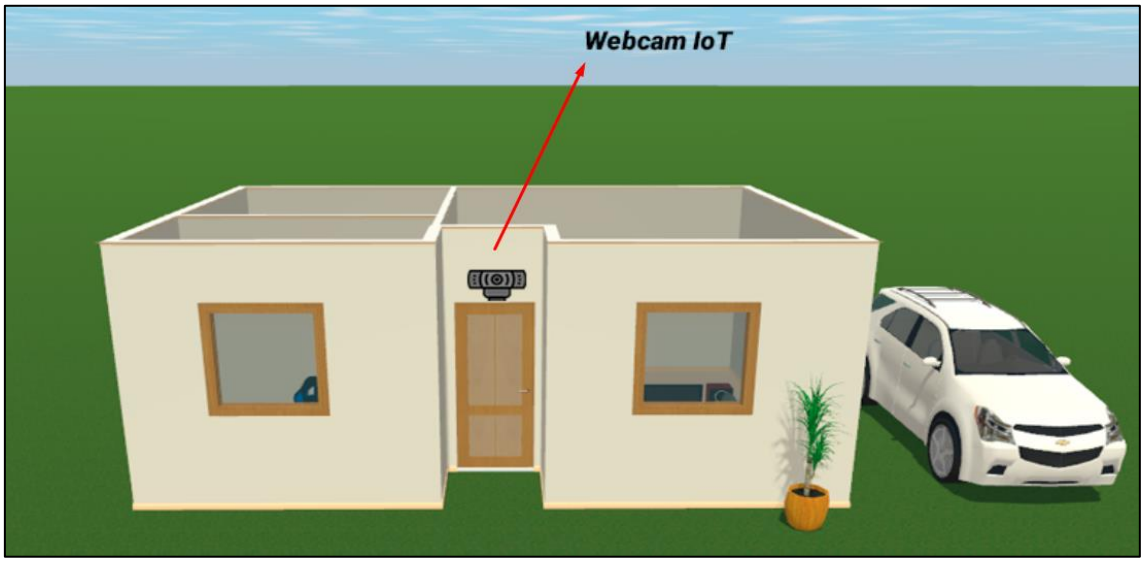
Figura 37. Diseño interfaz reconocimiento facial.

Fuente: Elaboración propia.

2.4.5 Diseño para la configuración domótica del prototipo.

En la figura 38 se presenta la configuración del prototipo para la residencia; partiendo desde la parte externa, sobre la puerta principal se ubica la webcam o cámara de video que servirá para el proceso de la vigilancia y acceso mediante el reconocimiento facial.

Parte externa



Parte interna

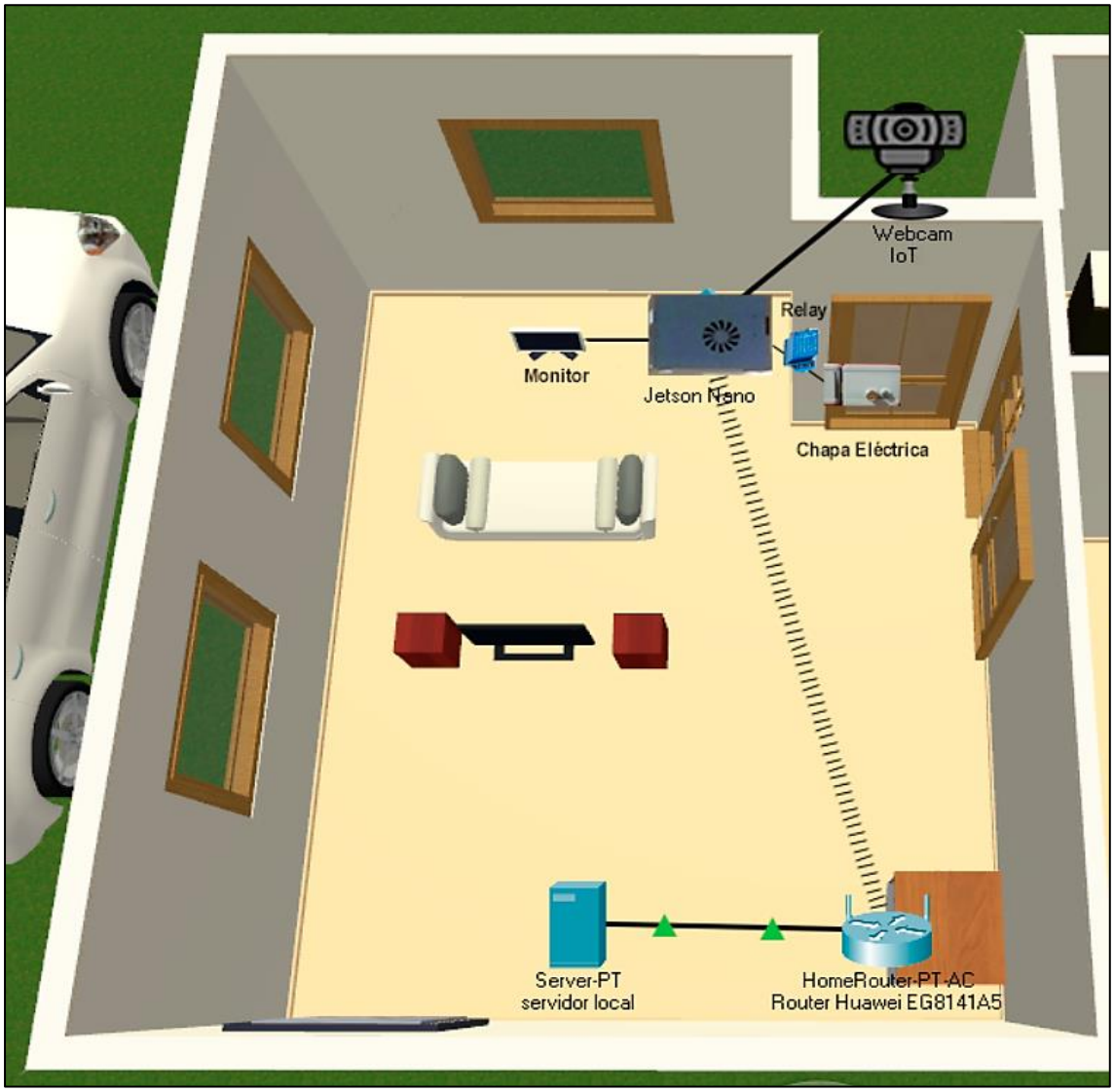


Figura 38. Diseño para la configuración domótica del prototipo.

Fuente: Elaboración propia

En la parte interna, se realiza la conexión de los dispositivos, la chapa eléctrica ubicada en la puerta está conectada al relay y esta a su vez a la jetson nano. La conexión del router al servidor local y a la jetson nano es de manera inalámbrica. Para visualizar el funcionamiento del sistema; se conecta un monitor en la tarjeta o también la conexión remota desde el servidor local (laptop) a la jetson nano para la gestión.

2.4.6 Diseño electrónico del sistema de acceso.

Para realizar la gestión de la chapa eléctrica se emplean 3 pines para la alimentación de circuito que es el pin 17, etiquetada con 3v3, este procesa un voltaje de 3.3 v, otro para señal de control que es el pin 12 con etiqueta D24, y otro que va conectado a tierra que es pin 6.

La Jetson Nano maneja solamente voltajes de 5V y 3V, no permite controlar la chapa eléctrica directamente, ya que se requiere de 12V para que entre en funcionamiento. Por tanto, se usa el relay de 1 canal que va ser como dispositivo intermediario para manipular los voltajes. En la figura 39 se presenta el diseño electrónico del sistema de acceso.

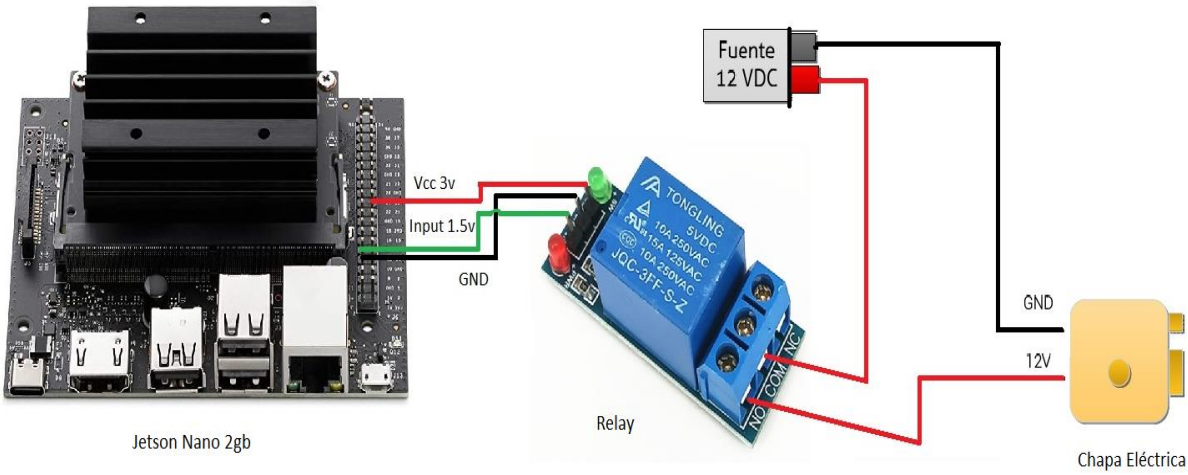


Figura 39. Diseño electrónico del sistema de acceso.
Fuente: Elaboración propia

2.5. Ejecución y/o ensamblaje del prototipo

Primeramente, se procede a la preparación del software del sistema y luego se ensamblará el hardware.

2.5.1 Instalación de Ubuntu LTS 18.04 SDK Kit 4.6

El kit de desarrollo usa una tarjeta microSD como dispositivo de arranque y para el almacenamiento primario. Es fundamental tener una tarjeta que sea lo suficientemente

veloz y de alta capacidad para proyectos; el requisito mínimo para este proyecto es de una tarjeta UHS-1 de 32 GB.

Debido a la limitada capacidad de memoria que posee la Jetson Nano de 2Gb de RAM, se usará espacio de intercambio (memoria SWAP) en la tarjeta microSD. Por esta razón, se recomienda en la página oficial una microSD de 64 GB o mayores y se sugiere usar tarjetas microSD de alta resistencia.

La figura 40 muestra el proceso de formateo de la microSD con la herramienta SD Card Formatter.

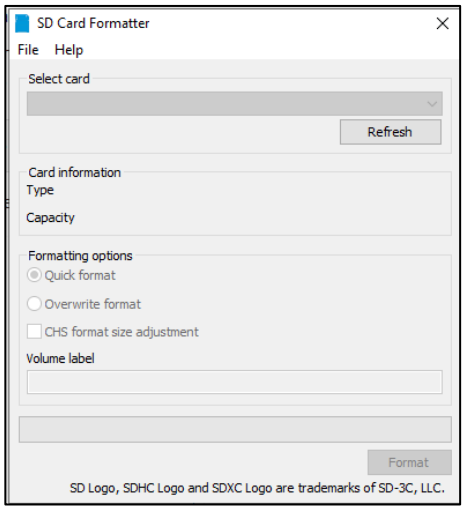


Figura 40. Formateo de memoria SD.
Fuente: Elaboración propia

Luego con el programa balena Etcher se hace el flasheo que tardará unos 10 minutos en escribir y validar la imagen si la tarjeta microSD está conectada a través de USB3.

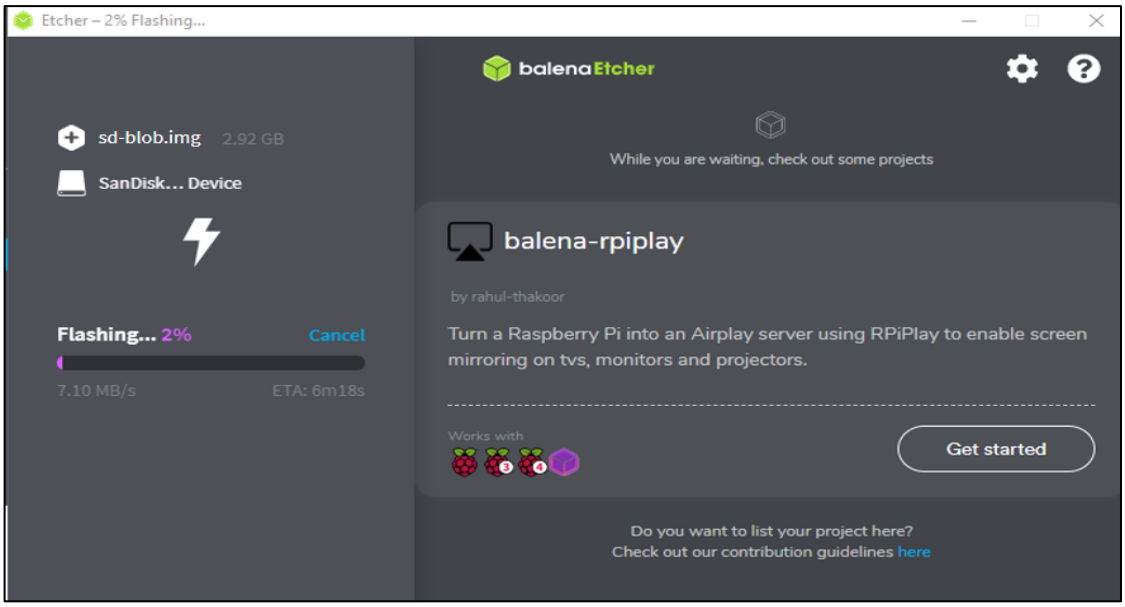


Figura 41. Flasheo de imagen iso en Etcher.
Fuente: Elaboración propia

Luego conectando a la Jetson Nano la microSD, se procederá a la instalación, configurando nombre de usuario, contraseña, y memoria swap. Finalizando se presentará la interfaz gráfica del sistema Ubuntu LTS 18.04 tal como se muestra en la figura 42.

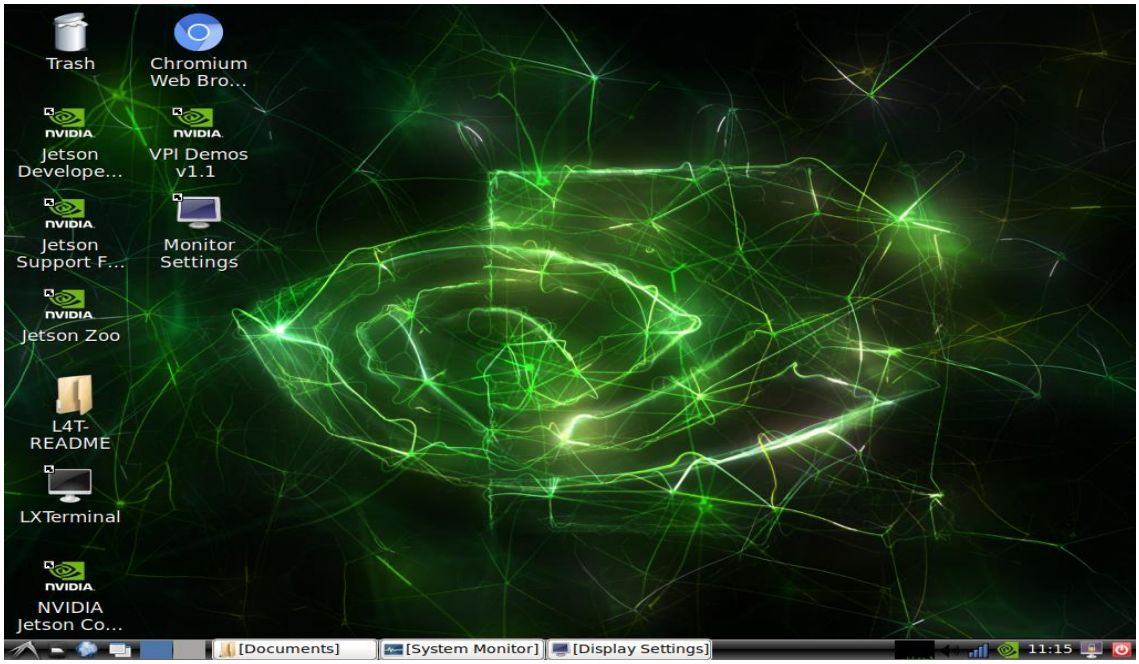


Figura 42. Pantalla inicial de Ubuntu 18.4 en Jetson Nano.

Fuente: Elaboración propia

En la figura 43 se muestra el monitoreo y características de todos los recursos de hardware del dispositivo Jetson Nano 2Gb.

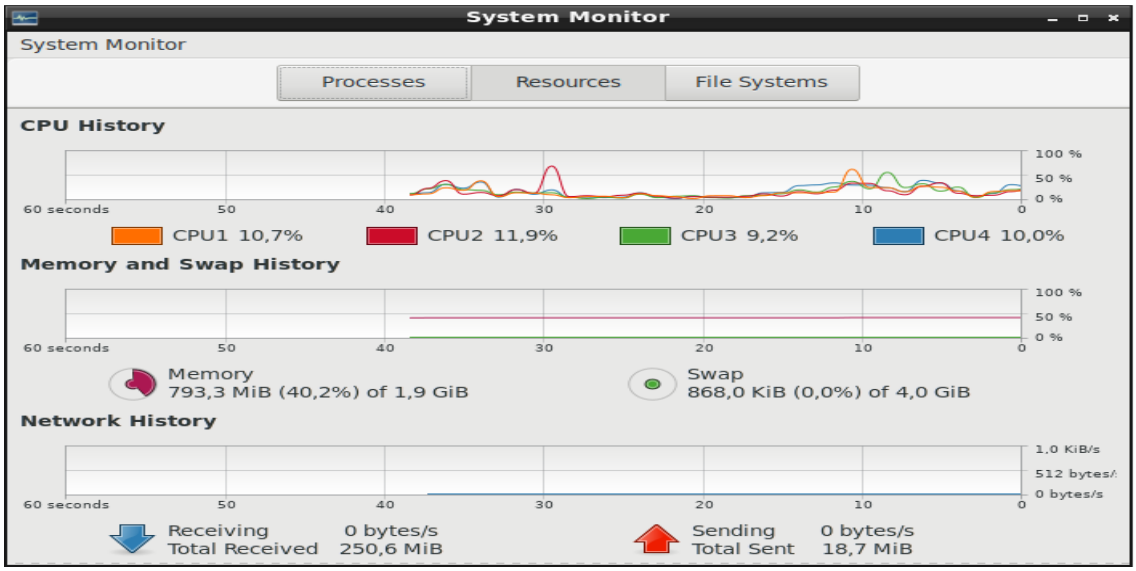


Figura 43. Monitoreo del sistema Jetson Nano 2gb.

Fuente: Elaboración propia

2.5.2 Instalación de librerías

En el proceso de instalación con la imagen iso descargado de la página oficial de Nvidia, trae consigo un SDK Jetson Linux Driver Package (L4T) con el sistema operativo Linux y en esta ya viene preinstalado librerías para realizar cualquier proyecto de inteligencia artificial y también API aceleradas CUDA-X para aprendizaje profundo (Deep Learning), visión por computador, computación acelerada y multimedia, que también incluye ejemplos, documentación y un set de herramientas para el desarrollo.

Para este proyecto se van a instalar librerías adicionales que no vienen por defecto como face_recognition y dlib ya que se necesitan para que funcione la detección y el reconocimiento del rostro.

Como buena práctica para el desarrollo de proyectos de inteligencia artificial en este dispositivo, todo se instala en un entorno virtual de Python en su versión 3.6.9 que viene por defecto de instalación, para este ambiente de desarrollo se lo denomina como (env-domotica).

En la tabla 14 se presentan las principales librerías a usar en el dispositivo Jetson nano, para crear el sistema de reconocimiento facial y acceso, estas se instalan con el comando “pip install”, estas traen otros paquetes que se implementan luego automáticamente. La explicación paso a paso de la configuración de un entorno virtual en Python e instalación de paquetes para este documento tornaría a ser extenso, por lo que no se describe ya que el tema de este proyecto no va enfocado a ello.

Tabla 14. Librerías para el sistema de reconocimiento facial.

Librería	Versión
numpy	1.19.0
opencv	4.5.4
Face_recognition	1.3.0
dlib	19.22.1
pillow	8.4.0
pymysql	1.0.2
Jetson.GPIO	2.0.17

Fuente: Elaboración propia

La figura 44 se muestra todas las librerías instaladas para el funcionamiento del sistema de reconocimiento facial ejecutando el comando “pip3 list”.

```
jerreyes2@jetson-nano: ~
File  Edit  Tabs  Help
jerreyes2@jetson-nano:~$ workon env-domotica
(env-domotica) jerreyes2@jetson-nano:~$ pip3 list
Package                Version
---------------------- -------
click                  8.0.3
cycler                 0.11.0
Cython                 0.29.25
dlib                   19.22.1
face-recognition       1.3.0
face-recognition-models 0.3.0
importlib-metadata     4.8.3
imutils                0.5.4
Jetson.GPIO            2.0.17
joblib                 1.1.0
kiwisolver             1.3.1
matplotlib             3.3.4
numpy                  1.19.0
Pillow                 8.4.0
pip                    21.3.1
PyMySQL                1.0.2
pyparsing              3.0.6
python-dateutil        2.8.2
scikit-learn           0.24.2
scipy                  1.5.4
setuptools             58.3.0
six                    1.16.0
threadpoolctl          3.0.0
typing_extensions      4.0.1
wheel                  0.37.0
zipp                   3.6.0
(env-domotica) jerreyes2@jetson-nano:~$ python
Python 3.6.9 (default, Dec  8 2021, 21:08:43)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.5.4
>>>
```

Figura 44. Librerías instaladas en Jetson Nano

Fuente: Elaboración propia

2.5.3 Conexiones de Jetson Nano en CASE

Se realiza la instalación de los botones de encendido y reset, ventilador a 5v para disipar el calor, adaptador de pines en la cabecera y el adaptador de memoria SD. En la figura 45 muestra el proceso de instalación de estos dispositivos.

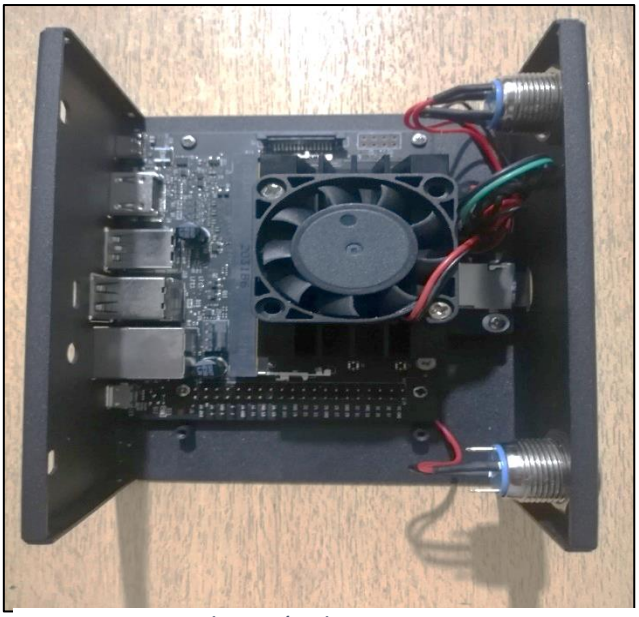


Figura 45. Colocación de Jetson Nano en CASE.

Fuente: Elaboración propia

La figura 46 se muestra la vista frontal del dispositivo en la que se encuentran el botón de encendido, reset y la ranura para inserción de memoria SD.

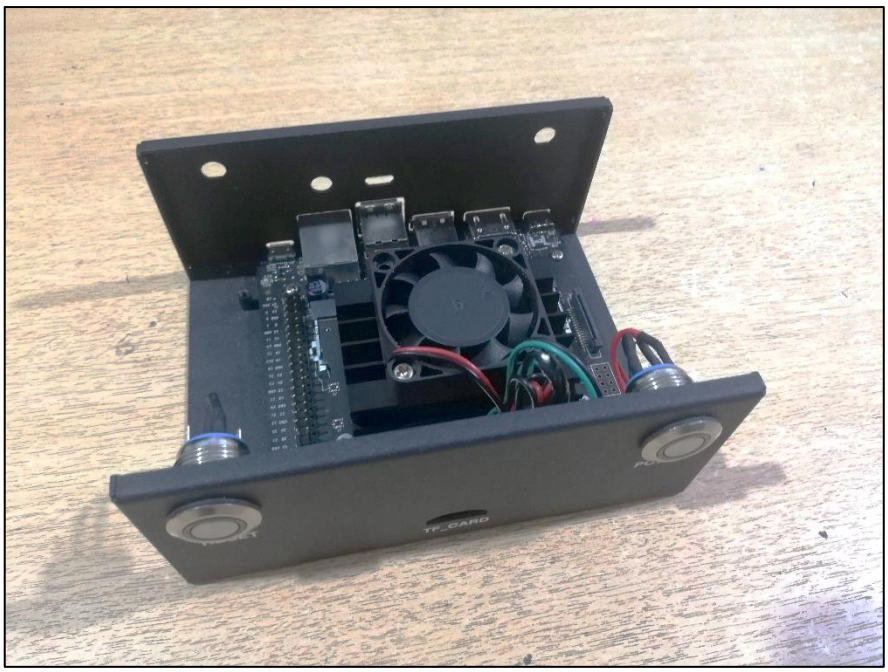


Figura 46. Vista frontal de Jetson Nano en CASE.

Fuente: Elaboración propia

La figura 47 se muestra 7 entradas en las que 3 son puertos USB, una para alimentación a energía para cargador con interfaz conector tipo c, una para entrada HDMI en la que va conectado el monitor, una interfaz ethernet par la conexión a internet y salida a la red.

Figura 47. Vista de interfaces de entrada y salida de Jetson Nano en CASE

Fuente: Elaboración propia

Antes de cerrar la cubierta, se realiza una prueba de encendido, funcionamiento de ventilador y pines para el controlador relay, como se muestra en la figura 48.

Figura 48. Prueba de encendido en Jetson Nano.

Fuente: Elaboración propia

Se continúa con la ubicación de cubierta del case y adaptador exterior para los 40 pines de la Jetson Nano figura 49, en ella se situarán los cables que van a ir conectados al relay.

Figura 49. Ubicación de adaptador exterior para encabezado de 10 pines en Jetson Nano.

Fuente: Elaboración propia

2.5.4 Conexión del sistema de acceso.

Una vez armada las conexiones de la Jetson Nano se procede a la conexión del controlador de encendido para la chapa eléctrica, se emplea 3 pines: para la alimentación de circuito que es el pin 17 va conectada al pin con etiquetado Vcc que es el de suministro de energía del relay. El pin 6 de la jetson va conectado al segundo pin etiquetado con GND conexión a tierra. El pin de control que es el 6 va conectado al tercer pin del relay con etiquetado IN. En la figura 50 se muestra el proceso. Se utiliza una fuente de alimentación para corriente continua de 12v, en la que el polo positivo va en el conector con etiqueta COM del relay. Se cierra el circuito ubicando el cable a tierra directamente en la fuente.

Figura 50.Conexión de pines relay y jetson nano.

Fuente: Elaboración propia

La chapa eléctrica está diseñada para ser ubicada en una puerta, la conexión del cable positivo va en la salida de conector relay “ON” y el cable negativo va conectado a tierra GND. En la figura 51 se muestra la instalación de la chapa eléctrica en la puerta de la residencia.

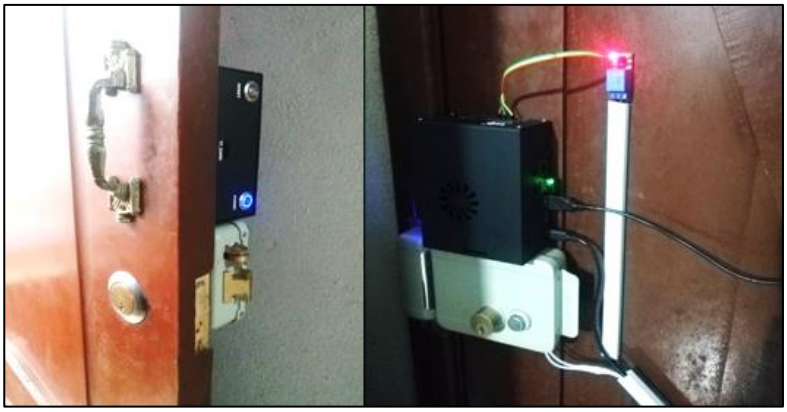


Figura 51. Instalación de chapa eléctrica.

Fuente: Elaboración propia

A continuación, se realiza la conexión con un módulo Wifi usb para minimizar los cables de conexión entre la jetson nano y el router físico, se puede también usar un cable ethernet para una menor latencia en la transmisión de paquetes. Para interactuar directamente a la jetson nano, se usan dispositivos de entrada y salida: monitor con el cable HDMI, un teclado y mouse, también mediante conexión remota. En los puertos USB se conecta la cámara y el módulo wifi.

Figura 52. Conexión de jetson nano a router y dispositivos de E/S.

Fuente: Elaboración propia**.**

2.5.4 Resultado final del ensamblado del prototipo.

La figura presenta el ensamblado del prototipo para el proceso de construcción del dataset, en la que se encuentran conectados los dispositivos de cámara para las capturas, router, monitor, jetson nano, servidor local (laptop) para el gestor de base de datos interconectados en la red de área local de la residencia.

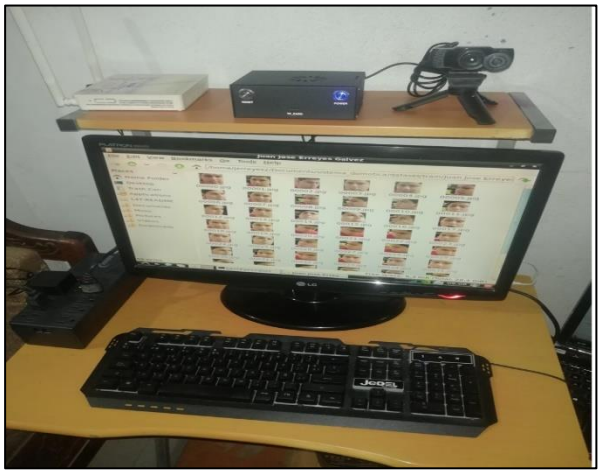


Figura 53. Ensamblado de prototipo para la construcción de dataset.

Fuente: Elaboración propia**.**

La figura 54 se muestra el ensamblado final del prototipo para el sistema de vigilancia y acceso a una residencia, en la que se encuentran conectados los dispositivos de cámara, router, chapa eléctrica en una puerta real, jetson nano con servidor local (laptop) para base de datos y conexión remota, fuente de alimentación de electricidad.

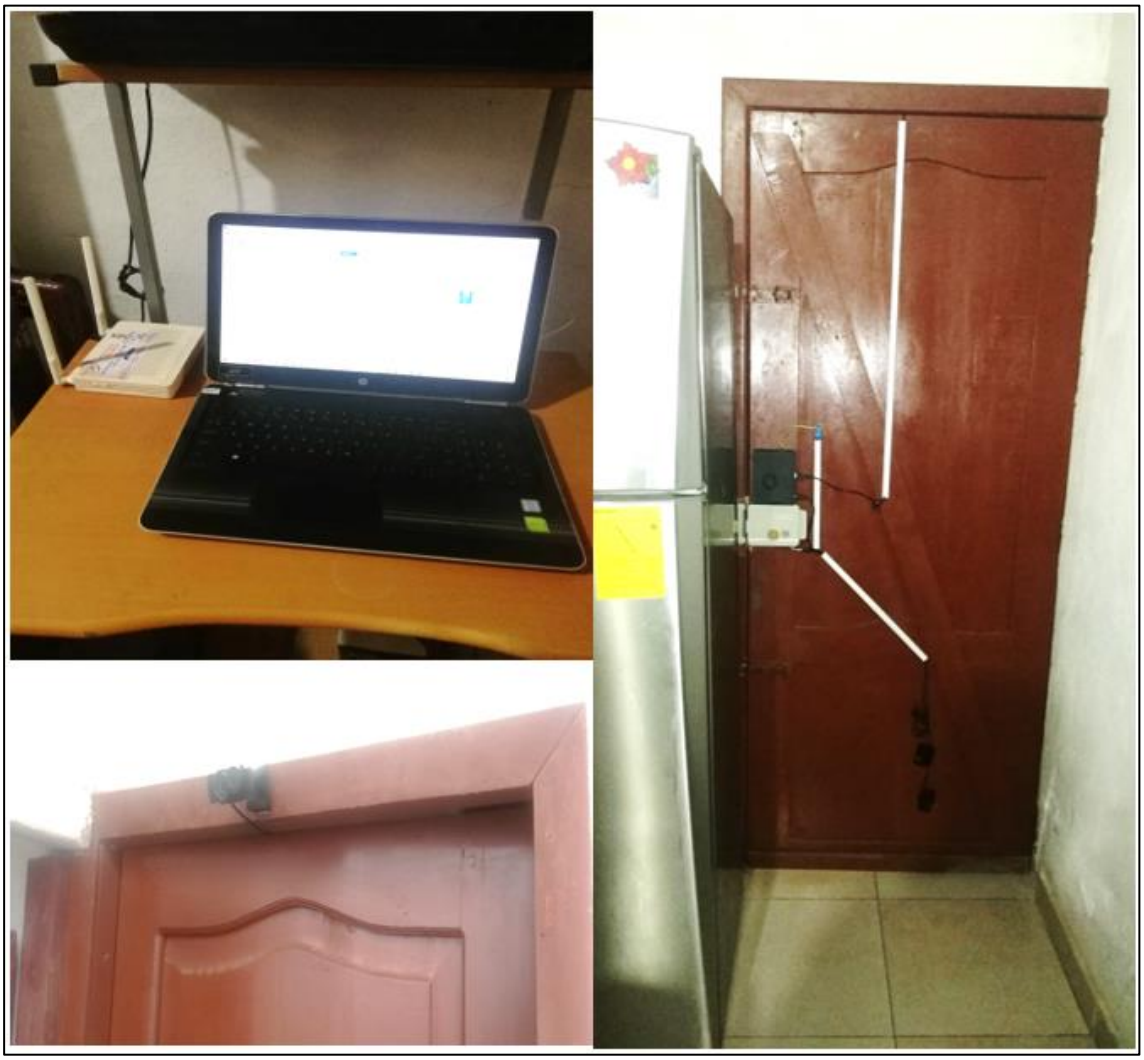


Figura 54. Ensamblado final de prototipo.

Fuente: Elaboración propia.

2.5.5 Ejecución del sistema

Se inicia el programa en el entorno virtual de Python denominado “env-domotica” tal como se muestra en la figura 55, en la que están instalados todos los paquetes que permiten el reconocimiento facial, y el acceso por medio de los pines. En la ruta /Documents/sistema_domotica/ se encuentra el archivo main.py y se procede a ejecutarlo.

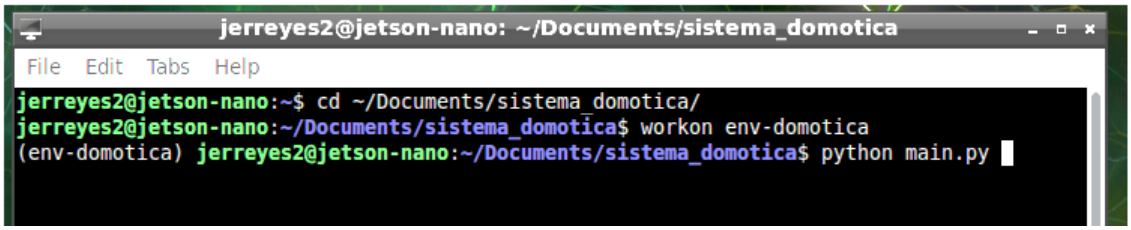


Figura 55. Inicio del sistema
Fuente: Elaboración propia

En el menú principal figura 56 el usuario administrador tendrá 3 opciones principales para acceder, en la que la primera podrá realizar el registro de personas conocidas del hogar, en la segunda iniciar el sistema de video vigilancia y acceso, como tercera poder salir del sistema.

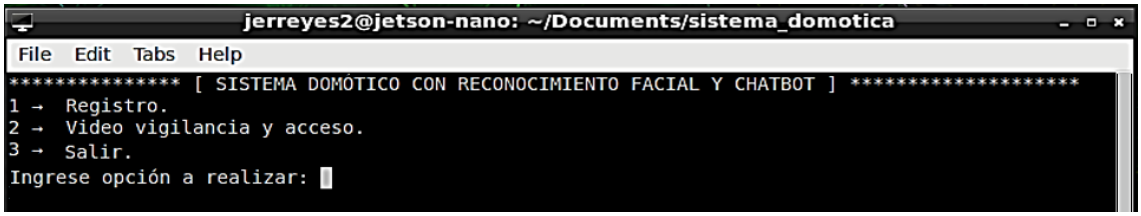


Figura 56. Menú principal del sistema
Fuente: Elaboración propia

Al seleccionar el registro se presentará el siguiente submenú figura 57 con otras opciones para el llenado de la información, captura de rostros, y el proceso de entrenamiento.

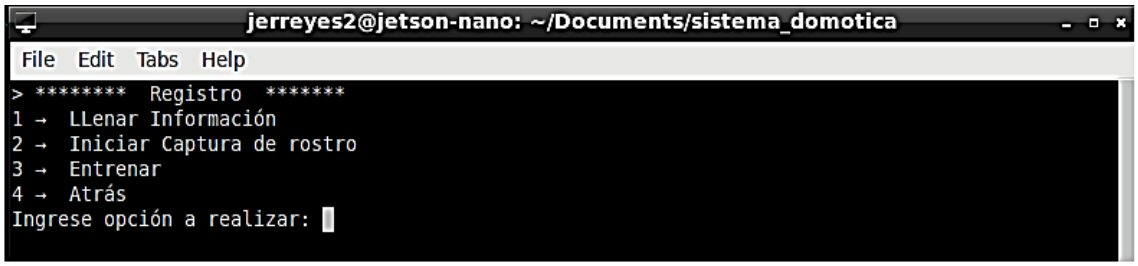


Figura 57. Submenú para registro y entrenamiento.
Fuente: Elaboración propia

Seleccionado la opción de llenado de información se presentará otro submenú como se visualiza en la figura 58, en la que el usuario administrador podrá llenar los campos necesarios para el registro de la persona conocida de la residencia, en este caso se pedirá el número de identificación, nombres y apellidos de la persona y el parentesco que tenga con los propietarios de la residencia.

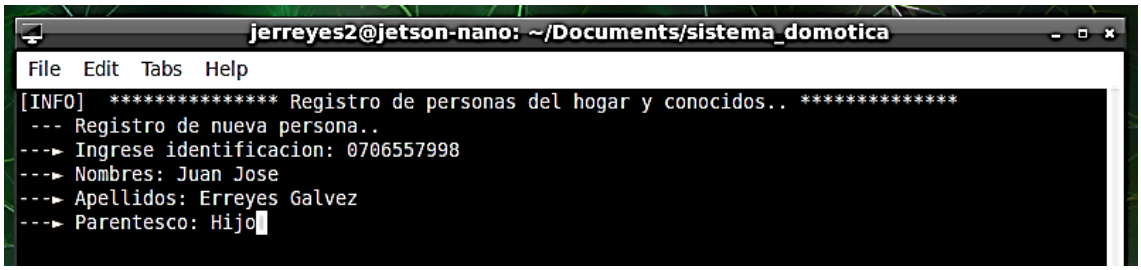


Figura 58. Registro de persona

Fuente: Elaboración propia

Finalizado el registro, se presenta nuevamente el submenú anterior, en la cual puede seleccionar la opción de captura de rostro, en esta se presentará otro submenú como en la figura 59 en la que se pide la identificación de la persona registrada anteriormente o de una existente que desee actualizar.

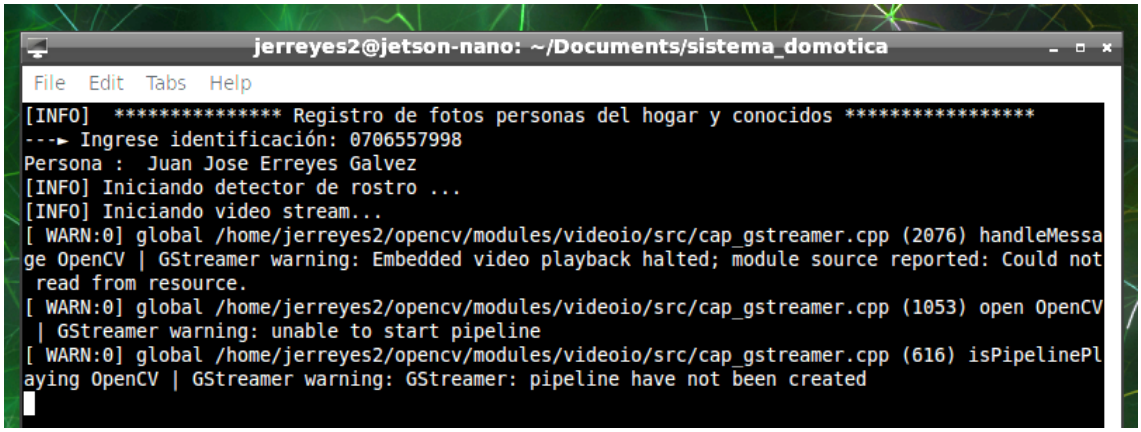


Figura 59. Captura de rostros en persona

Fuente: Elaboración propia

Identificada la persona se inicia el frame para la construcción de dataset tal como se presenta en la figura 60, en la cual se presentará el número de capturas realizadas en ese momento, las opciones de teclado: [G] para guardar y [Q] para salir del frame, la información de la persona: su identificación, nombres completos, parentesco, y el cuadro delimitador de la detección del rostro con un etiquetado “Capturando” en la parte superior.

Figura 60. Frame para la construcción del dataset.

Fuente: Elaboración propia

La figura 61 se presentan las capturas realizadas hacia una persona, con número correspondiente a cada una de ellas, cada persona va ser almacenada como subdirectorios de la carpeta “dataset/train” etiquetada con los nombres completos.

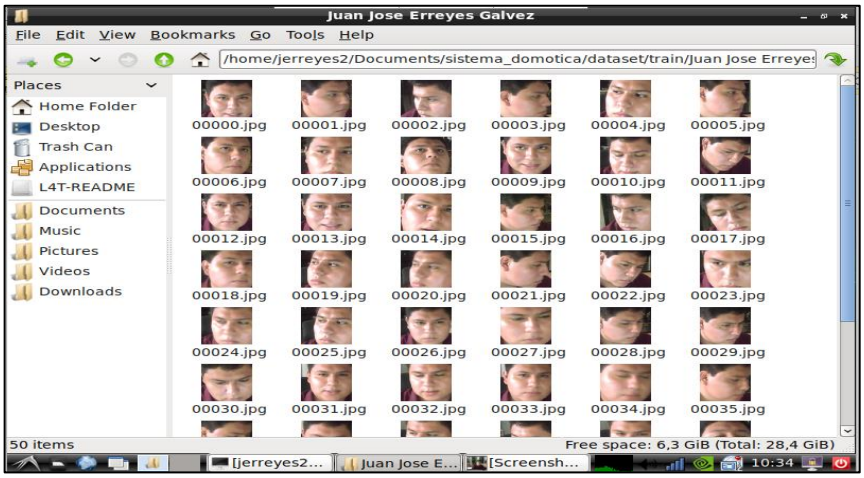


Figura 61. Capturas realizadas en una persona.

Fuente: Elaboración propia

Al salir, las capturas se guardarán y se presentará nuevamente el submenú. Para el proceso de entrenamiento se selecciona la opción 3, en esta se va ir presentando el procesamiento de cada imagen del dataset, en la figura 62 se presenta el funcionamiento.

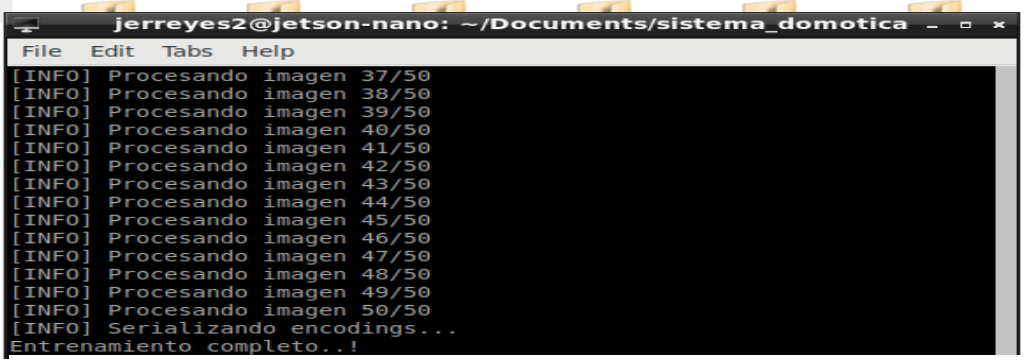


Figura 62. Proceso de construcción de codificaciones del rostro.

Fuente: Elaboración propia

Finalizado el entrenamiento, para volver al menú principal se selecciona la opción 4. Al iniciar el sistema de videovigilancia y acceso se muestra un submenú como se muestra en la figura 63 en la se permite o no el acceso a la residencia.

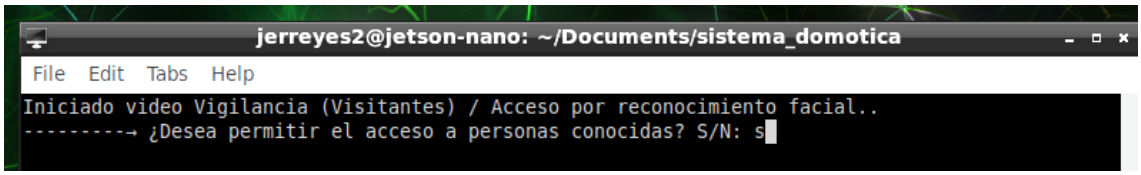


Figura 63. Proceso de gestión de acceso.
Fuente: Elaboración propia

Luego inicia el frame para el proceso de reconocimiento facial como se lo muestra en la figura 64, en esta aparece en la parte superior la fecha y hora en tiempo real para la vigilancia, también aparecerá la foto de la persona identificada, tomada desde la base de datos y un cuadro delimitador del rostro con la información de la persona conocida que se haya registrado en la base de datos. Se procederá al acceso si el usuario lo haya otorgado y este se almacenará en la carpeta historial/acceso caso contrario se lo almacenará también en la carpeta historial/vigilancia/conocido Posteriormente se guardará en la base de datos del sistema.

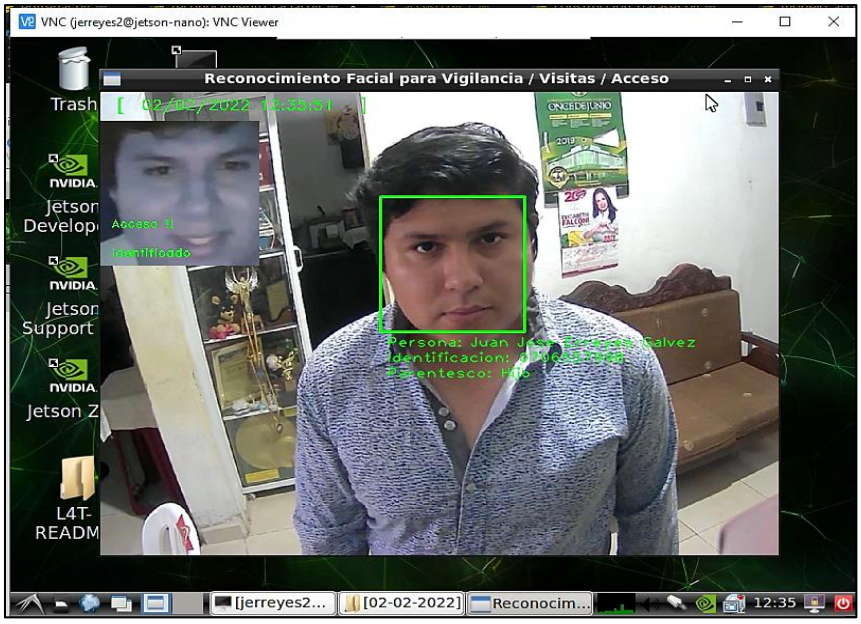


Figura 64. Frame para video vigilancia y acceso.
Fuente: Elaboración propia

En el caso de que la persona sea desconocida, se presentará un cuadro delimitador del rostro de color rojo con el etiquetado “Desconocido” como se lo muestra en la figura 65 y se guardará la imagen del contexto desconocido en la carpeta

historial/vigilancia/desconocido y posteriormente se lo almacena como registro en la base de datos del sistema.

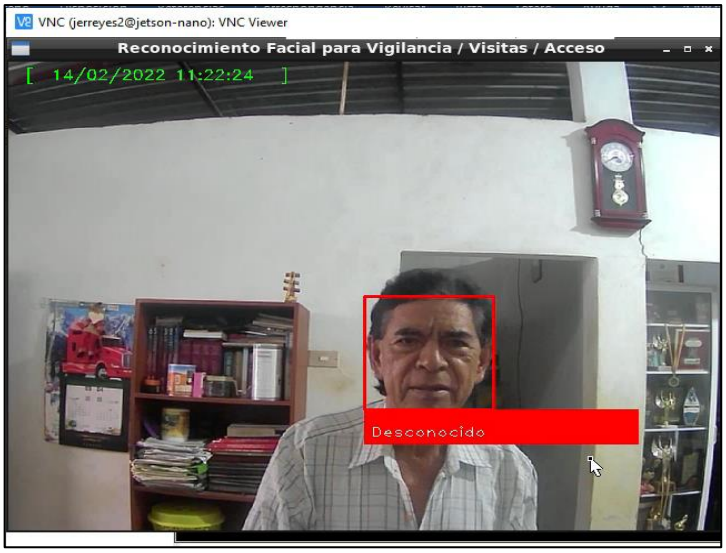


Figura 65. Detección de persona desconocida.

Fuente: Elaboración propia

En la carpeta historial se presentan 2 subcarpetas, en la que el usuario administrador puede visualizar en la parte de vigilancia las imágenes de las personas conocidas y desconocidas. En el de acceso las personas que han ingresado desde la puerta de la residencia. En la figura 66 se muestran los 2 directorios.

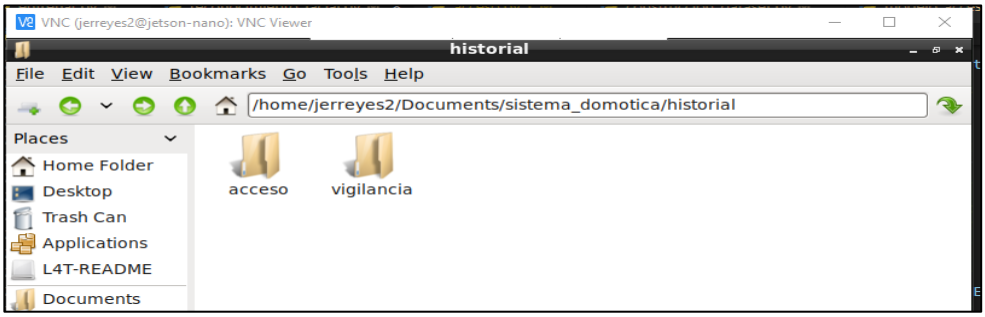


Figura 66. Carpeta historial.

Fuente: Elaboración propia

En la figura 67 se muestran los subdirectorios de vigilancia.

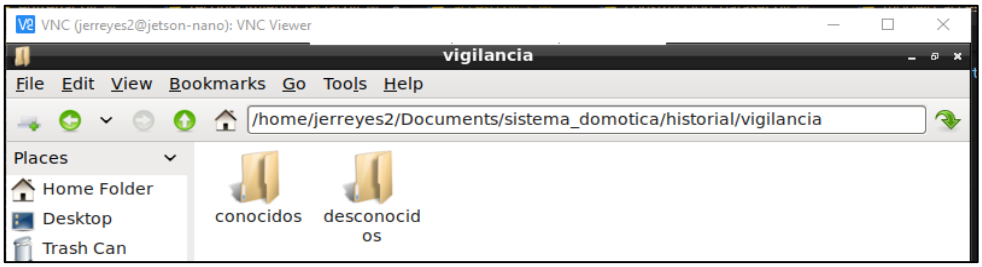


Figura 67. Subdirectorios de vigilancia.

Fuente: Elaboración propia

Al entrar al subdirectorio de conocidos como se visualiza en la figura 68, se presentan carpetas con la fecha correspondiente del día, y dentro de ellas se muestran imágenes de las personas que han aparecido en la puerta de la residencia.

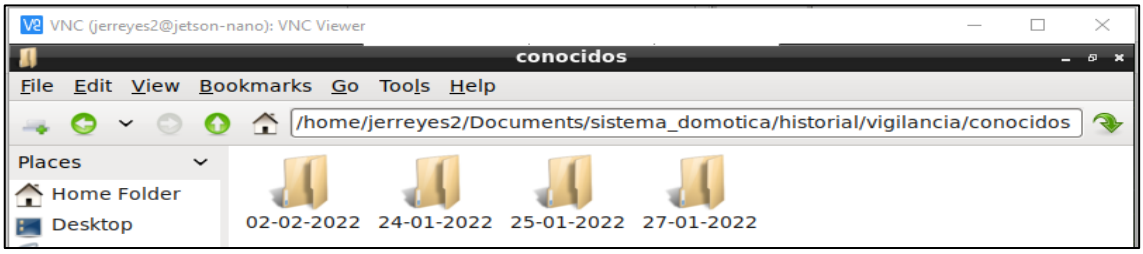


Figura 68. Subdirectorios de desconocidos.

Fuente: Elaboración propia

En cada una de ellas se etiqueta con la hora/minutos/segundos y los nombres completos.

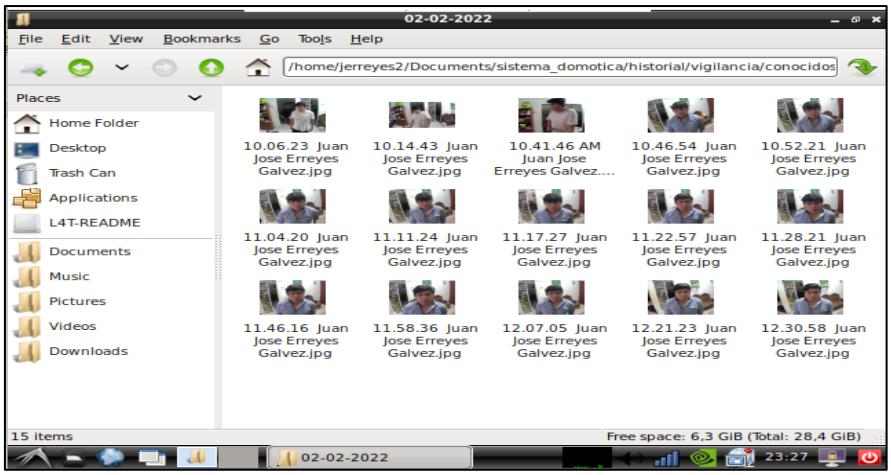


Figura 69. Imágenes de conocidos con etiquetado nombres - hora/minutos/segundos

Fuente: Elaboración propia

Al dar clic en una de ellas se ve grabada la fecha y hora en la parte superior, en la figura 70 se muestra un ejemplo de vigilancia de persona.

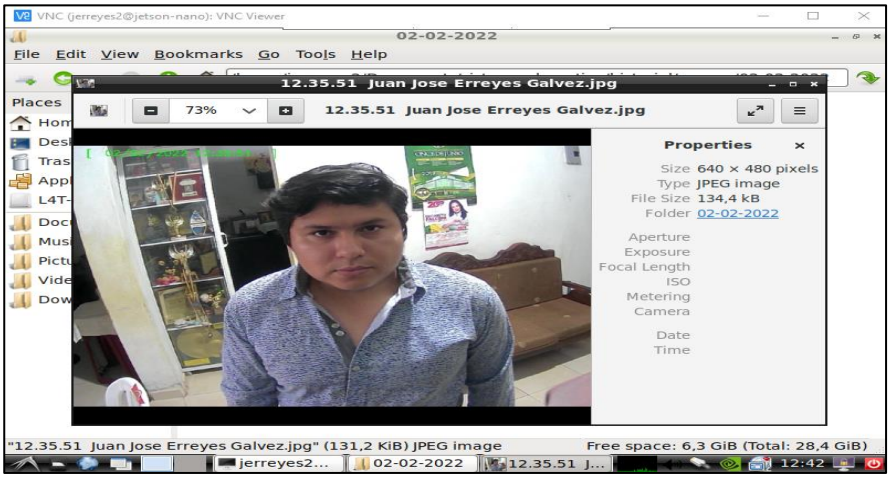


Figura 70. Captura de vigilancia a una persona.

Fuente: Elaboración propia

De igual forma para desconocidos el proceso es similar, pero en el etiquetado de cada imagen se graba solamente la hora/minutos/segundos de la persona que haya visitado a la residencia, en la figura 71 se presenta la hora de llegada de desconocidos.

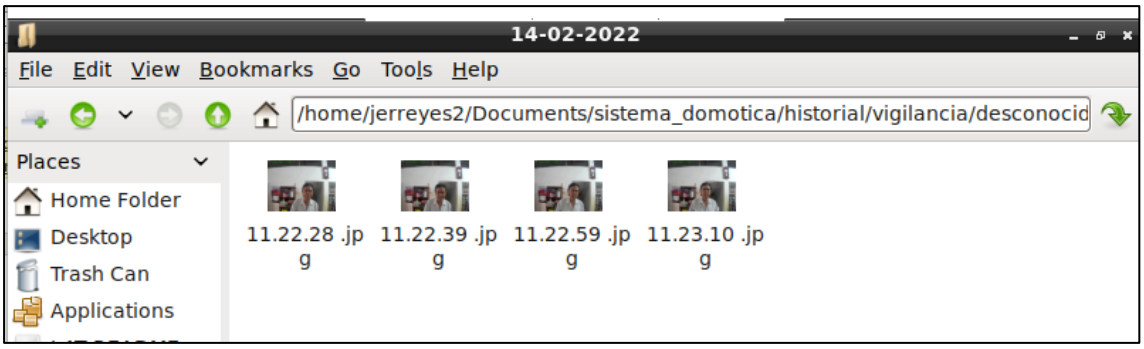


Figura 71. Imágenes de desconocidos con etiquetado hora/minutos/segundos

Fuente: Elaboración propia

En la parte de acceso, se indica el directorio con etiquetado de fecha correspondiente del día como se visualiza y dentro de ellas se muestran imágenes de las personas que han accedido desde la puerta de la residencia.

Similar al proceso anterior en vigilancia, en cada una de ellas se etiqueta con la hora/minutos/segundos del acceso, en la figura 72 se presenta un ejemplo

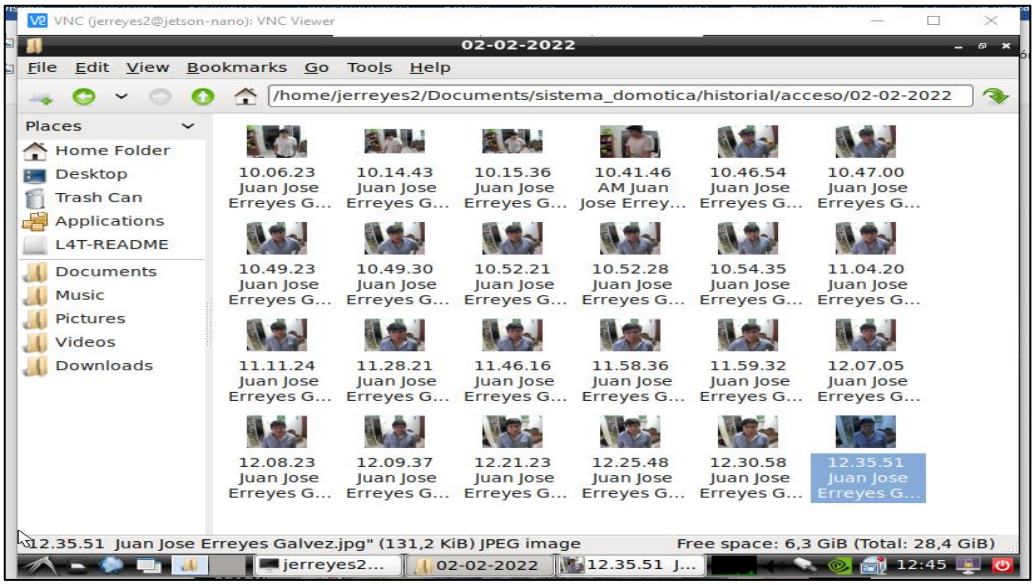


Figura 72. Imágenes de acceso con etiquetado nombres - hora/minutos/segundos

Fuente: Elaboración propia

En la base de datos MySQL están almacenados los datos de registro, acceso y vigilancia. En la figura 73 se muestra las consultas a cada una de las tablas.

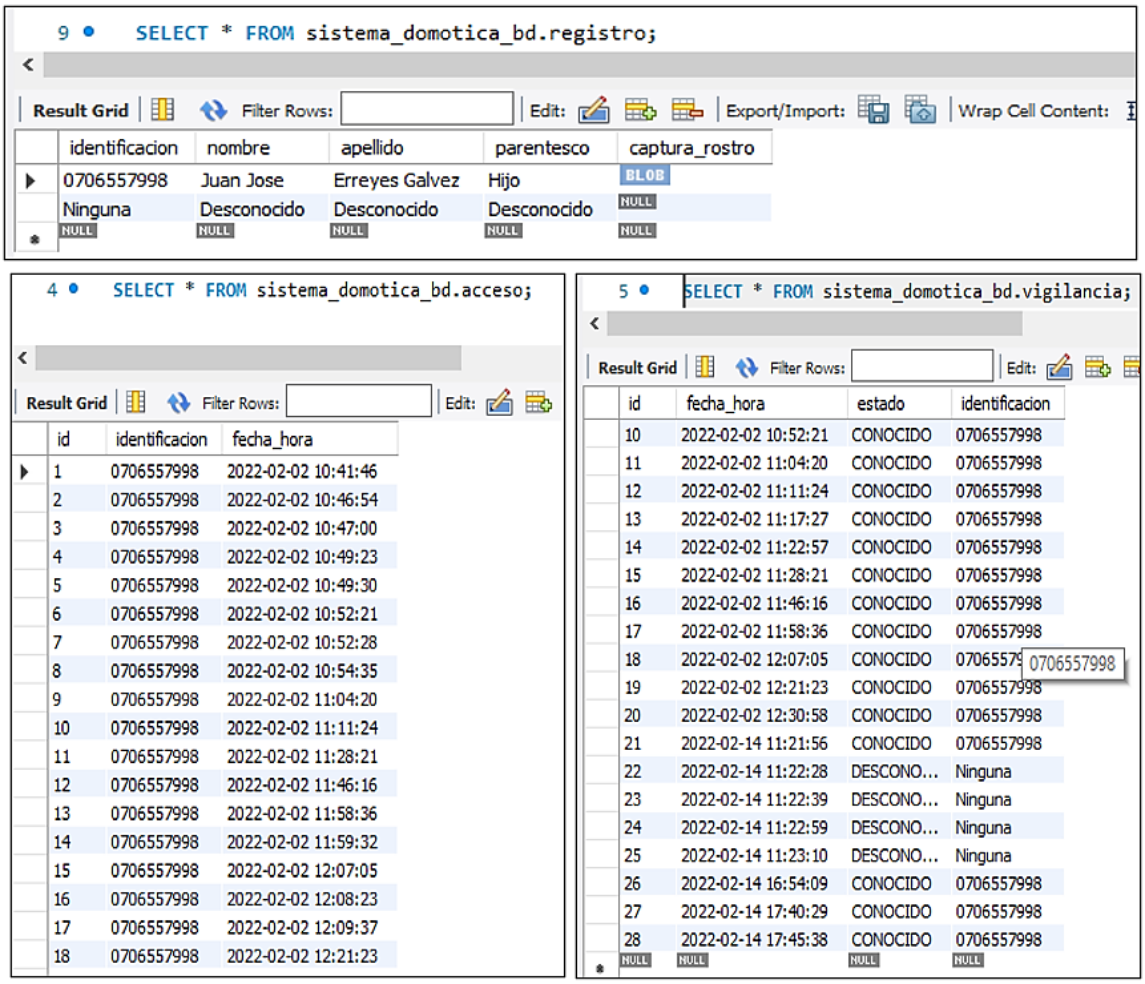


Figura 73. Registros en tablas.

Fuente: Elaboración propia

Chatbot

Desde el navegador del sistema se inicia el aplicativo web del chatbot. Se empieza una conversación de manera interactiva con lenguaje natural. En la figura 74 se muestra la respuesta que da al decirle un “Hola” y este responderá.

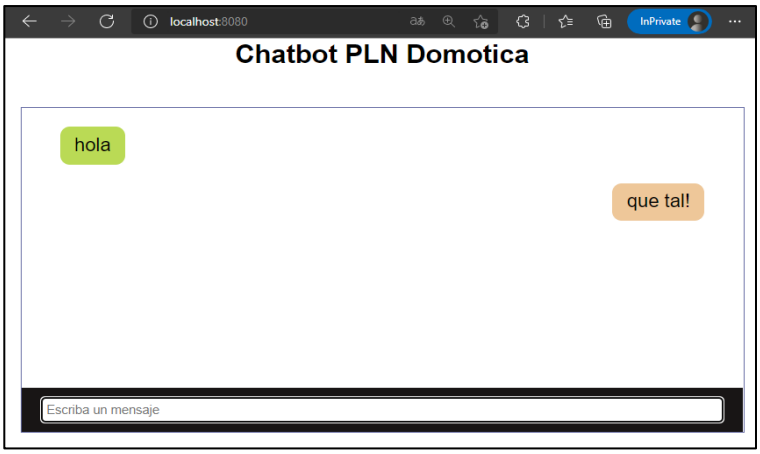


Figura 74. Conversación inicial de chatbot.

Fuente: Elaboración propia

En él contexto de este proyecto el chatbot está entrenado para que se le pueda preguntar o dar órdenes. A continuación de presentan los escenarios:

Para que el usuario pueda otorgar acceso a las personas conocidas se le comunica al chatbot que permita acceso, como respuesta dará la afirmación de que se realizó la orden como se presenta en la figura 75.

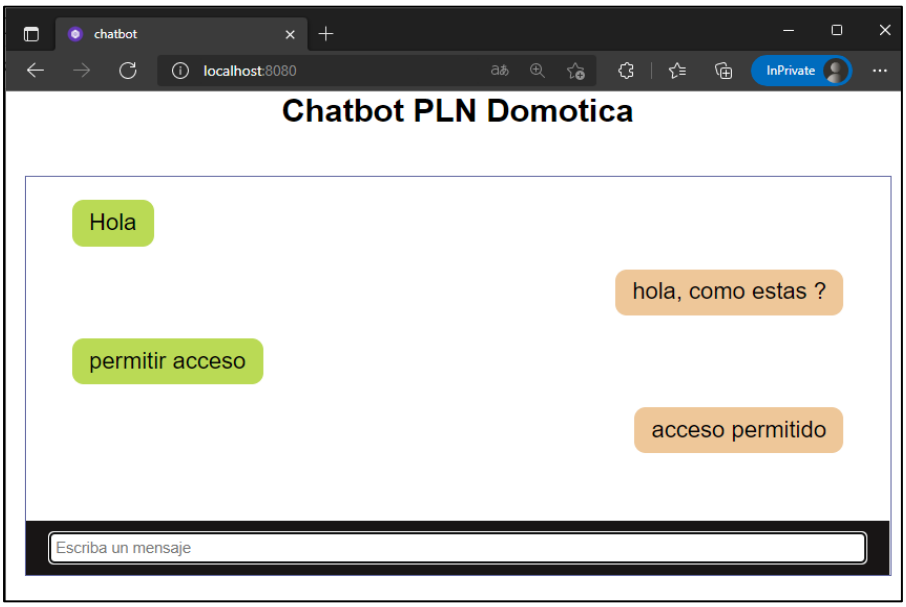


Figura 75. Gestión para otorgar acceso en chatbot.

Fuente: Elaboración propia

Caso contrario si se desea no permitir el acceso.

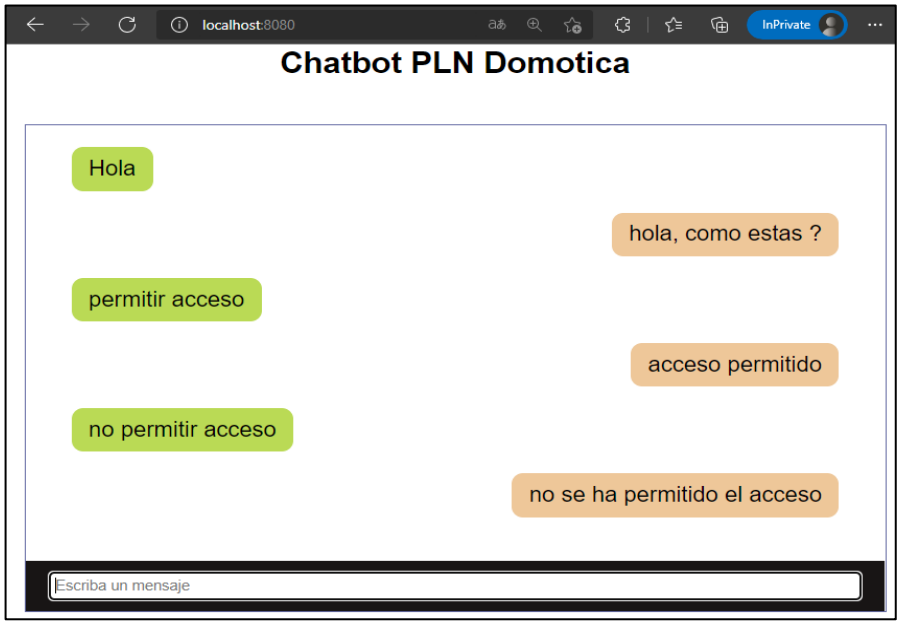


Figura 76. Gestión para no otorgar acceso en chatbot.

Fuente: Elaboración propia

En la tabla de gestión se reconoce la admistración de acceso.

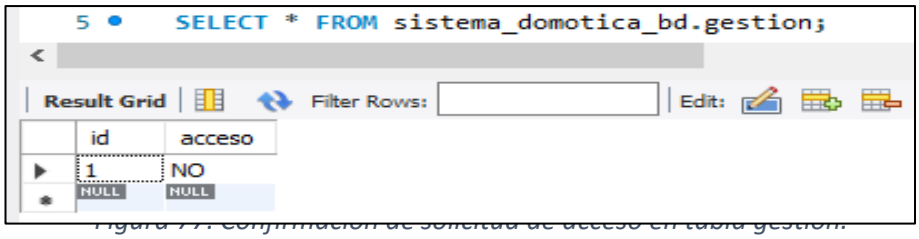


Figura 77. Confirmación de solicitud de acceso en tabla gestion.

Fuente: Elaboración propia

Se le puede indicar al chatbot que se quiere conocer la cantidad de conocidos y desconocidos en total que se han acercado a la residencia.

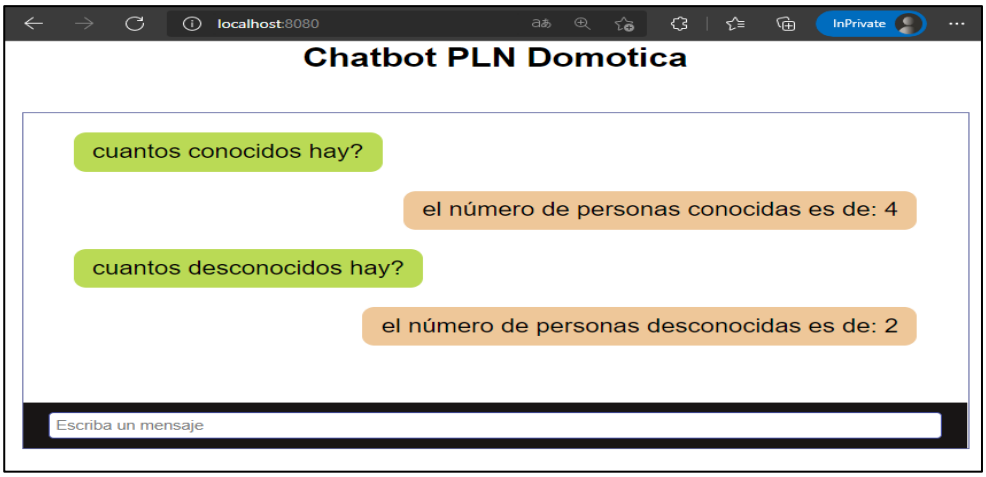


Figura 78. Mostrar Cantidad de conocidos y desconocidos en chatbot.

Fuente: Elaboración propia

En cuanto a la vigilancia, al chatbot se le puede pedir el historial de conocidos que han visitado en la puerta de la residencia.

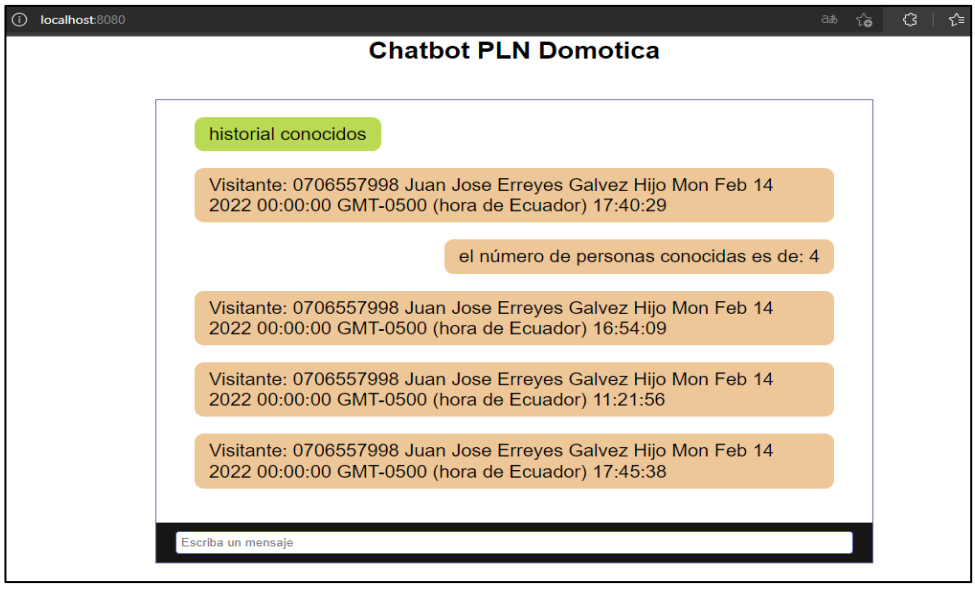


Figura 79. Historial de vigilancia conocidos en chatbot.

Fuente: Elaboración propia

Al pedir el historial de desconocidos sigue el mismo proceso anterior, este responderá indicando la información de lo que se pide, en la figura 80 se muestra el ejemplo.

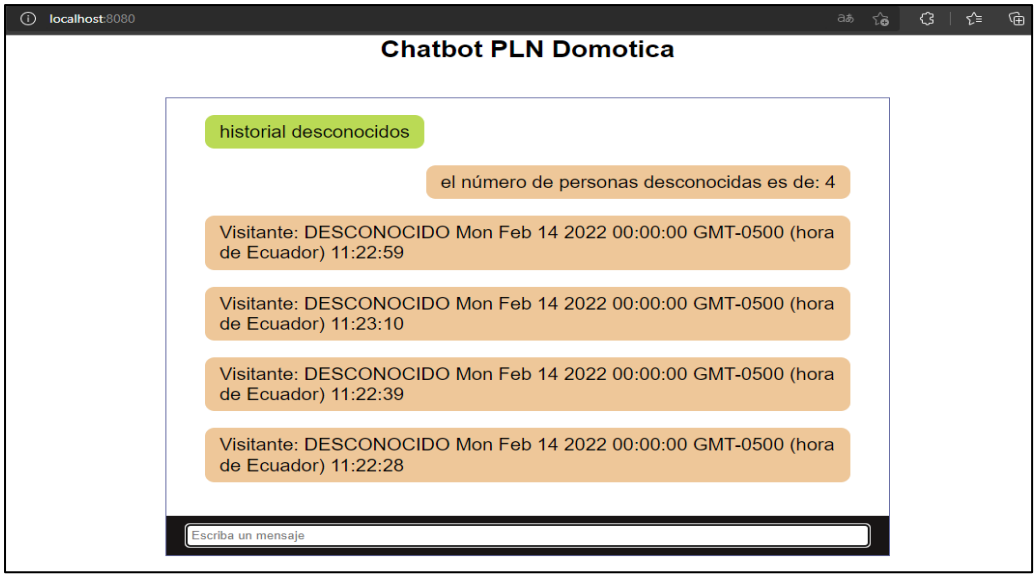


Figura 80. Historial de vigilancia desconocidos en chatbot.

Fuente: Elaboración propia

3. CAPÍTULO III: EVALUACIÓN DEL PROTOTIPO

3.1. Plan de Evaluación

El proceso de evaluación del prototipo va ser fundamental para demostrar la eficiencia y calidad del modelo o la técnica del reconocimiento facial. En este capítulo se presenta el método de evaluación para posteriormente mostrar resultados del rendimiento de la técnica de reconocimiento facial mencionada para este proyecto.

Para realizar la descripción cuantitativa de las pruebas, se obtendrán valores porcentuales que son evaluados mediante la matriz de confusión organizada en el esquema de la tabla 15.

Tabla 15. Organización de matriz de confusión.

		Valores de predicción	
Valores reales		Verdaderos Positivos VP	Falsos Positivos FP
		Falsos Negativos FN	Verdaderos Negativos VN

Fuente: Elaboración propia.

Esta matriz es evaluada con los siguientes criterios:

Tabla 16.Matriz de confusión para interpretación de resultados VP, VN, FN, FP.

Resultado	Descripción
VP = Verdaderos Positivos	El valor real es positivo y la predicción es positiva.
VN = Verdaderos Negativos	El valor real es negativo y la predicción es negativa.
FN = Falsos Negativos	El valor real es positivo y la predicción es negativa.
FP = Falsos Positivos	El valor real es negativo y la predicción es positiva.

Fuente: Elaboración propia

Los resultados se instituyen los criterios de la efectividad para el modelo propuesto, en la cual se procederá calcular las métricas como la precisión, exactitud (Acurracy), sensibilidad (TVP) y valor de referencia (F1 Score), especificidad. Estos valores encuentran definidos de la siguiente manera:

La precisión (%): es la proporción de casos positivos que se han encontrado se calcula verdaderos positivos / (verdaderos positivos + falsos positivos).

Recall o sensibilidad: es la métrica que va mostrar la capacidad que tiene el modelo en poder diferenciar los casos que son positivos de los negativos por ende se calcula verdaderos positivos / (verdaderos positivos + falsos negativos).

Especificidad: es la métrica que indica la tasa de verdaderos negativos, es decir los casos negativos que el algoritmo clasificó de manera correcta.

F1- score o valor de referencia: este es una métrica que sintetiza la precisión y la sensibilidad en conjunto por ende se calcula 2 * (sensibilidad *precisión) / (sensibilidad + precisión).

Accuracy: Es una métrica que mide la tasa porcentual de los argumentos que el modelo de clasificación ha predicho correctamente se deduce (verdaderos positivos + verdaderos negativos) / (VP + VN + FP +FN).

La sensibilidad va ser la encargada de medir la capacidad del modelo para clasificar de manera correcta a un rostro, asumiendo que los (VP) verdaderos positivos que cuentan todos los rostros son detectados y los identifica, los Falsos Negativos (FN) que cuentan todos los rostros que son detectados y reconocidos incorrectamente, es decir de los rostros que reconoce los muestra como correcto, pero en realidad son incorrectas.

Previamente comparando documentos, artículos científicos con temas similares a este trabajo y que han sido de guía del mismo, se ha tomado en consideración las siguientes pruebas experimentales:

- Análisis de rendimiento del reconocimiento facial según las distancias.
- Análisis del reconocimiento facial según niveles de luminosidad.
- Análisis del reconocimiento facial por cantidad de fotos.

Las pruebas que van a desarrollarse se relacionan con los 3 factores antes mencionado en cuanto a variación de distancias se experimenta conjuntamente con los diferentes niveles de luz y por número de fotos, a fin de poner a prueba el rendimiento del sistema de reconocimiento facial en tiempo real.

Para las pruebas del reconocimiento facial se ubicará al individuo de prueba a diferentes distancias, como en la figura 81, estas definidas de acuerdo a la lógica de las personas que tienen en el momento de ubicarse frente a la cámara. En la tabla 17 se especifica las distancias de prueba.

Tabla 17. Distancias para prueba.

Prueba	**Detalle**
Distancia 1	60 cm
Distancia 2	120 cm
Distancia 3	180 cm

Fuente: Elaboración propia.

Distancias

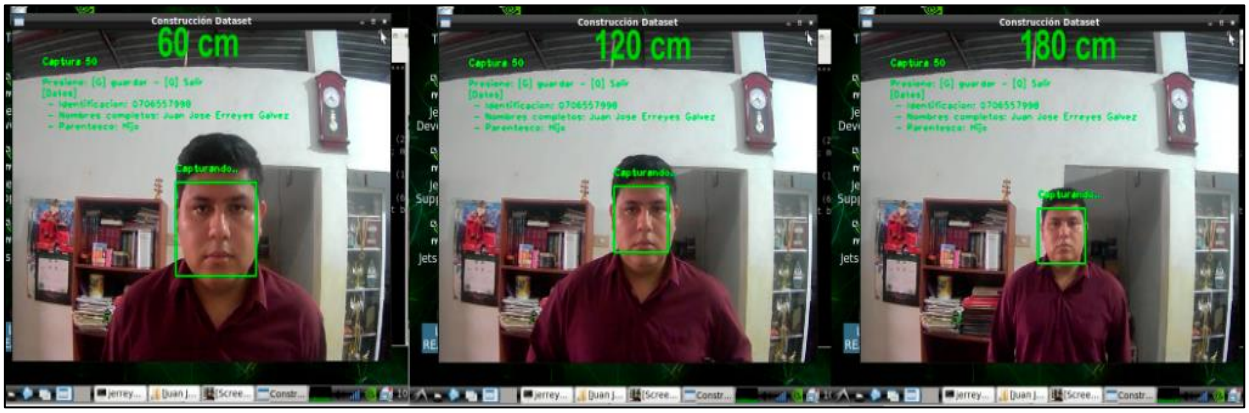


Figura 81. Distancias para pruebas del detector y reconocimiento facial.

Fuente: Elaboración propia.

Se experimentará en diferentes condiciones de iluminación como lo muestra la figura 82, estas definidas de acuerdo a las variaciones de tiempo en el día, los cuales determinan la dirección que toma la luz. Durante las pruebas no se coadyuvará con fuentes de luz artificial como focos, para poder evaluar cómo se comporta el modelo de reconocimiento facial en un entorno real. Para la medición se mantienen las distancias establecidas anteriormente. En la tabla 18 se detallan los diferentes niveles de iluminación para las pruebas.

Tabla 18. Condiciones de luz para pruebas.

Prueba	**Detalle**
Iluminación 1	Luz alta (11 am – 17 pm)
Iluminación 2	Luz baja (18 pm – 07 am)
Iluminación 3	Contraluz (8 am – 10 am)

Fuente: Elaboración propia.

Iluminación

Figura 82. Luminosidad para pruebas del detector y reconocimiento facial.

Fuente: Elaboración propia.

Para la cantidad de fotos que se tenga almacenado por cada persona es un componente importante, ya que al tener en conjunto dataset más fotografías se puede conseguir mayor efectividad al momento del reconocimiento, pero hay que tomar en consideración que esto implica más procesamiento es decir más consumo de los recursos de hardware del dispositivo. En la tabla 19 se especifica la cantidad de fotos para el entorno de pruebas.

Tabla 19. Cantidad de fotos para prueba.

Cantidad de fotos de prueba x persona
50

Fuente: Elaboración propia.

Entorno de pruebas

Se realizarán las pruebas con un conjunto de 6 personas en la que 5 se encuentran almacenadas en la base de datos y 1 no lo está. El escenario en que se va a proceder a desarrollar las pruebas correspondientes se ubica en una residencia con la finalidad de conseguir resultados en un ambiente real ya que el prototipo se lo diseño para ello, este se lo muestra en la figura 83.

Figura 83. Entorno de pruebas.

Fuente: Elaboración propia.

3.2. Resultados de la Evaluación

A continuación, realizada todas las pruebas y las evaluaciones con la matriz de confusión presentada figura 92 en anexos, la tabla 20 se muestran los resultados de precisión, acurracy, recall, f1-score del clasificador por defecto de face_recognition "KNN" evaluado con una tolerancia del 0.55% en diferentes distancias e intensidades

Tabla 20. Resultados de pruebas realizadas.

BASE DE DATOS CON 6 PERSONAS, 50 FOTOS POR PERSONA		MÉTODO KNN											
		60 cm				120 cm				180 cm			
		Precisión	Acurracy	Recall	F1-score	Precisión	Acurracy	Recall	F1-score	Precisión	Acurracy	Recall	F1-score
LUMINOSIDAD	Luz alta	94,77	98,28	94,71	81,20	91,55	97,12	91,51	78,44	87,51	95,67	87,66	75,03
	Luz baja	75,27	91,92	74,35	64,05	71,01	90,53	68,99	59,62	65,22	89,04	64,74	55,08
	Contraluz	86,35	95,45	86,87	74,20	80,89	93,55	82,72	69,87	77,28	92,28	79,38	66,81

Fuente: Elaboración propia.

Estos resultados demuestran que existe un mejor porcentaje de precisión de la detección y reconocimiento facial en el factor de luminosidad: luz alta con una

precisión del 94.77 % en una distancia de 60 cm. Lo que significa que se obtienen mejores resultados cuando sea mínimo la distancia entre la cámara de video y el rostro de la persona ya que el algoritmo demuestra tener un mejor enfoque del rostro.

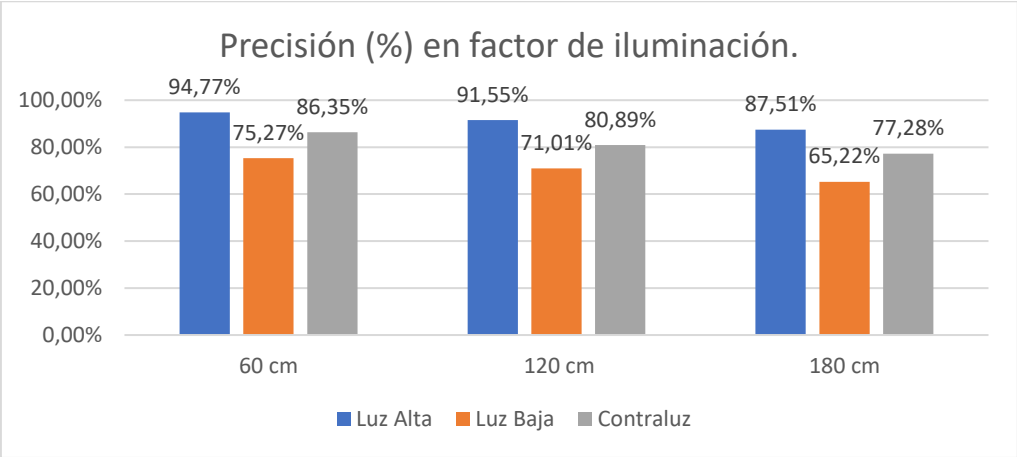


Figura 84. Precisión en factor de iluminación.

Fuente: Elaboración propia.

La luminosidad en el que se realizaron la pruebas es otro factor importante, en la figura se demuestra que al tener un ambiente con buena iluminación se va tener una mayor efectividad en el reconocimiento facial, ya que al contrario con una iluminación baja o de contraluz, en los resultados se observa que disminuye considerablemente la efectividad del reconocimiento y que el algoritmo del modelo no es competente de detectar comedidamente los rostros a baja iluminación para posteriormente realizar identificación y validación. La cantidad de fotos del rostro por cada persona es también otro factor clave, ya que, a mayor cantidad de fotos, se consigue mayor precisión para el reconocimiento, pero hay que considerar que implica más procesamiento.

A continuación, en la tabla 21 se presenta la precisión en porcentaje del reconocimiento facial en luz alta, según la distancia en las personas testeadas.

Tabla 21. Precisión en porcentaje del reconocimiento facial en las personas testeadas.

Sujetos	**60 cm**	**120 cm**	**180 cm**
	Precisión [%]	**Precisión [%]**	**Precisión [%]**
Juan	95,24%	91,18%	90,91%
Clara	91,18%	90,00%	89,74%
Amanda	93,75%	92,59%	89,29%
Cindy	90,91%	89,29%	86,21%
Alex	92,31%	91,67%	80,00%
Desconocido	97,22%	94,59%	88,89%

Fuente: Elaboración propia.

La figura 85 se muestran los resultados de precisión (%) que posee cada uno de las personas testeadas, se obtiene mejores resultados de predicción en el entorno de luminosidad alta con una distancia de 60 cm, lo que significa que el clasificador que compara las codificaciones del rostro, funciona efectivamente en este ambiente.

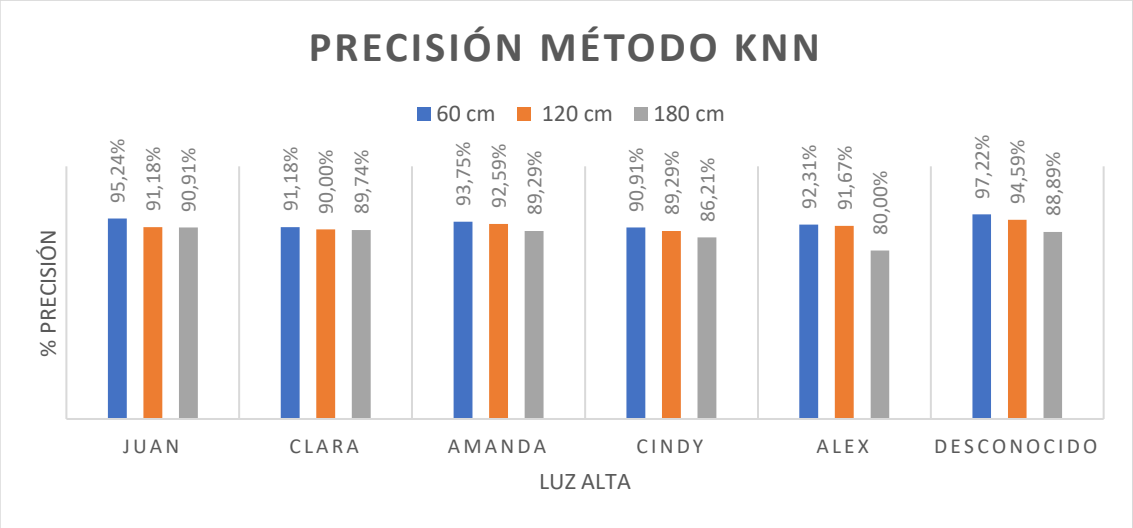


Figura 85. Precisión del método knn.

Fuente: Elaboración propia.

Ante estos resultados se puede decir que se obtuvo una velocidad media de 11 FPS para una imagen de 640x480, con una sensibilidad de 94.71% y una especificidad de 98.97%. Los rostros de la base de datos son reconocidos con una precisión del 94.77% y un acurracy del 98.28 % muy cerca del 99.38% de exactitud del modelo utilizado. Los desconocidos se identifican con una precisión del 97.22%. Estos valores se los sintetiza en la tabla 23.

Tabla 22. Resultados finales de pruebas realizadas.

Precisión %	Acurracy	Sensibilidad	Valor de Referencia F1-score	Especificidad
94,77%	98,28%	94,71%	81,20%	98,97%

Fuente: Elaboración propia.

3.3. Conclusiones

- Se desarrolló un sistema domótico con reconocimiento facial que permite la gestión y control de seguridad en una residencia, utilizando aprendizaje profundo (Deep Learning) en una jetson nano.
- Se implementó una aplicación de consola que permite el registro de información, capturas de rostros, video vigilancia en tiempo real y el acceso a personas que son validadas como conocidas y se encuentren registradas en la base de datos del sistema utilizando reconocimiento facial.
- Se desplegó una base de datos que admite almacenar y gestionar la información de registro, vigilancia y acceso conectada en una red de área local.
- Se utilizó un chatbot que permite gestionar la entrada, visualizar el historial de acceso y vigilancia de una residencia.
- Se ensambló un prototipo para la vigilancia y acceso con el uso de una cámara USB, relé y chapa eléctrica conectadas al dispositivo jetson nano 2 Gb.
- El procesamiento del sistema va a ser demasiado alto al incrementar la cantidad de fotos de rostros de personas registradas, lo que consume totalmente los recursos de hardware de la Jetson Nano produciendo una lentitud y hace que los tiempos de espera para la creación de incrustaciones y reconocimiento aumenten.
- Se destaca que los modelos de redes neuronales pre-entrenados para la construcción de incrustaciones y predictores reducen el costo computacional.
- El uso del modelo pre-entrenado "cnn" demostró ser muy preciso en cuanto a la detección del rostro en diferentes ángulos y distancias, ya que está acelerado por GPU/CUDA y la Jetson Nano posee una integrada por lo que tiene altas prestaciones para usarlo.
- Los resultados de los experimentos muestran que el algoritmo de detección y reconocimiento funcionan de forma efectiva en un sistema integrado de bajo coste. El clasificador por defecto de face_recognition “knn” o vecinos más cercanos demostró tener mejor rendimiento en condiciones de iluminación alta y distancia menor de 60 cm con una exactitud del 98.28%.

3.4. Recomendaciones

- Este sistema es recomendado para pequeños grupos de personas en una residencia, ante la limitada capacidad de recursos de hardware en el dispositivo Jetson Nano, se sugiere limitar el registro a solamente a 6 personas con 50 fotos por cada persona.
- Para poder incrementar la capacidad de registros, y disminuir los tiempos de ejecución, se recomienda registrar mínimo 15 fotos por cada persona.
- Para unificar la base de datos con el sistema en un solo dispositivo como servidor principal, se sugiere usar otros modelos de jetson nano con mayor capacidad en recursos de hardware, se deberá seleccionar dependiendo a las necesidades.
- Para mantener la precisión del 98.28% en detección y reconocimiento facial en una iluminación baja se sugiere ubicar un foco led para mantener una luminosidad alta permanentemente.
- Para mayor portabilidad se sugiere usar cámaras IP de conexión inalámbrica, además que son especializadas para la video vigilancia en factores de luminosidad variadas permite reducir las conexiones cableadas del sistema.
- Para escalar el sistema se sugiere implementar técnicas anti-falsificación de rostros, entre ellas anti-spoofing llamada también prueba de vida, en la que valida la identidad de un individuo mediante el parpadeo de ojos. Para la detección de objetos en la cara como gafas, mascarillas, se deberá usar otros modelos detección y reconocimiento ya que la jetson nano está preparada para ejecutar algoritmos de inteligencia artificial.

4. BIBLIOGRAFÍA

[1] P. Bhatia, S. Rajput, S. Pathak, y S. Prasad, «IOT based facial recognition system for home security using LBPH algorithm», en *2018 3rd International Conference on Inventive Computation Technologies (ICICT)*, nov. 2018, pp. 191-193. doi: 10.1109/ICICT43934.2018.9034420.

[2] X. Zhang, W.-J. Yi, y J. Saniie, «Home Surveillance System using Computer Vision and Convolutional Neural Network», en *2019 IEEE International Conference on Electro Information Technology (EIT)*, may 2019, pp. 266-270. doi: 10.1109/EIT.2019.8833773.

[3] T. Chakraborty y S. K. Datta, «Home automation using edge computing and Internet of Things», en *2017 IEEE International Symposium on Consumer Electronics (ISCE)*, nov. 2017, pp. 47-49. doi: 10.1109/ISCE.2017.8355544.

[4] A. Mathew y N. R. Mahanta, «Artificial Intelligence for Smart Interiors - Colours, Lighting and Domotics», en *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, jun. 2020, pp. 1335-1338. doi: 10.1109/ICRITO48877.2020.9197890.

[5] S. Almabdy y L. Elrefaei, «Deep Convolutional Neural Network-Based Approaches for Face Recognition», *Appl. Sci.*, vol. 9, n.º 20, Art. n.º 20, ene. 2019, doi: 10.3390/app9204397.

[6] N. Saxena y D. Varshney, «Smart Home Security Solutions using Facial Authentication and Speaker Recognition through Artificial Neural Networks», *Int. J. Cogn. Comput. Eng.*, vol. 2, pp. 154-164, jun. 2021, doi: 10.1016/j.ijcce.2021.10.001.

[7] P. Jaturawat y M. Phankokkruad, «An evaluation of face recognition algorithms and accuracy based on video in unconstrained factors», en *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, nov. 2016, pp. 240-244. doi: 10.1109/ICCSCE.2016.7893578.

[8] S. M. Bah y F. Ming, «An improved face recognition algorithm and its application in attendance management system», *Array*, vol. 5, p. 100014, mar. 2020, doi: 10.1016/j.array.2019.100014.

[9] M. Dhyani y R. Kumar, «An intelligent Chatbot using deep learning with Bidirectional RNN and attention model», *Mater. Today Proc.*, vol. 34, pp. 817-824, ene. 2021, doi: 10.1016/j.matpr.2020.05.450.

[10] T. Binford, J. Ruby, P. D, J. Nedumaan, J. Lepika, y J. Tisa, *Modern Deep Learning and Advanced Computer Vision [Book]*. 2019.

[11] D. Liu, N. Bellotto, y S. Yue, «Deep Spiking Neural Network for Video-Based Disguise Face Recognition Based on Dynamic Facial Movements», *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, n.º 6, pp. 1843-1855, jun. 2020, doi: 10.1109/TNNLS.2019.2927274.

[12] S. Wang, B. Pan, H. Chen, y Q. Ji, «Thermal Augmented Expression Recognition», *IEEE Trans. Cybern.*, vol. 48, n.º 7, pp. 2203-2214, jul. 2018, doi: 10.1109/TCYB.2017.2786309.

[13] W. Caijun, J. Xi, y Z. Zhenzhou, «Analysis of Systematic Reform of Future Teaching in the Age of Artificial Intelligence», en *2021 2nd International Conference on Artificial Intelligence and Education (ICAIE)*, jun. 2021, pp. 704-707. doi: 10.1109/ICAIE53562.2021.00154.

[14] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, y M. S. Lew, «Deep learning for visual understanding: A review», *Neurocomputing*, vol. 187, pp. 27-48, abr. 2016, doi: 10.1016/j.neucom.2015.09.116.

[15] M. D. RADU, I. M. COSTEA, y V. A. STAN, «Automatic Traffic Sign Recognition Artificial Inteligence - Deep Learning Algorithm», en *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, jun. 2020, pp. 1-4. doi: 10.1109/ECAI50035.2020.9223186.

[16] D. Santos, L. Dallos, P. A. Gaona-García, D. Santos, L. Dallos, y P. A. Gaona-García, «Algoritmos de rastreo de movimiento utilizando técnicas de inteligencia artificial y

machine learning», *Inf. Tecnológica*, vol. 31, n.º 3, pp. 23-38, jun. 2020, doi: 10.4067/S0718-07642020000300023.
[17] X. Zhang y S. Xu, «Research on Image Processing Technology of Computer Vision Algorithm», en *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, jul. 2020, pp. 122-124. doi: 10.1109/CVIDL51233.2020.00030.
[18] T. Lindner, D. Wyrwał, M. Białek, y P. Nowak, «Face recognition system based on a single-board computer», en *2020 International Conference Mechatronic Systems and Materials (MSM)*, jul. 2020, pp. 1-6. doi: 10.1109/MSM49833.2020.9201668.
[19] S. Ampamya, J. M. Kitayimbwa, y M. C. Were, «Performance of an open source facial recognition system for unique patient matching in a resource-limited setting», *Int. J. Med. Inf.*, vol. 141, p. 104180, sep. 2020, doi: 10.1016/j.ijmedinf.2020.104180.
[20] «IBM Docs», 17 de agosto de 2021. https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/spss-modeler/SaaS?topic=networks-neural-model (accedido 21 de enero de 2022).
[21] L. Quiñones Huatangari *et al.*, «Red neuronal artificial para estimar un índice de calidad de agua», *Enfoque UTE*, vol. 11, n.º 2, pp. 109-120, jun. 2020, doi: 10.29019/enfoque.v11n2.633.
[22] A. A. Wazwaz, A. O. Herbawi, M. J. Teeti, y S. Y. Hmeed, «Raspberry Pi and computers-based face detection and recognition system», en *2018 4th International Conference on Computer and Technology Applications (ICCTA)*, may 2018, pp. 171-174. doi: 10.1109/CATA.2018.8398677.
[23] «CS231n Convolutional Neural Networks for Visual Recognition». https://cs231n.github.io/convolutional-networks/#overview (accedido 21 de enero de 2022).
[24] K. He, X. Zhang, S. Ren, y J. Sun, «Deep Residual Learning for Image Recognition», *ArXiv151203385 Cs*, dic. 2015, Accedido: 22 de enero de 2022. [En línea]. Disponible en: http://arxiv.org/abs/1512.03385
[25] A. Basulto-Lantsova, J. A. Padilla-Medina, F. J. Perez-Pinal, y A. I. Barranco-Gutierrez, «Performance comparative of OpenCV Template Matching method on Jetson TX2 and Jetson Nano developer kits», en *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, ene. 2020, pp. 0812-0816. doi: 10.1109/CCWC47524.2020.9031166.
[26] «Welcome to Python.org», *Python.org*. https://www.python.org/ (accedido 22 de enero de 2022).
[27] Alexander Mordvintsev & Abid K, *OpenCV-Python Tutorials*. 2017. [En línea]. Disponible en: https://opencv24-python-tutorials.readthedocs.io/_/downloads/en/stable/pdf/
[28] N. Boyko, O. Basystiuk, y N. Shakhovska, «Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library», en *2018 IEEE Second International Conference on Data Stream Mining Processing (DSMP)*, ago. 2018, pp. 478-482. doi: 10.1109/DSMP.2018.8478556.
[29] N. Hema y J. Yadav, «Secure Home Entry Using Raspberry Pi with Notification via Telegram», en *2020 6th International Conference on Signal Processing and Communication (ICSC)*, mar. 2020, pp. 211-215. doi: 10.1109/ICSC48311.2020.9182778.
[30] C. J. Baby, F. A. Khan, y J. N. Swathi, «Home automation using IoT and a chatbot using natural language processing», en *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, abr. 2017, pp. 1-6. doi: 10.1109/IPACT.2017.8245185.
[31] D. Caina, R. Carillo, M. Carrillo, D. Caina, R. Carillo, y M. Carrillo, «Técnicas de Visión Computacional para determinar el estado fitosanitario en plantaciones de brócoli», *Siembra*, vol. 4, n.º 1, pp. 51-58, dic. 2017, doi: 10.29166/siembra.v4i1.499.
[32] «Biblioteca dlib C++». http://dlib.net/ (accedido 11 de enero de 2022).

[33] A. Geitgey, *face-recognition: Recognize faces from Python or from the command line*. Accedido: 11 de enero de 2022. [En línea]. Disponible en: https://github.com/ageitgey/face_recognition
[34] «OpenCV Haar Cascades», *PyImageSearch*, 12 de abril de 2021. https://www.pyimagesearch.com/2021/04/12/opencv-haar-cascades/ (accedido 24 de enero de 2022).
[35] «Face recognition with OpenCV, Python, and deep learning», *PyImageSearch*, 18 de junio de 2018. https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/ (accedido 22 de enero de 2022).
[36] D. E. King, *modelos dlib*. 2022. Accedido: 24 de enero de 2022. [En línea]. Disponible en: https://github.com/davisking/dlib-models
[37] «face_recognition package — Face Recognition 1.4.0 documentation». https://face-recognition.readthedocs.io/en/latest/face_recognition.html (accedido 24 de enero de 2022).
[38] «Deepcaps: Going deeper with capsule networks». https://scholar.google.co.uk/citations?view_op=view_citation&hl=da&user=98ItndMAAAAJ&citation_for_view=98ItndMAAAAJ:u-x6o8ySG0sC (accedido 29 de enero de 2022).
[39] K. Zhang, Z. Zhang, Z. Li, y Y. Qiao, «Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks», *IEEE Signal Process. Lett.*, vol. 23, n.º 10, pp. 1499-1503, oct. 2016, doi: 10.1109/LSP.2016.2603342.
[40] A. Geitgey, «Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning», *Medium*, 24 de septiembre de 2020. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78 (accedido 24 de enero de 2022).
[41] N. Rao, M. H. Rajan, K. Rebello, y S. Gharat, «Energy-efficient Face Recognition Authentication System Using Human Detection IoT Modules», en *2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)*, abr. 2020, pp. 125-129. doi: 10.1109/CSCITA47329.2020.9137776.
[42] S. Sivakumar y R. G. Bhavani, «Image Processing Based System for Intrusion Detection and Home Security Enhancement», en *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, may 2018, pp. 1676-1680. doi: 10.1109/RTEICT42901.2018.9012295.
[43] C. W. Okonkwo y A. Ade-Ibijola, «Chatbots applications in education: A systematic review», *Comput. Educ. Artif. Intell.*, vol. 2, p. 100033, ene. 2021, doi: 10.1016/j.caeai.2021.100033.
[44] M. Y. Sai, R. V. C. Prasad, P. R. Niveditha, T. Sasipraba, S. Vigneshwari, y S. Gowri, «Low cost automated facial recognition system», en *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, feb. 2017, pp. 1-5. doi: 10.1109/ICECCT.2017.8117829.
[45] «Jetson Nano 2GB Developer Kit User Guide», *NVIDIA Developer*, 5 de octubre de 2020. https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide (accedido 11 de enero de 2022).
[46] por E. G. Novelec |, «Cerradura eléctrica: tipos y características», *Grupo Novelec*. https://blog.gruponovelec.com/redes-vdi/cerradura-electrica-tipos-y-caracteristicas/ (accedido 11 de enero de 2022).
[47] «El Relé: para qué es, para qué sirve y qué tipos existen | Blog SEAS», 22 de agosto de 2019. https://www.seas.es/blog/automatizacion/el-rele-para-que-es-para-que-sirve-y-que-tipos-existen/ (accedido 11 de enero de 2022).

Anexos

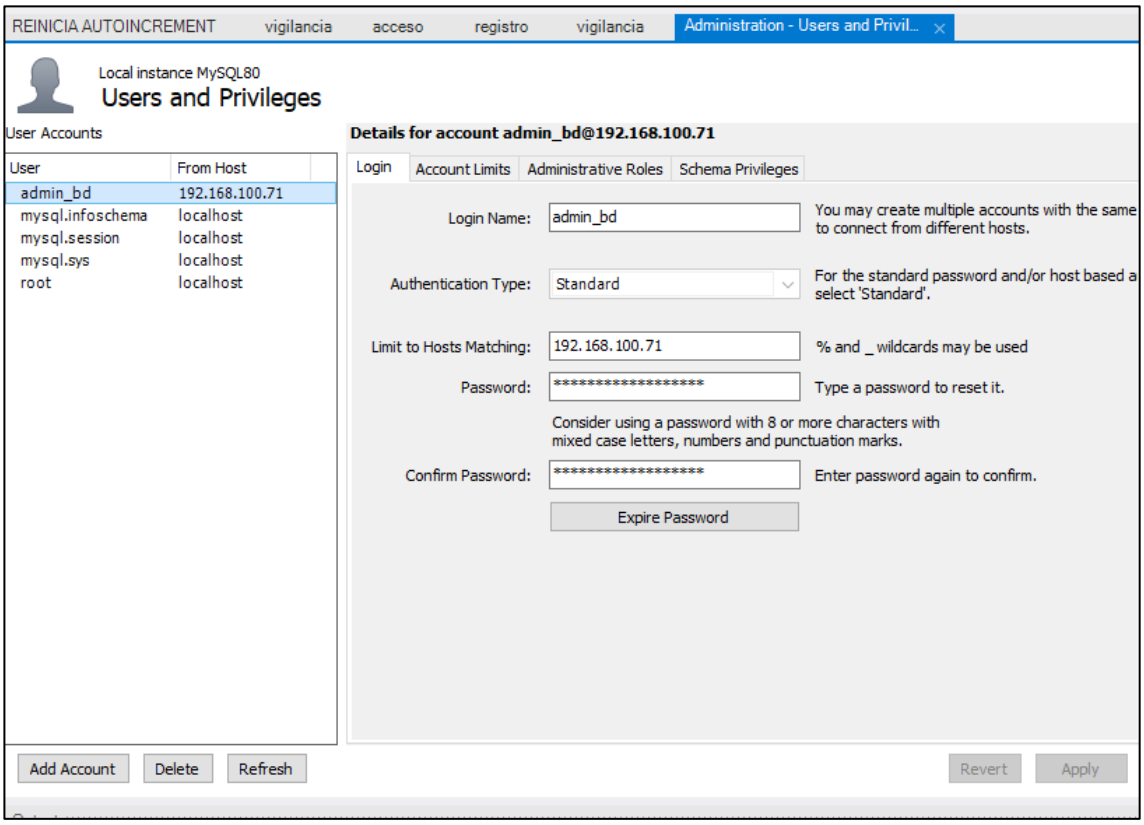


Figura 86. Configuración de conexión a base de datos para red local.

Fuente: Elaboración propia.

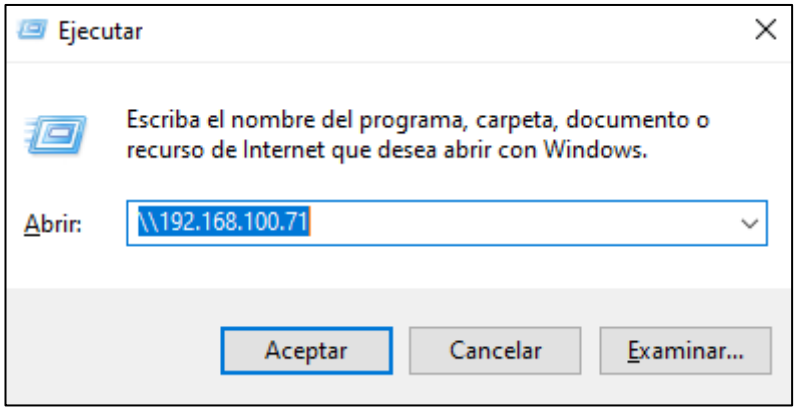


Figura 87. Acceso a carpeta compartida.

Fuente: Elaboración propia.

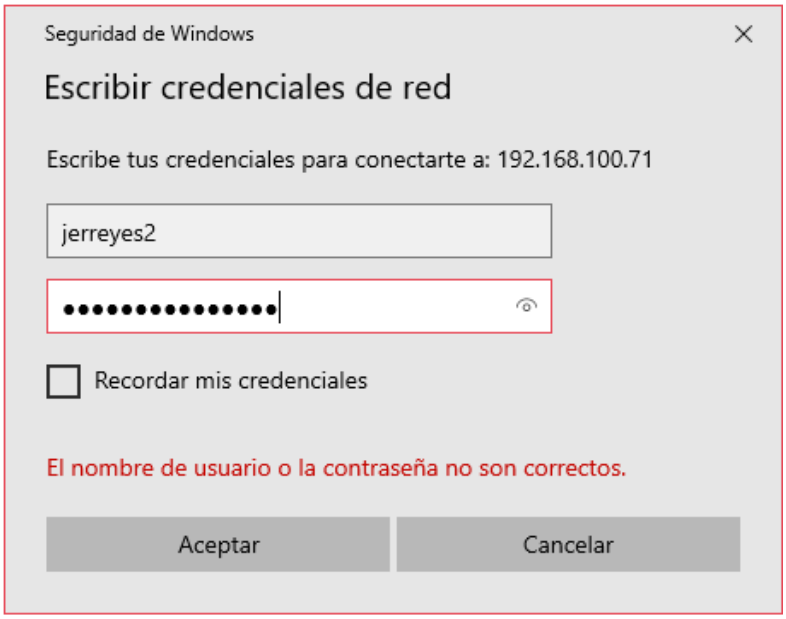


Figura 88. Credenciales para acceso a carpeta compartida.

Fuente: Elaboración propia.

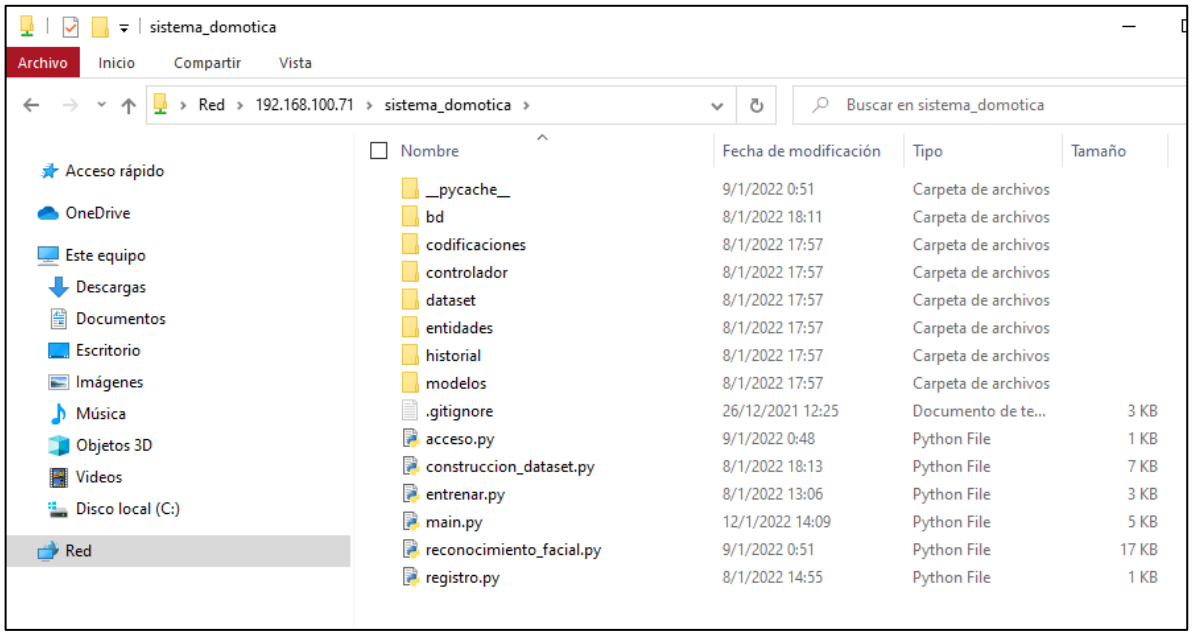


Figura 89. Acceso a carpeta compartida del sistema.

Fuente: Elaboración propia.

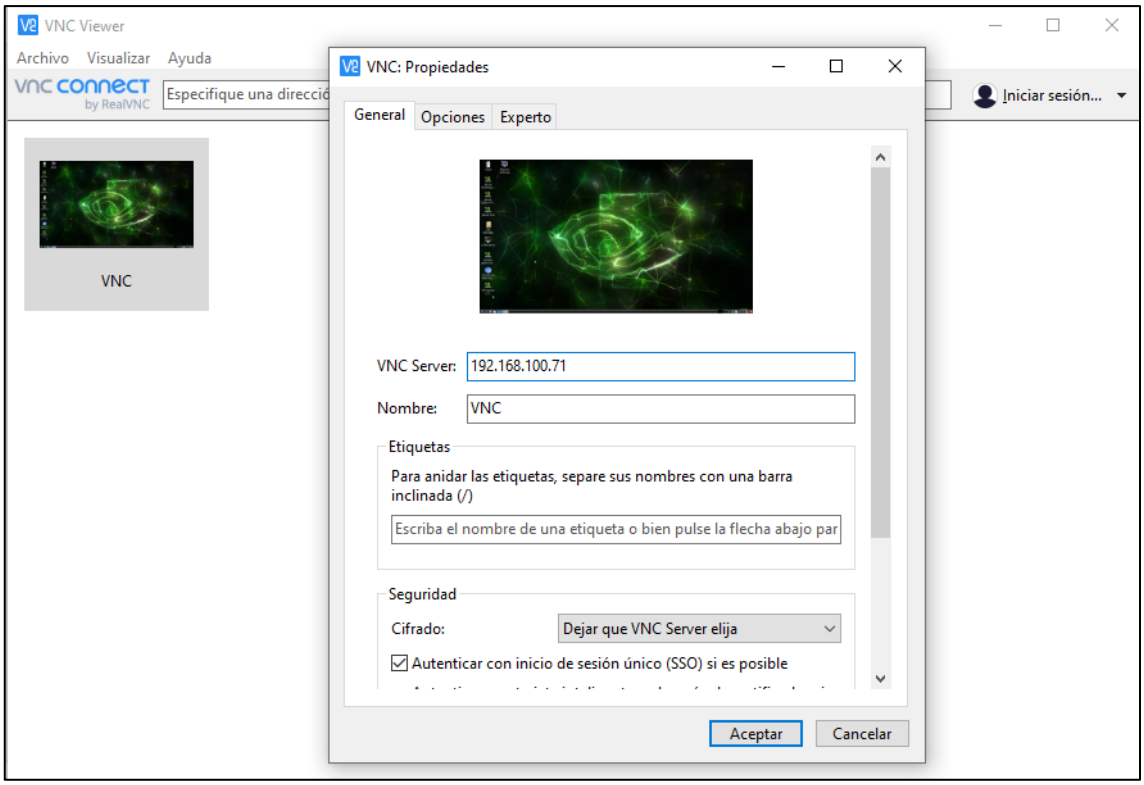


Figura 90. Credenciales para acceso remoto de jetson nano por medio de VNC Viewer.

Fuente: Elaboración propia.

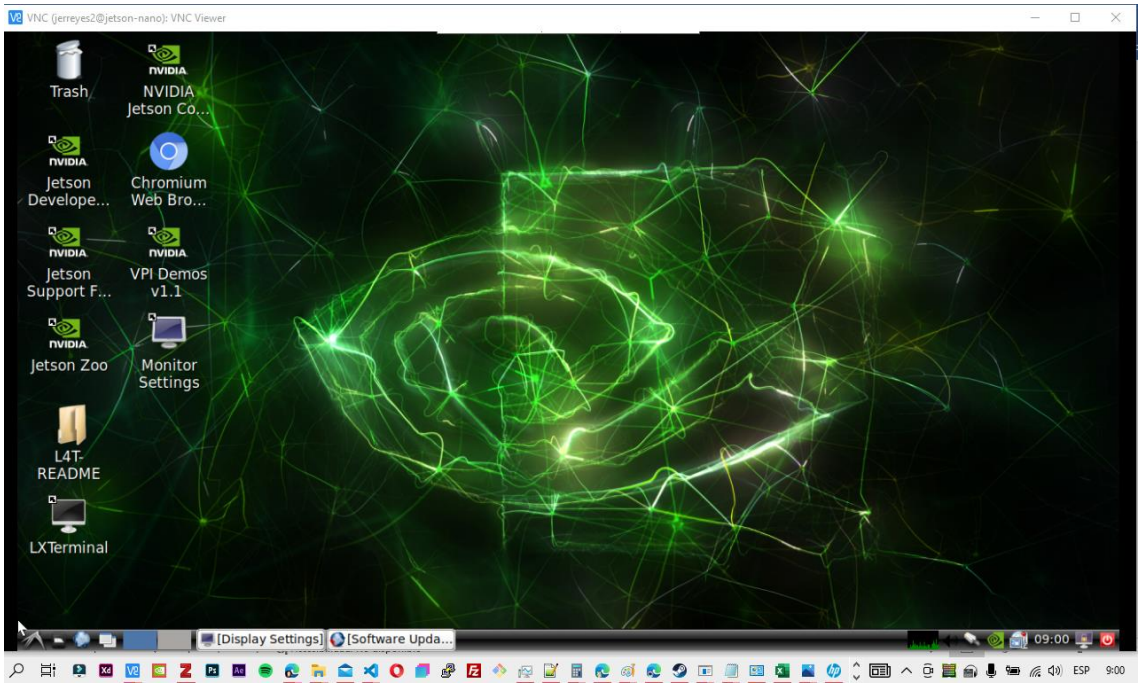
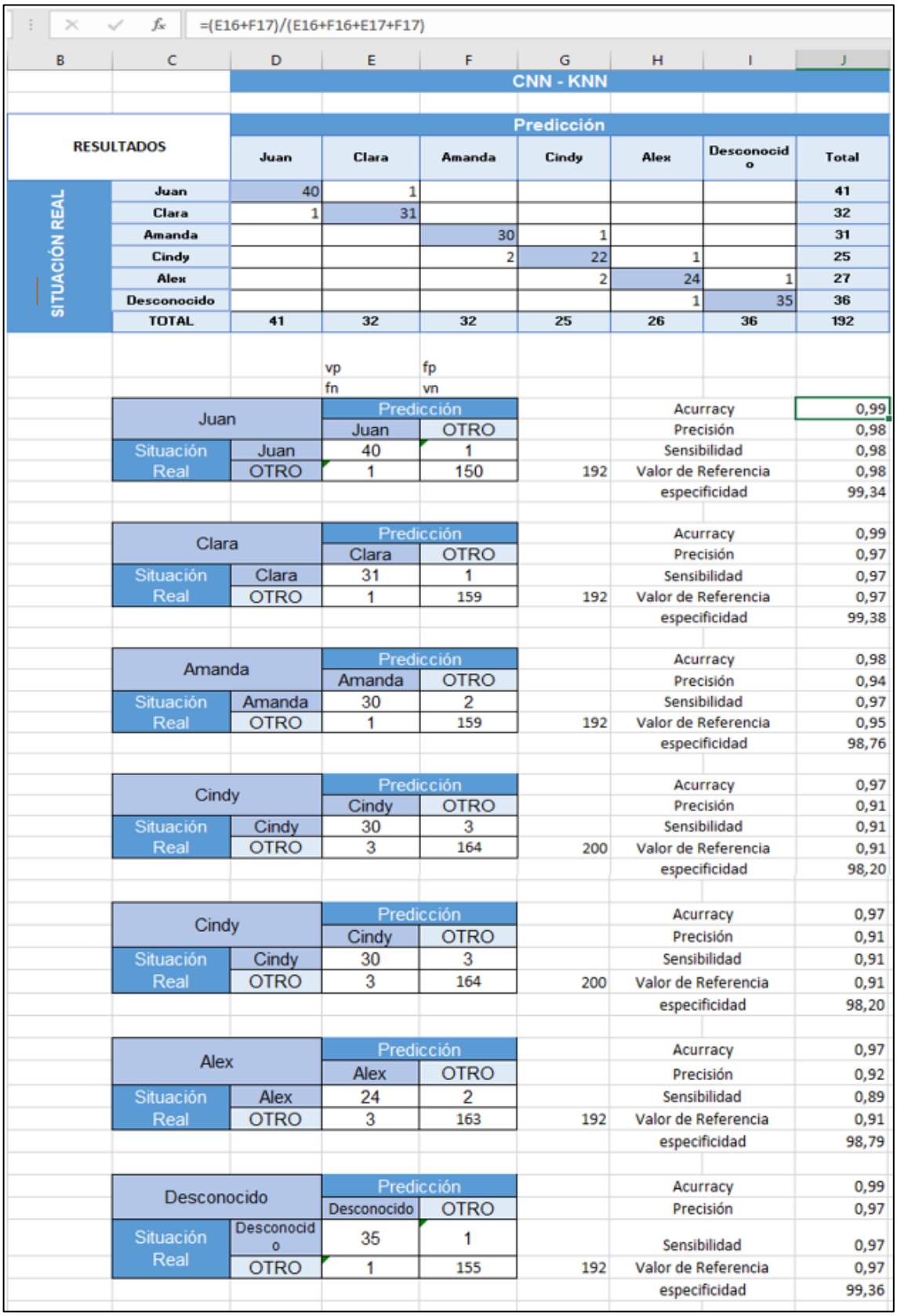


Figura 91. Acceso remoto de Jetson Nano.

Fuente: Elaboración propia.



=(E16+F17)/(E16+F16+E17+F17)

CNN - KNN

RESULTADOS		Predicción						
		Juan	Clara	Amanda	Cindy	Alex	Desconocido	Total
SITUACIÓN REAL	Juan	40	1					41
	Clara	1	31					32
	Amanda			30	1			31
	Cindy			2	22	1		25
	Alex				2	24	1	27
	Desconocido					1	35	36
	TOTAL	41	32	32	25	26	36	192

vp fp
fn vn

Juan		Predicción	
		Juan	OTRO
Situación Real	Juan	40	1
	OTRO	1	150

192

Acurracy	0,99
Precisión	0,98
Sensibilidad	0,98
Valor de Referencia	0,98
especificidad	99,34

Clara		Predicción	
		Clara	OTRO
Situación Real	Clara	31	1
	OTRO	1	159

192

Acurracy	0,99
Precisión	0,97
Sensibilidad	0,97
Valor de Referencia	0,97
especificidad	99,38

Amanda		Predicción	
		Amanda	OTRO
Situación Real	Amanda	30	2
	OTRO	1	159

192

Acurracy	0,98
Precisión	0,94
Sensibilidad	0,97
Valor de Referencia	0,95
especificidad	98,76

Cindy		Predicción	
		Cindy	OTRO
Situación Real	Cindy	30	3
	OTRO	3	164

200

Acurracy	0,97
Precisión	0,91
Sensibilidad	0,91
Valor de Referencia	0,91
especificidad	98,20

Cindy		Predicción	
		Cindy	OTRO
Situación Real	Cindy	30	3
	OTRO	3	164

200

Acurracy	0,97
Precisión	0,91
Sensibilidad	0,91
Valor de Referencia	0,91
especificidad	98,20

Alex		Predicción	
		Alex	OTRO
Situación Real	Alex	24	2
	OTRO	3	163

192

Acurracy	0,97
Precisión	0,92
Sensibilidad	0,89
Valor de Referencia	0,91
especificidad	98,79

Desconocido		Predicción	
		Desconocido	OTRO
Situación Real	Desconocido	35	1
	OTRO	1	155

192

Acurracy	0,99
Precisión	0,97
Sensibilidad	0,97
Valor de Referencia	0,97
especificidad	99,36

Figura 92. Evaluación con matriz de confusión.

Fuente: Elaboración propia.