ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

"IMPLEMENTACIÓN DE UN PROTOTIPO DE DETECCIÓN DE MASCARRILLA Y MEDICIÓN DE TEMPERATURA PARA EL PERSONAL QUE INGRESA A LA EMPRESA PAUFIT, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL"

Trabajo de titulación

Tipo: Dispositivo Tecnológico

Presentado para optar al grado académico de:

INGENIERA EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTORA: ALEXANDRA ELIZABETH PAULLÁN PUNGUIL

DIRECTOR: DR. GEOVANNY ESTUARDO VALLEJO VALLEJO

Riobamba – Ecuador

2021

Yo, Alexandra Elizabeth Paullán Punguil, declaro que el presente trabajo de titulación es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autora asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 07 de diciembre del 2021.

Alexandra Elizabeth Paullán Punguil
C.I: 0604580191

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal de Trabajo de Titulación certifica que: El trabajo de titulación: Tipo: Dispositivo Tecnológico: **"IMPLEMENTACIÓN DE UN PROTOTIPO DE DETECCIÓN DE MASCARILLA Y MEDICIÓN DE TEMPERATURA PARA EL PERSONAL QUE INGRESA A LA EMPRESA PAUFIT, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL"** de responsabilidad de la señorita **Alexandra Elizabeth Paullán Punguil**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal autoriza su presentación.

	FIRMA	FECHA
Ing. Diego Ramiro Ñacato Estrella Msc. **PRESIDENTE DEL TRIBUNAL**	Firmado electrónicamente por: DIEGO RAMIRO NACATO ESTRELLA	2021-07-12
Dr. Geovanny Estuardo Vallejo Vallejo Msc. **DIRECTOR DEL TRABAJO DE TITULACIÓN**	GEOVANNY ESTUARDO VALLEJO VALLEJO Firmado digitalmente por GEOVANNY ESTUARDO VALLEJO VALLEJO Fecha: 2021.12.07 14:51:55 -05'00'	2021-07-12
Ing. José Luis Tinajero León Msc. **MIEMBRO DEL TRIBUNAL**	Firmado electrónicamente por: JOSE LUIS TINAJERO	2021-07-12

DEDICATORIA

El presente trabajo de titulación se lo dedico en primer lugar a Dios, que sin su ayuda no hubiese sido posible llegar a este momento, a mis padres Oswaldo y Elena, que han sido el pilar fundamental para poder alcanzar este logro que muchas veces por circunstancias de la vida parecía estar cada vez más lejos, que sin su apoyo, su amor, sus palabras de aliento, su fe en mí y sobre todo su compresión, no habría sido posible hoy conseguir esto. A mis hermanos Dennys y Génesis por creer en mí, entenderme, apoyarme, mostrarme su amor y por alegrarme siempre con sus ocurrencias, son los mejores hermanos que la vida me pudo dar, con todo el amor este logro se los dedico a ustedes.

Alexandra

AGRADECIMIENTO

No me alcanza las palabras para expresar mi gratitud, en primer lugar a Dios por lo que ha hecho en mi vida, por haberme guiado, acompañado y haberme permitido llegar a este momento y cumplir con este sueño, a mis padres Oswaldo y Elena porque siempre lucharon por darnos lo mejor a mí y a mis hermanos, les agradezco mucho por el ejemplo de perseverancia, dedicación, amor y esfuerzo. Por no dejarme nunca sola, por siempre apoyarme, confiar en mí y por ser ese apoyo incondicional. A mi hermano Dennys por que con sus ocurrencias siempre me alegraba, a mi hermana Génesis por su amor y sobre todo por su paciencia, gracias por todos los momentos de alegría y tristeza porque siempre nos hemos mantenido juntos y por qué la alegría de uno siempre ha sido la alegría de todos y la tristeza de uno siempre será la tristeza de todos, los amo mucho son los mejores hermanos y padres que Dios me pudo dar.

A mis tíos que siempre me apoyaron de una u otra manera gracias por estar pendientes de mí, también de manera especial a mis Tíos Marina y Ruperto quienes siempre estuvieron prestos a brindarme su ayuda. A mis abuelitos Magdalena, Rosa y Mentor, a mis tías Rosa y Lola por estar siempre acompañándome. A Daniel, que ha estado conmigo a lo largo de la carrera, gracias por su paciencia y amor, gracias por el apoyo, la ayuda incondicional que siempre me brindo y por todos los momentos que vivimos juntos. Al Dr., Geovanny Vallejo y al Ing., José Luis Tinajero por compartir sus conocimientos, por la paciencia y por la ayuda oportuna para la culminación de este trabajo. A todos ustedes de todo corazón mil gracias.

Alexandra

TABLA DE CONTENIDO

ÍNDICE DE TABLAS ..... **xi**

ÍNDICE DE FIGURAS ..... **xii**

ÍNDICE DE ANEXOS ..... **xv**

RESUMEN ..... **xvi**

SUMMARY ..... **xvii**

INTRODUCCIÓN **1**

CAPÍTULO I

1. MARCO TEÓRICO **5**

1.1 Epidemia COVID-19 ..... **5**

1.1.1 Medidas para evitar el contagio del Covid-19 ..... ***5***

1.1.1.1 Uso de mascarilla ***5***

1.2 Dispositivos de detección de Mascarilla **6**

1.2.1 Sistemas comerciales para detección de mascarilla y toma de temperatura ***8***

1.3 Inteligencia Artificial **10**

1.3.1 Visión Artifcial ***11***

1.3.1.1 Reconocimiento de objetos. ***12***

1.3.2 Adquisición de Imágenes ***13***

1.*3.2.1 La cámara* ***13***

1.3.3 Pre procesamiento ***14***

1.4 Detección de imágenes **15**

1.4.1 Deep Learning 15

1.***4.2 Algoritmos***..... 16

1.4.2.1 Redes Neuronales ***16***

1.4.2.2 Redes Neuronales Convolucionales ***17***

1.4.3 TensorFlow ***17***

1.4.3.1 Arquitectura MobilenetV2 ..... ***18***

1.5 Temperatura Corporal **19**

1.5.1 Adelantos tecnológicos en la medición de temperatura corporal ***20***

1.6 Sensores y Transductores 20

1.6.1 Tipos de sensores 21

1.7 Python 22

1.8 Raspberry 23

1.9 Arduino 24

1.10 NodeMCU EP8266 25

CAPÍTULO II

2. METODOLOGÍA 25

2.1 Requerimientos del prototipo 25

2.1.1 Requerimientos de Hardware 26

2.1.2 Requerimientos de Software 26

2.2 Arquitectura del prototipo 27

2.3 Hardware 27

2.3.1 Adquisición de información 27

2.3.2 Unidad de Procesamiento 28

2.3.3 Arduino Uno 29

2.3.4 Amiplificador de Audio PAM8403 30

2.3.5 Placa de desarrollo NodeMCU ESP8266 31

2.3.6 Sensor Ultrasónico 32

2.3.7 Sensor de temperatura Infrarrojo 33

2.3.8 Tarjeta de Acondicionamiento 33

2.3.9 Modulo Rele 34

2.3.10 Conexiones del Hardware 35

2.4 Software 36

2.4.1 Montaje del sistema Operativo en la Raspberry Pi 36

2.4.1.1 Instalacion de Python 3.7 37

2.4.1.2 Instalación de OpenCV 37

2.4.1.3 Instalacion de TensorFlow y librerias complementarias 38

2.4.2 Instalación del Protocolo MQTT en la Raspberry Pi 39

2.4.3 Configuración de la pantalla de 3.5 pulgadas para la visualización 40

2.4.4 Programa de recolección de imágenes para el entrenamiento 40

2.4.4.1 Entrenamiento del modelo de Red Neuronal 42

2.4.4.2 Importación de librerias usadas para el entrenamiento de la red neuronal..... 43

2.4.4.3 Analizador de argumentos para la creación de variables 45

2.4.4.4 Asignación de tasa de aprendizaje, número de épocas y el tamaño del lote 46

2.4.5 Preprocesamiento y conversión de imágenes de la base de datos. 46

2.4.6 Segmentación de la base de datos 47

2.4.7 Creación de nuevas imágenes a partir de la base de datos 48

2.4.8 Configuración de parámetros del modelo pre entrenado MobileNetV2 49

2.4.8.1 Construcción de la Arquitectura en la cabecera del modelo base 49

2.4.8.2 Compilación del modelo de Red Neuronal 51

2.4.8.3 Evaluación y validación del entrenamiento 52

2.4.9 Programa principal para detecccción de mascarilla y temperatura 53

2.4.10 Algoritmo para la etapa de procesamiento de imágenes. 57

2.4.11 Programa de adquisición de temperatura y distancia en Arduino 60

2.4.12 Protocolo de comunicación para control de acceso 62

2.4.12.1 Implementación del protocolo MQTT en el proceso de trasmisión 64

2.4.12.2 Circuito de acondicionamiento de voltaje y amplificción de audio 66

2.4.13 Diseño de la estructura del prototipo en SolidWorks 67

CAPÍTULO III

3. MARCO DE RESULTADOS 68

3.1 Validación del sistema 68

3.1.1 Pruebas y resultados de funcionamiento para detectar de uso de mascarilla 68

3.1.2 Pruebas y resultados de la medición de temperatura corporal 76

3.1.3 Análisis de tiempo de respuesta 77

3.1.3.1 Tiempo de respuesta para la detección de mascarilla y toma de temperatura.... 78

3.2 Consumo de energía del dispositivo 79

CAPÍTULO IV

4. ANÁLISIS DE COSTOS DEL PROTOTIPO IMPLEMENTADO..... 80

4.1 Costos Directos 80

4.2 Comparación de costos entre dispositivos..... 81

CONCLUSIONES..... 83

RECOMENDACIONES..... 83

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE TABLAS

Tabla 1-2: Rangos de temperatura Corporal 20
Tabla 2-2: Características Cámara para Raspberry Pi 28
Tabla 3-2: Características de la Raspberry Pi4 29
Tabla 4-2: Especificaciones técnicas de la placa Arduino UNO 30
Tabla 5-2: Especificaciones técnicas PAM8403 30
Tabla 6-2: Especificaciones técnicas de la placa NodeMCU 32
Tabla 7-2: Especificaciones técnicas del sensor ultrasónico 32
Tabla 8-2: Especificaciones técnicas del sensor de temperatura 33
Tabla 9-2: Especificaciones técnicas del sensor de temperatura 34
Tabla 1-3: Pruebas de detección de mascarilla realizadas en la mañana 69
Tabla 2-3: Pruebas de detección de mascarilla realizadas al medio día 70
Tabla 3-3: Pruebas de detección de mascarilla realizadas a la media tarde 71
Tabla 4-3: Pruebas de detección de mascarilla realizadas en la noche 72
Tabla 5-3: Resultados de las pruebas realizadas en los diferentes escenarios 73
Tabla 6-3: Resultados de las pruebas realizadas para determinar la distancia adecuada 74
Tabla 7-3: Resultados obtenidos por medición de temperatura día 1 76
Tabla 8-3: Resultados obtenidos por medición de temperatura día 2 77
Tabla 9-3: Resultados obtenidos por tiempos de respuesta 78
Tabla 10-3: Consumo en corriente del dispositivo 79
Tabla 1-4: Costo directos de implementación del prototipo 80
Tabla 2-4: Comparación entre dispositivos de detección de mascarilla y temperatura 81

ÍNDICE DE FIGURAS

Figura 1-1. Uso correcto de la mascarilla. 6
Figura 2-1. Representación gráfica de una red convolucional 7
Figura 3-1. Ejemplo del clasificar en cascada Haar 7
Figura 4-1. Dispositivo de Automatic System 9
Figura 5-1. Dispositivos desarrollados por Hikvision 10
Figura 6-1. Dispositivo de detección implementado por By Demes Group 10
Figura 7-1. Representación de un sistema de visión por computadora 12
Figura 8-1. Cámaras para aplicaciones de visión artificial 13
Figura 9-1. Representación de una Red Neuronal. 16
Figura 10-1. Estructura de una Red Convolucional 17
Figura 11-1. Representación de tensores 18
Figura 12-1. Bloques convolucionales de la red Mobilenet V1 y V2 19
Figura 13-1. Sensor de distancia infrarrojo 21
Figura 14-1. Sensor de temperatura 22
Figura 15-1. Sensor de proximidad inductivo 22
Figura 16-1. Logo de Python. 23
Figura 17-1. Raspberry Pi **4** 24
Figura 18-1. Arduino UNO distribución de terminales 24
Figura 19-1. NodeMCU ESP8266 25
Figura 1-2. Arquitectura del sistema de detección de mascarilla y temperatura 27
Figura 2-2. Cámara Raspberry Pi 28
Figura 3-2. Raspberry Modelo Pi4 B 29
Figura 4-2. Tarjeta Arduino Uno 30
Figura 5-2. Amplificador PAM8403 31
Figura 6-2. Dispositivo NodeMCU. 31
Figura 7-2. Sensor Ultrasónico. 32
Figura 8-2. Sensor de temperatura 33
Figura 9-2. Diseño de la tarjeta de acondicionamiento 34
Figura 10-2. Sensor de temperatura 34
Figura 11-2. Esquema de conexión 35
Figura 12-2. Diagrama eléctrico de conexiones. 36
Figura 13-2. Entorno del sistema operativo Raspbian 37
Figura 14-2. Comandos para instalar Python en Raspbian 37
Figura 15-2. Comandos OpenCV 38
Figura 16-2. Instalación correcta de OpenCV en Python 38

Figura 17-2. Comandos para instalar Tensorflow....39
Figura 18-2. Configuración de la pantalla LCD 40
Figura 19-2. Librerías y funciones importadas para la recolección de imágenes. 42
Figura 20-2. Recolección de imágenes para el data base 42
Figura 21-2. Librerías necesarias para el entrenamiento de la Red Neuronal 44
Figura 22-2. Normalización de imagen 47
Figura 23-2. Generación de nuevas imágenes. 49
Figura 24-2. Patrones significativos de un rostro. 50
Figura 25-2. Representación gráfica del método Flatten. 51
Figura 26-2. Compilación del modelo de red neuronal. 52
Figura 27-2. Respuesta de precisión en la etapa de entrenamiento 53
Figura 28-2. Librerías importadas para el funcionamiento de la red 55
Figura 29-2. Resultados respecto a la detección del rostro y mascarilla 57
Figura 30-2. Imagen adquirida 59
Figura 31-2. Imagen adquirida aplicando la compensación de brillo....60
Figura 32-2. Identificación del sensor de temperatura y el de presencia 62
Figura 33-2. Comunicación con el usuario - Alerta visual 663
Figura 34-2. Representación gráfica del control de acceso....66
Figura 35-2. Circuito de acondicionamiento de señal....66
Figura 36-2. Diseño en SolidWork del Prototipo 67
Figura 1-3. Resultados de la pruebas realizadas en la mañana 70
Figura 2-3. Resultados de pruebas realizadas al medio día....70
Figura 3-3. Resultado de pruebas realizadas a media tarde 71
Figura 4-3. Resultado de pruebas realizadas en la noche....72
Figura 5-3. Datos obtenidos de Python respecto al tiempo de respuesta....79

ÍNDICE DE GRÁFICOS

Gráfico 1-3. Diagrama de flujo para recolección de imágenes 41
Gráfico 2-3. Diagrama de flujo para entrenamiento 43
Gráfico 3-3. Diagrama de flujo para detección de mascarrilla y temperatura 53
Gráfico 4-3. Diagrama de flujo para el procesamiento de imágenes.....58
Gráfico 5-3. Diagrama de flujo para control de temperatura 61
Gráfico 6-3. Número de aciertos de deteccion de mascarrilla.....73
Gráfico 7-3. Número de aciertos promedio del dispositivo 74
Gráfico 8-3. Eficiencia de detección para determinar la distancia adecuada 76
Gráfico 9-3. Consumo de corriente del dispositivo.....80

ÍNDICE DE ANEXOS

ANEXO A: Programación algoritmo de visión artificial

ANEXO B: Protocolo general frente a la exposición del covid-19

RESUMEN

El presente trabajo describe el diseño e implementación de un prototipo de detección de mascarilla y medición de temperatura corporal, empleando técnicas de visión artificial, como dispositivo de control para el personal que ingresa a la empresa PAUFIT. Con el algoritmo de aprendizaje realizado en la plataforma TensorFlow y los algoritmos de visión artificial desarrollados sobre la plataforma Python haciendo uso de las herramientas con la librería OpenCV, se desarrolló el algoritmo para el reconocimiento y control del uso de mascarilla, necesario en la pandemia que se ha dado a nivel global, a este se vinculó un protocolo de señales de alerta gestionado por un módulo Arduino Uno que detecta la presencia y toma la temperatura corporal del usuario, se enlazó una señal de control para reproducir un mensaje de voz, el cual le informa a la persona si cumple con los requerimientos de mascarilla y temperatura corporal, gestionando así el ingreso del personal mediante el protocolo de comunicación MQTT. La estructura del prototipo se diseñó en SolidWorks dando un acabado estético y adecuado para el proyecto. Mediante pruebas de posicionamiento de equipo con adquisición de imágenes se verificó una eficiencia del 92% en reconocimiento de mascarilla cuando la persona se ubica a 100 cm con respecto a la estación de sensado. Para la evaluación del prototipo se realizó pruebas en tiempo real en 4 escenarios diferentes, determinando así una eficiencia del dispositivo en un 94.24%. En conclusión, el dispositivo es capaz de cumplir con los objetivos planteados

Palabras clave: <VISIÓN ARTIFICIAL>, <APRENDIZAJE PROFUNDO (DEEP LERNING)>, <PROGRAMACIÓN EN PYTHON>, <ARDUINO (HARDWARE SOFTWARE)>, <TENSORFLOW (SOFTWARE)>.

ABSTRACT

This work describes the design and implementation of a mask detection prototype and body temperature measurement, using artificial vision techniques, as a control device for personnel entering the PAUFIT company. With the learning algorithm performed on the TensorFlow platform and the artificial vision algorithms developed on the Python platform making use of the tools with the OpenCV library, it was developed the algorithm for the recognition and control of the use of face masks, necessary in the pandemic that has been given globally, to this was linked a protocol of alert signals managed by an Arduino Uno module that detects the presence and measures the user's body temperature; additionally, it linked a control signal to play a voice message, which informs the people if they meet the requirements of a mask and body temperature, thus managing admission personnel using the MQTT communication protocol. The structure of the prototype was designed in SolidWorks giving an aesthetic and suitable finish for the project. Through tests of equipment positioning through image acquisition, an efficiency of 92% was verified in mask recognition when the person is 100 cm from the sense station. For the evaluation of the prototype, real-time tests were carried out in 4 different scenarios, thus determining a device efficiency of 94.24%. In conclusion, the device meets the objectives set.

Keywords: <ARTIFICIAL VISION>, <DEEP LEARNING (DEEP LEARNING)>, <PROGRAMMING IN PYTHON>, <ARDUINO (HARDWARE.SOFTWARE)>, <TENSORFLOW (SOFTWARE)>

Firmado electrónicamente por:
NELLY MARGARITA
PADILLA PADILLA

INTRODUCCIÓN

Actualmente los sistemas de visión artificial han tomado mucha importancia en los más recientes avances tecnológicos que existe hoy en día alrededor del mundo, siendo el reconocimiento facial uno de los más fidedignos y los que más se acogen, así como la tarea de reconocimiento de patrones a partir de imágenes digitales capturadas mediante uno o varios sensores estáticos o en movimiento utilizando para ello técnicas enmarcadas en el procesamiento de imágenes.

El desarrollo de la visión artificial, inteligencia artificial y de las técnicas de aprendizaje profundo está ocasionando un fundamental cambio en la sociedad. (Moreno, 2004) En la actualidad la detección de objetos ha tenido un enorme efecto en distintas áreas tales como el área de medicina utilizada para poder detectar una patología a tiempo y poder prevenirla, también se lo utiliza en la robótica, en la programación, video vigilancia reconocimiento de rostros entre otras. Conseguir que un ordenador aprenda a tomar sus propias decisiones implementando visión artificial unida con las redes neuronales es un enorme desafío.

Mediante la utilización de visión artificial se pretende ofrecer a los sistemas robóticos una aproximación a la capacidad del sentido de la vista humana, de forma que sean capaces de conocer por sí solos su ámbito e interactuar con él de acuerdo con la información recogida tal y como lo hace el cerebro humano (Vázquez, 2017, p. 1). Para el desarrollo del dispositivo que detecta el uso de mascarilla se hizo uso de un algoritmo basado en redes neuronales empleando TensorFlow y la biblioteca OpenCV generando una solución automatizada para abordar este problema que no solo maximizará la eficiencia y garantizará el cumplimiento, sino que además salvará vidas.

La temperatura corporal es la capacidad que tiene un sistema que en esta situación es el cuerpo humano, para producir y remover calor, la temperatura del cuerpo es energía cinética, producto del desplazamiento de las partículas que nos conforman a todos los organismos vivos del mundo. Para medir la temperatura del cuerpo, se ha diseñado un artefacto denominado termómetro y debido a su ayuda se tiene la posibilidad de conocer cuál es la proporción de calor o frio de una persona. La temperatura puede ser señal de que algo no anda bien en el cuerpo lo cual puede ser provocado por alguna infección causando daños al organismo, es por ello que como parte del dispositivo se desarrolló un sistema de medición de temperatura corporal, el cual se obtiene en base a un sensor de temperatura (Leyva, 2019, pp. 1-2)

PLANTEAMIENTO DEL PROBLEMA

En marzo de 2020, la Organización Mundial de la Salud declaró pandemia la enfermedad provocada por el nuevo coronavirus SARS-CoV-2, más conocido como COVID-19, el virus se transmite por medio de pequeñas, gotas de saliva que son expulsadas por las personas al estornudar, toser o al hablar, esta enfermedad se contagia de persona a persona cuando están en contacto cercano. Hoy en día precautelar la salud de las personas se ha convertido en una problemática muy urgente en resolver debido a que el virus que causa esta enfermedad puede infectar a una persona cuando se encuentra expuesta a estas pequeñas gotas que son casi imperceptibles, en nuestro país en el mes de marzo del 2020 se debió acatar varias medidas adoptadas por el gobierno para tratar de contener el virus entre ellas se declaró el uso obligatorio de la mascarilla, todo esto para tratar de proteger la salud de toda la población.

El uso de la mascarilla y la medición de temperatura corporal , son dos medidas adoptadas por distintos países entre los cuales se encuentra Ecuador, es por ello que se determinó implementar un prototipo que sea capaz de reconocer si la persona está usando mascarilla, mediante el uso de técnicas de visión artificial que permitirán detectar y controlar el uso adecuado de la mascarilla, así como la medición de temperatura, para ello se llevará a cabo diferentes pruebas las cuales permitirán obtener los resultados deseados.

A partir de que el COVID-19 hizo su aparición en todo el mundo se pudo mirar como la tecnología se convirtió en una herramienta primordial para evitar su extensión. Mientras tanto que realizarlo de forma manual es muy dificultoso, herramientas como la visión e inteligencia artificial facilitan examinar fragmentos inmensos de datos en apenas segundos o de manera directa en tiempo real. En este sentido, recientemente han ido apareciendo tecnologías españolas como la creada por un grupo de investigadores de la Universidad de Barcelona quienes desarrollaron la tecnología LogMask que permite detectar a personas con y sin mascarilla, controlar los aforos siendo capaces de evitar contagios en supermercados. En la actualidad un sistema de inteligencia artificial puede identificar y saber si se incumple alguna de las medidas de seguridad sanitaria.(Fiter, 2020)

JUSTIFICACIÓN DEL PROYECTO

Desde la aparición del covid-19, se ha evidenciado un incremento de contagios debido al no uso de mascarilla, según la secretaria de salud Ximena Abarca informó que se mantiene una tasa de positividad del 24.7% de contagios en el Ecuador, es por ello que mediante la implementación de este prototipo de detección de uso de mascarilla aplicando técnicas de visión artificial, se pretende

controlar que la persona use mascarilla para poder acceder a su lugar de trabajo. Tomando en cuenta el problema generado por el COVID-19 en este momento a nivel mundial, y en el cual el distanciamiento social debe primar, el uso de la tecnología es sumamente importante. Desde el punto de vista de la ingeniería, se propone la creación de dispositivos que posean un comportamiento inteligente como son: el aprendizaje, la capacidad de adaptación a entornos cambiantes, la creatividad, etc.

Para el desarrollo de este prototipo se utilizó un algoritmo de visión artificial basado en redes neuronales, empleado para detectar el uso de mascarilla de las personas que vayan a ingresar a la empresa PAUFIT, así como también se pretende que este dispositivo ayude a tomar la temperatura sin necesidad de que sea una persona quien supervise dichos requerimientos para ingresar a un lugar, tal como se ha venido evidenciando que ocurre en los lugares de afluencia, evitando así la propagación del virus.

Hoy en día en el mercado se encuentran equipos comerciales con sistemas de detección y toma de temperatura, uno de ellos son los que la empresa Hikvision comercializa, pero que por su elevado costo no son accesibles para las pequeñas y medianas empresas, es por ello que se ha propuesto la implementación de este prototipo el cual ayudará a la empresa PAUFIT a controlar la temperatura y el uso de mascarilla para tratar así de evitar contagios dentro de la empresa.

OBJETIVOS

Objetivo General

Implementar un prototipo de detección de mascarilla y medición de temperatura para el personal que ingresa a la empresa PAUFIT, usando técnicas de visión artificial.

Objetivos Específicos

Investigar si existen dispositivos que puedan realizar la detección de la mascarilla y la medición de temperatura.

Definir los requerimientos de hardware y software para desarrollar el prototipo de detección de uso de mascarilla y medición de temperatura por medio de revisión bibliográfica.

Diseñar y desarrollar un protocolo de comunicación adecuado para controlar el acceso mediante la verificación de usuarios que no cumplan con el uso obligatorio de la mascarilla.

Validar el prototipo de detección de mascarilla y medición de temperatura mediante pruebas de funcionalidad.

Alcance

El trabajo de titulación está destinado a desarrollar un modelo basado en visión artificial para la detección de uso de mascarilla y toma de temperatura corporal para el personal que trabaja en la empresa PAUFIT, el mismo que tiene como objetivo importante mantener el control de la utilización conveniente de la mascarilla y la temperatura precautelando de esta forma la salud de todas las personas que laboran en la empresa.

CAPÍTULO I

1. MARCO TEÓRICO

En el presente capitulo se hace una introducción en relación a conceptos que están dirigidos hacia este trabajo, en primer lugar, se definen ciertos conceptos básicos como, inteligencia artificial, visión artificial, redes neuronales, sensores y definiciones como COVID-19, temperatura corporal, entre otros aspectos relevantes.

1.1 Epidemia COVID-19

El COVID-19 se dio a conocer en diciembre de 2019, demostrando que este virus causa el síndrome respiratorio agudo severo (SARS), esta enfermedad infecciosa causada por el coronavirus que se ha descubierto recientemente y que su brote tuvo origen en Wuhan (China). El virus puede causar neumonía, insuficiencia renal y hasta la muerte. En otros casos, algunas personas infectadas no desarrollan ningún síntoma, pero pueden contagiar igualmente al resto de población. (Bupa, 2020)

El virus se propaga primordialmente de persona a persona por medio de las gotas de saliva o las secreciones nasales que se crean una vez que una persona infectada tose o estornuda, por lo cual es fundamental que se tome precauciones al toser y estornuda. Los síntomas más usuales del COVID-19 son la fiebre, tos seca y la fatiga. Otros indicios menos ordinarios que están afectando a ciertos pacientes son los dolores y molestias corporales, la congestión nasal, el dolor de cabeza, el dolor de garganta, la pérdida del gusto o el olfato. Estas señales suelen ser leves y empiezan gradualmente. Algunas de las personas infectadas solo muestran indicios leves. (OMS, 2020)

1.1.1 Medidas para evitar el contagio del Covid-19

La OMS recomienda el uso de mascarilla como parte de una estrategia integral en la lucha contra el COVID-19, así como el distanciamiento social, la toma de temperatura corporal el uso de gel o alcohol antibacterial, dada la presente crisis sanitaria en todo el mundo la utilización de cubre bocas es indispensable para todo el público más aun en lugares en donde sea difícil el distanciamiento físico, como oficinas, en el transporte público, en tiendas o en otros ámbitos confinados o abarrotados.

1.1.1.1 Uso de mascarilla

Según el centro para el control y la prevención de enfermedades las mascarillas deberían cubrir por completo la nariz, la boca, y adaptarse firmemente contra los lados de la cara sin conformar huecos de forma que además se cubra el mentón, además de lavarse las manos o utilizar un antiséptico previo a posicionarse la mascarilla. La mascarilla debe ajustarse contra los lados de su cara con las tiras detrás de las orejas, o atar los lazos detrás de la cabeza, como se puede observar en la figura 1-1. Si se tiene que ordenar la mascarilla una y otra vez, quiere decir que no se adapta a una forma correcta y quizá se deba buscar otro tipo o marca de mascarilla, además que esta le deberá permitir poder respirar de forma fácil. (CDC, 2021)

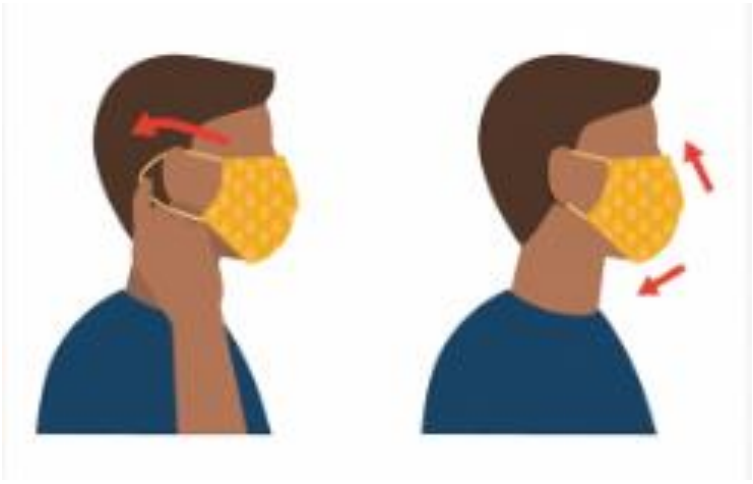

Figura 1-1. Ubicación de cámara parte superior interna del parabrisas
Fuente: (CDC, 2021)

1.2 Dispositivos de detección de Mascarilla

El uso de un algoritmo para la detección de la mascarilla en tiempo real, está elaborado con herramientas de aprendizaje automático, el propósito de la aplicación de este algoritmo es que logre identificar si una persona lleva o no la mascarilla puesta, para ello se utilizó una tecnología conocida como Deep Learning (Aprendizaje Profundo), en términos de procesos primero se enseña al algoritmo a que "aprenda" y logre distinguir si lo que visualiza en primera instancia es una persona y luego si esta lleva o no puesta la mascarilla, Esto hará con imágenes o fotografías de personas con y sin mascarilla, luego de visualizar, observar muchas veces las imágenes el sistema habrá capturado las características de las mismas para poder determinar así si una persona lleva o no mascarilla (Barrios, 2020)

Para el estado de emergencia que se está viviendo, se han venido creando dispositivos que ayuden a mitigar en cierta manera la propagación de este virus, dentro de la investigación se encontró algunos trabajos relacionados con Machine Learning y TensorFlow que trabaja con CNN (Convolutional Neural Networks) para efectuar la detección de objetos, los elementos

primordiales de una CNN son las capas convolucionales, las capas de pooling y las capas plenamente conectadas , como se puede observar en la figura 2-1

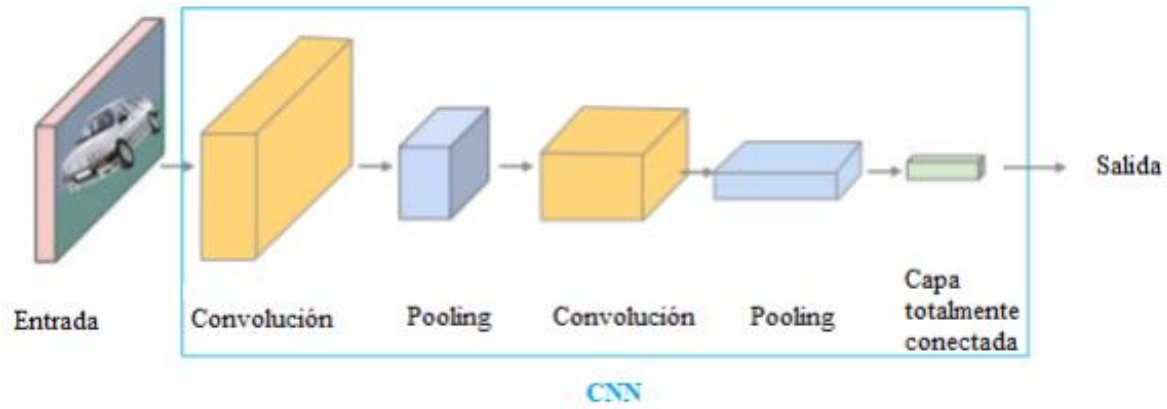

Figura 1-1. Representación gráfica de una red convolucional

Fuente: (Hugo, y otros, 2021)

Dentro de los trabajos que formaron parte de la revisión bibliográfica se puede destacar el realizado por Goyal, Agarwl y Kumar, siendo este uno de los referentes en la detección de rostros, en este trabajo se menciona que el reconocimiento facial implica tres pasos básicos que incluyen la detección de rostros ,reconocimiento de rostros y extracción de rostros , cualquier sistema necesita encapsular la imagen y luego administrarla y registrar las características vitales para determinar la ubicación de la cara (Kruty, 2017), para ello hizo uso de un clasificador Haar en una Raspberry Pi; la función de este clasificador en cascada es entrenar a través de diversas muestras positivas de un objeto en especial, así como imágenes negativas arbitrarias del mismo tamaño. Una vez entrenado el clasificador, este se puede aplicar a una región de una imagen e identificar el objeto en cuestión, la figura 3-1 muestra un ejemplo del clasificador Haar, además también hizo uso de la librería OpenCV con una resolución de 1080 pixeles y 30 cuadros por segundo.

Figura 2-1. Ejemplo del clasificar en cascada Haar

Fuente: (Maliha, 2019)

Otro trabajo relacionado es de Khan, Chakraborty, Astya, y Khepra (Maliha, 2019, p. 117) que hace detección de rostros por medio de un sistema que recibe datos de acceso por medio de una cámara y que trabaja en tiempo real con el algoritmo PCA, el cual hace posible minimizar la proporción de datos de procesamiento al enfocarse solamente en los elementos primordiales. Existen varias investigaciones donde se realiza la detección de mascarillas usando las herramientas mencionadas. Una ejemplificación es aquella utilizada para la detección de mascarillas desarrollado por Ansor, Ritzkal y Afrianto, que usa TensorFlow y una CNN pre-entrenada, en el cual analizan las métricas de la red neuronal construida para el sistema tomando en cuenta la precisión, la exactitud y la exhaustividad.

Otra investigación importante es la de Cakiroglu, Ozer y Gunsel (Ozan, 2019), el cual entrena un detector de objetos para la detección de mascarillas usando TensorFlow con una CNN basada en zonas y que analizan los resultados del rendimiento de la misma. El primordial aporte de esta investigación es el entrenamiento con una pequeña porción de datos, ya que en general se dispone de 2695 imágenes, realizando la detección de mascarillas en un clip de videos utilizando Deep. Learning.

.

1.2.1 Sistemas comerciales para detección de mascarilla y toma de temperatura

En la actualidad, empresas como Automatic Systems, Hikvision, By Demes Group han implementado dispositivos capaces de detectar el uso de mascarilla y medir la temperatura corporal basada en cámaras y sensores con el fin de controlar estas medidas de bioseguridad para tratar de controlar la propagación del COVID-19, a continuación, se muestran ejemplos de dispositivos para detección de mascarilla y medición de temperatura existentes en el mercado.

La empresa Automatic Systems, utiliza el sistema SafeFlow, una solución patentada y fabricada en Europa, que ofrece herramientas indispensables para mejorar y automatizar los procedimientos de detección de presencia de mascarilla y de temperatura. Para la toma de temperatura el dispositivo posee un sensor térmico integrado basado en tecnologías de infrarrojos que funciona con un procesador de alto rendimiento, su alcance de detección comienza a 50 centímetros y una precisión de 0,5 °C para la temperatura.

Para la detección de presencia de mascarilla lo realiza con la cámara de precisión que funciona con el procesador de alto rendimiento, permite llevar a cabo un análisis rápido de la presencia o ausencia de una mascarilla además el dispositivo cuenta con un pantalla LCD a color de ocho pulgadas, adicional a esto el dispositivo cuenta con un control de aforo, en la pantalla se visualiza

el número de personas que aún pueden acceder e indica el momento que llegue a su capacidad máxima. (BORRMART, SA, 2020)

Como sistema de alerta al cumplir las condiciones para poder ingresar se emitirá una señal acústica y luminosa. En la figura 4-1 se representa el diseño del dispositivo creado por Automatic System.

Figura 3-1. Dispositivo de Automatic System
Fuente: (BORRMART, SA, 2020)

Hikvision, una empresa proveedora de soluciones para la industria de seguridad desarrolló un prototipo de detección de temperatura y mascarilla que permite una rápida detección de temperatura en personas sin necesidad de contacto, así como también tiene la posibilidad de detección de uso de mascarilla. El dispositivo de la marca Hikvision tiene una resolución de 120 × 160, un lente termográfico de 3mm su alcance de detección empieza desde los 30 centímetros hasta los 2 metros, posee una pantalla de 10,1 pulgadas con un rango de medición de temperatura de 30 a 45 grados centígrados y una presión de ± 0.5 ℃ (Campos, 2020)

Para el sistema de alerta de este dispositivo lo hace a través de una notificación de audio. La figura 5-1 representa los dispositivos desarrollados por la empresa Hikvision.

Figura 4-1. Dispositivos desarrollados por Hikvision

Fuente: (Seguridad, 2020)

By Demes Group, ofrece un dispositivo de detección de temperatura y verificación de mascarilla, los fabricantes son Hyundai, Dahua y Zkteco, este dispositivo posee una serie de características como una pantalla táctil con doble cámara, un sensor de medición de temperatura corporal con una distancia que comienza en los 30 centímetros hasta los 2 metros. Además, posee una precisión de lectura de temperatura de ± 0.3 °C a ± 0.5 °C (Seguridad, 2020)

El dispositivo cuenta con un sistema de alarmas las cuales se activan cuando la temperatura corporal sobrepasa los límites permitidos. También emite una alarma al notar ausencia de la mascarilla en una persona, dando la posibilidad de bloquear el acceso a quien no cumpla con estos requisitos. La figura 6-1 representa el dispositivo creado por la empresa By Demes Group.

Figura 5-1. Dispositivo de detección implementado por By Demes Group

Fuente: (Seguridad, 2020)

1.3 Inteligencia Artificial

El desarrollo tecnológico que se ha producido en las últimas décadas en el campo de la ingeniería informática ha dado paso a un sin número de novedosas ramas científicas y campos de exploración. La tecnología y en particular los ordenadores permanecen cada vez más unidos e incluidos en nuestra vida diaria. Hoy en día los sistemas inteligentes son capaces de comprender y procesar órdenes utilizando un lenguaje natural. Definiendo así a la inteligencia artificial como la inteligencia que poseen los ordenadores, un agente inteligente percibe el entorno que le rodea y actúa de acuerdo a un escenario desconocido. Una maquina tan flexible que es capaz de realizar tareas "cognitivas", tales como el aprendizaje con el fin de resolver diferentes problemas. Hoy en día se han realizado muchos avances en esta área científica de aprendizaje, planificación, razonamiento, comunicación, percepción y manipulación. (Andreea, 2016)

Así como en el ser humano la manera en cómo recibe los datos es a través de su visón, un sistema inteligente recibe la información a través de una cámara compuesta por un sin número de lentes y sensores específicos para cada sistema y campo de aplicación. Una vez efectuado todos los ajustes mecánicos, la visión artificial posibilita al sistema identificar la mayor parte de los objetos para determinar cómo reaccionar y actuar ante ellos. Este proceso está dotado de una inmensa dificultad, debido a la infinidad de situaciones u objetos a los que el sistema se puede enfrentar.

1.3.1 Visión Artifcial

La visión artificial abarca todo proceso óptico en el cual un sistema inteligente es capaz de adquirir información de determinado medio para su interpretación mediante el uso de un ordenador. Al inicio los sistemas de visión artificial estaban estrechamente basados con la visión humana, sin embargo, debido a la falta de conocimiento de los procesos que tienen lugar en el cerebro a la hora de interpretar los datos del sistema visual, dichos sistemas artificiales resultaron imprácticos. La visión artificial también se la puede definir como un campo de la inteligencia artificial que por medio de la utilización de técnicas adecuadas permite la obtención, procesamiento y análisis de la información adquirida por medio de imágenes digitales. Está formada por un conjunto de distintos procesos encaminados a realizar el análisis de imágenes, estos procesos son captación de imágenes, memorización de la información, proceso e interpretación de resultados (Gonzales, 2014)

El cerebro del ser humano, en conjunto con el sistema visual del que dispone nuestro cuerpo, posibilita un reconocimiento y una interpretación del ambiente que lo rodea, siendo este mucho más flexible y adaptable a cambios, más que cualquier sistema de visión artificial. Sin embargo, los sistemas de visión artificial son capaces de procesar porciones de datos mucho más grandes y en menos tiempo si se trata de trabajos repetitivos. La figura 7-1 muestra una representación del sistema de visión por computadora. La infalibilidad de la exactitud matemática posibilita un análisis y estudio mucho más descriptivo y amplio por parte de un sistema basado en visión artificial. Esta característica posibilita la construcción de un sistema bastante diferente en cuanto a elementos y habilidades, según el trabajo que se va a realizar.

El sistema debe ser capaz de procesar la información obtenida y ejercer los algoritmos adecuados para su procesamiento e idónea interpretación. Pero esto resultaría imposible sin un detallado pre-procesado de la imagen, debido a que los patrones a reconocer pasan a ser infinitos. El sistema es capaz de detectar al objeto independientemente de su iluminación, color o postura dentro de la imagen y por lo tanto conoce cada una de las probables variaciones que el objeto puede tener.

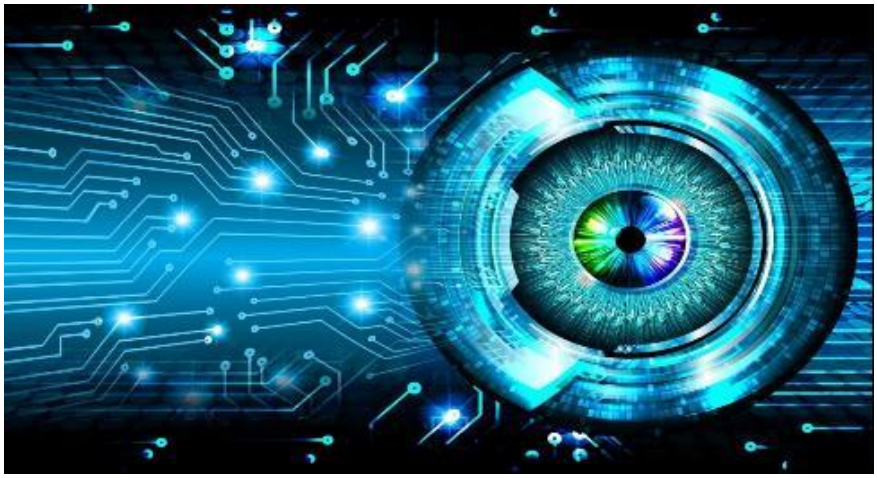

Figura 6-1. Representación de un sistema de visión por computadora

Fuente: https://iaarbook.github.io/img/computer-vision.jpg

1.3.1.1 Reconocimiento de objetos.

El reconocimiento de objetos en imágenes tomados por un sistema de visión tiene siempre la misma configuración al momento de trabajar.

- Adquisición de imagen, se hace por medio de cualquier dispositivo capaz de tomar imágenes como, por ejemplo, una cámara fotográfica, con una imagen basta para iniciar el proceso, el tiempo de adquisición es bastante bajo y no requiere de supervisión siempre y cuando las imágenes cumplan con un mínimo de calidad

- Digitalización, se refiere a la forma de como una imagen se puede convertir en un idioma compresible para los ordenadores es decir una conversión analógico-digital.

- Extracción de características, lo cual consiste en la búsqueda de patrones en una imagen, ya sean de tipo geométrico, estadístico, topológico, etc.

- Sistema de clasificación, este sistema trata de asignar a cada clase de objeto que se busca reconocer, un patrón o grupo de patrones peculiares, de forma que se pueda asociarlos y de esta forma clasificar a cada uno de ellos.

De esta forma, cada vez que se disponga de una imagen nueva a clasificar, se obtiene sus detalles primordiales para ser comparadas con las que ya se habían estudiado anteriormente para cada clase de objeto. En base de la similitud encontrada, la imagen se asocia a una clase u otra, explicando de esta forma la posibilidad que tendría de pertenecer a esta clase. Definiendo así que ha mas similitud, más posibilidad se recibe. Una vez que se asocian patrones de propiedades a cada objeto, es habitual que se defina un vector de propiedades, como se muestra en la ecuación 1, definiendo de esta forma un patrón como el grupo de propiedades de una imagen (Vázquez, 2016, p. 5).

$$x = [x_1, x_2, \dots, x_n]^T \qquad \text{Ec.1}$$

Donde x representa el patrón que se estudia mientras que x_1 son las características que lo compone.

1.3.2 Adquisición de Imágenes

En visión artificial la adquisición de imágenes juega un papel importante al instante de obtener buenos resultados del proceso de reconocimiento. Es decir, se trata de lograr que la imagen sea lo más idónea posible para poder seguir con el proceso, por medio de implementos tecnológicos como cámaras y teléfonos.

1.3.2.1 La cámara

Dispositivo encargado de cambiar las señales luminosas que aparecen en la escena, en señales analógicas capaces de ser transmitidas. Su funcionalidad es capturar la imagen proyectada en el sensor para lograr transferirla a un sistema electrónico. Las cámaras usadas en visión artificial necesitan de una secuencia de propiedades que permitan el control del disparo para capturar objetos que llamen la atención del hombre. Siendo estas más sofisticadas que las cámaras convencionales, debido a que deben ser capaces de poder realizar un control completo de tiempos, señales, rapidez de capturación, sensibilidad, entre otras. En la figura 8-1 se muestran ejemplos de cámaras con diferentes lentes e iluminación usadas para sistemas de visión artificial.

Figura 7-1. Cámaras para aplicaciones de visión artificial

Fuente: https://iaarbook.github.io/img/computer-vision.jpg

Entre las cámaras más reconocidas para la adquisición de imágenes se puede nombrar las siguientes:

- **Cámara web,** es un dispositivo pequeño que se conecta al puerto USB del ordenador y de esta forma permite captar clip de videos y tomar fotos digitales con resolución baja, ofreciendo una mala calidad de gráficos.(Jiménez, 2015)
- **Cámaras digitales,** es una cámara de fotos que posee un sensor electrónico para digitalizar las imágenes y guardarlas en una menoría, poseen una buena resolución, brindando una excelente calidad de imágenes.
- **Kinet,** es un dispositivo desarrollado para mejorar la experiencia de los usuarios de videojuegos, es una apuesta tecnológica basada en un sistema que posibilita la identificación en 3D a través de una cámara RGBD, de infrarrojos, sensores de profundidad, un micrófono y un procesador personalizado. (Page, 2019)

La imagen digital obtenida se mide en pixeles, esta es la unidad de medida del tamaño de la imagen, cada pixel está formado de una conjunción de los 3 colores primordiales: rojo, verde, azul, cada pixel tiene una localización matricial en el espacio de la imagen

1.3.3 Pre procesamiento

Toda imagen que se consigue por medios ópticos, electroópticos o electrónicos sufre en cierta medida los efectos de la degradación que se expresan a modo de sonido, pérdida de definición y de fidelidad de la imagen. La degradación viene provocada por el sonido de los sensores de captura, imprecisiones en el enfoque de la cámara, desplazamiento de la misma o perturbaciones aleatorias, teniendo más relevancia el impacto de la propagación de la radiación en el medio de transmisión. Los mecanismos que intentan contrarrestar dichos efectos se integran en la fase de pre procesado, tomando el nombre de operaciones de restablecimiento. Los algoritmos de pre procesado permiten cambiar la imagen para excluir ruido, transformarla geométricamente, mejorar la magnitud o el contraste. (Alejandro, 2019)

En el pre-procesado se aplican 4 etapas para normalizar y alinear la imagen:

- Rotación. Una de las utilidades de calcular las coordenadas de los ojos, radica en poder decidir el ángulo de giro de una cara en una imagen y compensarlo.
- Escalado. Para lograr que cada una de las imágenes tenga el mismo tamaño, se usa la distancia entre los centros de los ojos para lograr un radio por el que la imagen debería ser aumentada o limitada.
- Recorte. Una vez la imagen fue rotada y escalada, se hace el recorte de la misma para obtener solamente la zona de interés. Para conceptualizar la zona se usa la coordenada del ojo derecho. Hay diversos tamaños de imagen estandarizados por los cuales se puede sustraer la zona de interés relativa a las necesidades del sistema.

- Normalización. Las imágenes tienen la posibilidad de exponer variabilidad en la luminosidad y en el contraste, lo cual genera que imágenes semejantes sean bastante diferentes respecto al valor de magnitud de sus píxeles. Por medio de la normalización, se pretende que las imágenes que poseen la mayoría de sus valores de magnitud concentrados en una región limitada pasen a extenderse por todo el rango de valores. Esto resulta en imágenes con más contraste y con menor variabilidad lumínica.

En un proceso de visión artificial dichos algoritmos deben ser usados lo menos posible, un uso desmesurado de ellos afectará el tiempo de proceso total e indicará que la calibración, iluminación y selección de los recursos de la fase de obtención no ha sido la correcta.

1.4 Detección de imágenes

El cerebro es un procesador de información con características relevantes, es capaz de procesar con gran rapidez mucha información que procede de los sentidos, combinarla o compararla con la información almacenada en el cerebro y ofrecer respuestas adecuadas inclusive en situaciones novedosas. Consiguiendo así discernir un murmullo en una sala ruidosa, diferenciar una cara en una calle mal iluminada o leer entre líneas en una declaración política, pero lo más increíble de todo es su capacidad de aprender a representar la información esencial para desarrollar estas tareas sin normas explicitas.

Aunque hoy en día todavía se ignora sobre la manera en que el cerebro aprende a procesar la información, se han desarrollado modelos que tratan de mimetizar tales capacidades, llamados redes neuronales artificiales o modelos de computación convencionales. La elaboración de dichos modelos implica la deducción de los aspectos o características fundamentales de las neuronas y sus conexiones. (Palacios, 2018)

1.4.1 Deep Learning

El Deep Learning es una técnica de Machine Learning que instruye a los ordenadores a realizar lo que resulta natural para los individuos, aprender por medio de ejemplos. Las redes neuronales fundamentadas en Deep Learning son potentes modelos que logran un elevado rendimiento en procesos que resultan difíciles, como por ejemplo el reconocimiento de patrones. Por medio del Deep Learning, un modelo informático es capaz de aprender a realizar labores de clasificación desde imágenes, texto o sonido. Alcanzando unos niveles de exactitud de reconocimiento jamás observados. Debido a que la mayoría de los procedimientos de aprendizaje emplean arquitectura de redes neuronales constantemente los modelos de Deep Learning se llaman redes neuronales

profundas, el concepto profundo suele hacer alusión al número de capas escondidas en la red neuronal. (Palacios, 2018)

1.4.2 Algoritmos

En la actualidad existe gran variedad de algoritmos y procedimientos para el reconocimiento de imágenes, uno de ellos está basado en redes neuronales, el cual consiste en el entrenamiento de una red neuronal a través de diversas imágenes incluyendo cambios en la iluminación, gestos, etc. Uno de los beneficios al utilizar este algoritmo es su buena precisión, aunque para ello es necesario de un entrenamiento previo.

1.4.2.1 Redes Neuronales

A medida que los procesadores modernos se tornan cada vez más poderosos los científicos continúan siendo desafiados a utilizar máquinas de una manera más acertada para labores que son subjetivamente básicas para los humanos. Las redes neuronales son de interés para los estudiosos en muchos sectores por diferentes causas, por ejemplo, los Ingenieros Eléctricos hallan varias aplicaciones en el procesamiento de señales y la teoría de control, así mismo los Ingenieros Informáticos se encuentran asombrados por el potencial que el hardware posee para llevar a cabo un proceso de redes neuronales de forma eficiente. Las redes neuronales procesan la información de manera semejante a como lo hace el cerebro humano. Una red neuronal está formada de un grandioso número de recursos de procesamiento altamente interconectados que trabajan en paralelo para solucionar un problema específico. En la figura 9-1 se puede observar una representación de la red neuronal. Las redes neuronales, como los individuos, aprenden con el ejemplo (Palacios, 2018)

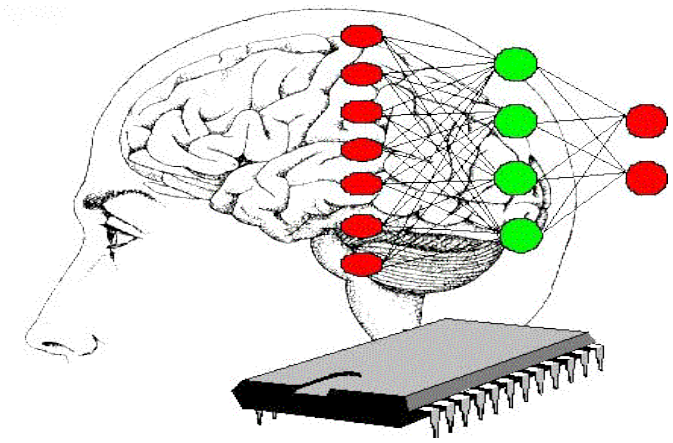

Figura 8-1. Representación de una Red Neuronal.

Fuente: https://n9.cl/st2jas

1.4.2.2 Redes Neuronales Convolucionales

Las redes neuronales convolucionales son una de las redes neuronales profundas más conocidas, consideradas un caso particular de las redes neuronales de aprendizaje profundo, capaces de aprender en los diferentes niveles de abstracción. Su uso se ha incrementado en las últimas décadas y se emplean principalmente para la identificación y clasificación de características en imágenes, sonidos y videos

Estas redes se caracterizan por tener una gran precisión ya que los modelos generados son utilizados constantemente para reconocimiento y procesamiento de imágenes, las redes convolucionales contienen varias capas ocultas, donde las primeras puedan detectar líneas, curvas y así se van especializando hasta poder reconocer modalidades complejas como un rostro, siluetas, etc. La figura 10-1 muestra la estructura de una red convolucional. (Rios, 2020)

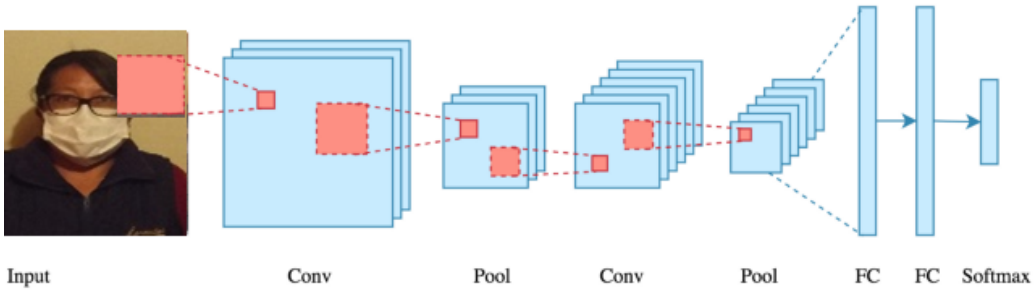

Figura 9-1. Estructura de una Red Convolucional

Fuente: (Bootcamp, 2019)

Generalmente, las redes neuronales convolucionales están construidas con una distribución que contendrá 3 tipos diversos de capas:

- Una capa convolucional, La operación de convolución obtiene como acceso o input la imagen y después aplica sobre ella un filtro o kernel que nos regresa un mapa de las propiedades de la imagen original, logrando minimizar la medida de los parámetros.

- Una capa de reducción o de pooling, la cual va a minimizar la proporción de límites al quedarse con las propiedades más frecuentes.

- Una capa clasificadora plenamente conectada, la cual nos va a ofrecer el resultado final de la red.

1.4.3 TensorFlow

TensorFlow es una librería de software de código abierto lanzada en 2015 por Google para facilitar a los desarrolladores diseñar, edificar y capacitar modelos de Deep Learning. TensorFlow

se originó como una librería interna que los desarrolladores de Google usaron para edificar modelos en la organización, y se espera que se añadan funcionalidades extras a la versión de código abierto. Aun cuando TensorFlow es solo una de algunas posibilidades accesibles para los desarrolladores, se optan por utilizarla gracias a su diseño inteligente y facilidad de uso. (Palacios, 2018)

TensorFlow, como se observa en la figura 11-1 los nodos representan operaciones matemáticas, mientras que los bordes del grafico representan los conjuntos de datos multidimensionales (tensores) comunicados entre ellos. La arquitectura flexible le permite implementar cálculos en una o más GPU (Unidad de Procesamiento Gráfico), la cual se encarga de procesar todos los gráficos que usa un sistema de cómputo y así acelerar el trabajo del procesador, en una computadora de escritorio, servidor o dispositivo móvil con solo una API. (Application Programming Interfaces).

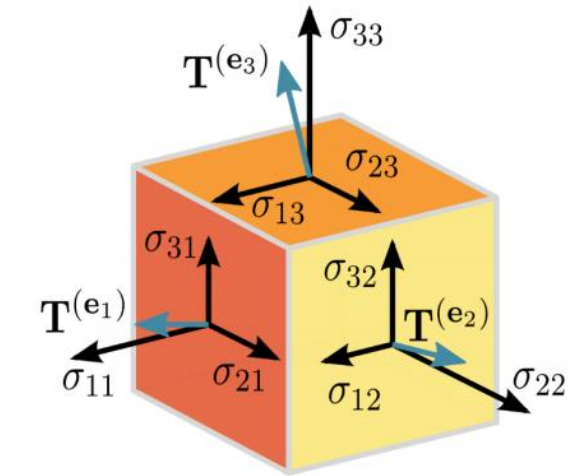

Figura 10-1. Representación de tensores

Fuente: (Palacios, 2018)

1.4.3.1 Arquitectura MobilenetV2

Esta arquitectura MobilenetV2 es parte de la biblioteca de TensorFlow, para poder entender esta arquitectura primero se describe lo que realiza la primera versión de MobileNetV1, la gran idea detrás de esta versión es que las capas convolucionales pueden ser sustituidas por las llamadas capas de convoluciones separables en profundidad. Formadas por una convolución profunda, que filtra la entrada y después se aplica una convolución con un kernel de tamaño 1x1. La figura 12-1 muestra los bloques convolucionales de la red MobilenetV1 y de la V2. (Tsang, 2019)

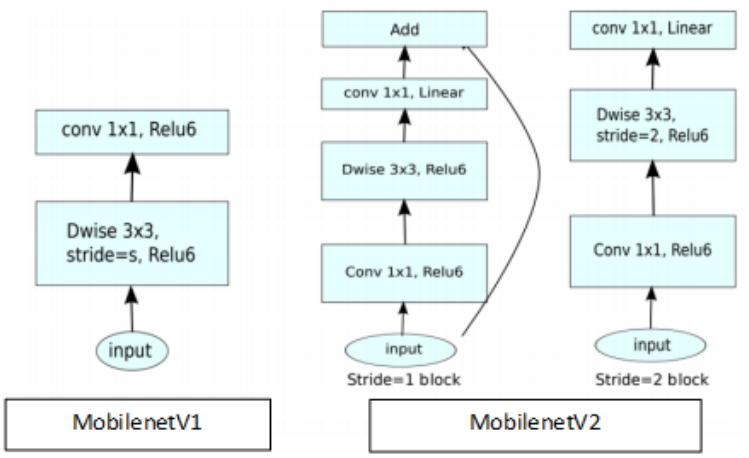

Figura 11-1. Bloques convolucionales de la red Mobilenet V1 y V2

Fuente: (Tsang, 2019)

En cambio, la versión V2 de la red ahora tiene un bloque compuesto por tres convoluciones en serie, donde la convolución puntual al final del bloque hace justo lo contrario, reduce el número de canales conocida como capa de cuello de botella porque reduce la cantidad de datos que fluyen a través de la red, mientras que la capa interna encapsula la capacidad del modelo para transformarse de conceptos de nivel inferior, como píxeles, a descriptores de nivel superior, como categorías de imágenes. Por último, al igual que con las conexiones residuales tradicionales, los atajos permiten un entrenamiento más rápido, eficiente y con una mayor precisión logrando un rendimiento de vanguardia para la detección de objetos y segmentación.

1.5 Temperatura Corporal

Según el personal de Healthwise, la temperatura del cuerpo es un indicador de la capacidad del organismo de crear y remover calor. El cuerpo humano es bastante eficiente para conservar su temperatura en límites seguros, inclusive una vez que la temperatura exterior cambia mucho. Cuando existe mucho calor, los vasos sanguíneos en la dermis se dilatan para mover el exceso de calor al área de la dermis. La temperatura del cuerpo se puede medir en varios sitios del cuerpo. Una alta temperatura del cuerpo o fiebre, es una de las maneras en que el sistema inmunológico aspira combatir una infección, no obstante, en ocasiones la temperatura puede ser bastante elevado, en aquel caso, la fiebre podría ser grave y llevar a complicaciones, Los termómetros indican la temperatura del cuerpo bien en grados Fahrenheit (°F) o en grados Celsius (°C) (Healthwise, 2020)

Según la organización mundial de la salud la temperatura corporal se determina como se muestra en la tabla 1-2.

Tabla 1-2: Rangos de temperatura Corporal

Clasificación	Temperatura
Temperatura Normal	35.5 - 37.5°C
Hipotermia	Menor a 35°C
Febrícula	37.8 - 38 °C
Fiebre	Mayor a 38 °C

Realizado por: Paullán, A, 2021

Fuente: (ELSEVIER, 2017)

1.5.1 Adelantos tecnológicos en la medición de temperatura corporal

La temperatura del cuerpo se puede medir de diferentes maneras, convencionalmente, la temperatura del cuerpo se ha medido usando termómetros de contacto que se colocan en la frente, en la boca, la oreja, la axila o el recto, hoy en día la manera de medir temperatura ha ido cambiando, la tecnología en la actualidad da muchas ventajas para los elaboradores de termómetros, los proveedores de instrumentación continúan desarrollando nuevos productos, perfeccionando todavía más la exactitud, fiabilidad y facilidad de uso de estos dispositivos en ámbitos de producción exigentes. Por esto, se ha usado exitosamente la tecnología infrarroja (IR) para medir y mantener el control de la temperatura. (Schneider, 2007)

1.6 Sensores y Transductores

Un circuito electrónico debería ser capaz de comunicarse con el planeta real para ajustarse al entorno, esto no podría ser viable sin la existencia de los llamados sensores y actuadores. Los sensores tienen la posibilidad de ser utilizados para medir de diferentes maneras un variado rango de energía como desplazamiento, señales eléctricas, radiación térmica o magnética, etcétera. Los Actuadores tienen la posibilidad de ser utilizados para interrumpir voltajes o corrientes, hay una variedad de dispositivos que tienen la posibilidad de ser análogos o digitales, el tipo de ingreso o salida del transductor es dependiente del tipo de señal que se encuentre procesando, sea "sensada" o "controlada" sin embargo puede definirse un sensor y actuador como dispositivos que transforman una variable física en otra. Los dispositivos que dan la capacidad de acceso son habitualmente denominados sensores, estos "sensan" un acontecimiento en el planeta físico y tiene como contestación una excitación que fuerza a cambiar dicha variable física en una señal eléctrica. Los dispositivos que tienen la posibilidad de dar salida son principalmente denominados

actuadores y son utilizados para el control de cualquier dispositivo externo, como desplazamiento o ruido (Vallejo, 2015)

1.6.1 Tipos de sensores

Existen diversos tipos de sensores los cuales son usados para convertir la información que llega del exterior en un impulso eléctrico, clasificándolos así de acuerdo al tipo de variable que tenga que medir o detectar, en función de esto a continuación se procede a describir algunos de los sensores que existen. (Logrono, 2016)

- **Sensor de distancia**

Son dispositivos que permiten medir distancias; además, dependiendo del tipo, tienen la posibilidad de utilizarse como sensores de presencia o desplazamiento. Un ejemplo de sensor de distancia es el infrarrojo el cual se muestra en la figura 13-1 su desempeño se basa en un sistema de emisión y recepción de radiación, otro ejemplo de sensor de distancia, es el sensor ultrasónico, su desempeño se basa en la emisión de sonidos con frecuencias menores a las que puede notar el oído humano, permitiendo identificar presencia o medir distancias, por medio de la medición del tiempo transcurrido, entre la emisión de la señal, y la recepción del eco que corresponde. La señal que se emite podría ser de tipo pulso, onda, etc.

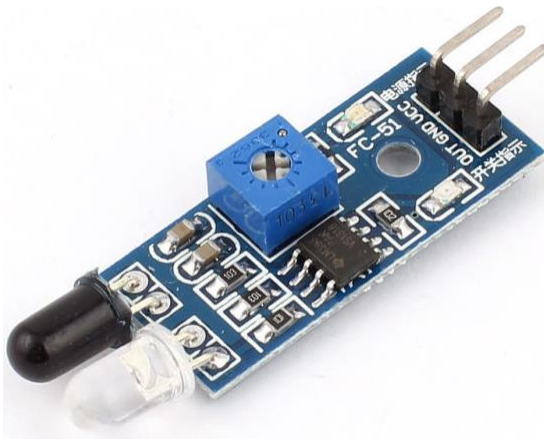

Figura 12-1. Sensor de distancia infrarrojo

Fuente: https://sandorobotics.com/wp-content/uploads/2017/06/58.jpg

- **Sensor de temperatura**

Son elementos eléctricos y electrónicos que, en calidad de sensores, permiten medir la temperatura por medio de una señal eléctrica definida. Esa señal puede enviarse de manera directa o por medio del cambio de la resistencia. Además, se llaman sensores de calor o termo sensores. Un sensor de temperatura se utiliza, entre otras aplicaciones, para controlar

circuitos. Los sensores de temperatura además se denominan sensores de calor, detectores de calor o sondas térmicas. La figura 14-1 muestra el sensor de temperatura LM35.

Figura 13-1. Sensor de temperatura

Fuente: http://www.prometec.net/wp-content/uploads/2015/08/LM35DZ.png

Sensores de Proximidad

Este tipo de sensores se aplican para identificar la presencia de objetos cercanos sin que exista algún tipo de contacto físico. Estos sensores tienen muchas aplicaciones como por ejemplo para sistemas de advertencia y dispositivos móviles. Los sensores de proximidad usan una secuencia de procedimientos de detección físicos que integran el acoplamiento capacitivo, captador inductivo, infrarrojo, foto detección de luz ambiental. La figura 15-1 muestra una imagen de un sensor inductivo.

Figura 14-1. Sensor de proximidad inductivo

Fuente: https://images-na.ssl-images-amazon.com/images/I/51vWq3aLkJL._SX342_.jpg

1.7 Python

Python es el lenguaje de programación recomendado por los fundadores de la Raspberry Pi, debido que es un lenguaje de programación de elevado grado, un lenguaje con una sintaxis fácil,

ideal para usarse en los temas educativos, ya que es un lenguaje de programación sencillo por lo que Raspberry Pi lo utiliza para aprender a programar. Python es un lenguaje de programación creado a fines de los ochenta por Guido Van Rossum, aun cuando su fama bien podría decirse que fue algo reciente. (Gama, 2020) La figura 16-1 muestra el logotipo del software Python.

Figura 15-1. Logo de Python.

Fuente: http://1000marcas.net/wp-content/uploads/2020/11/Python-logo.jpg

1.8 Raspberry

Según (Lucas, 2019), en su portal web la Raspberry Pi es un mini ordenador completo que dispone de una placa de reducidas magnitudes, del tamaño de una tarjeta de crédito. Tiene cada una de las conexiones usuales de un ordenador estándar: puertos USB, acceso a internet y de red, salidas de audio, clip de video, etc., además cuenta con los periféricos necesarios como ratón, teclado y debido a su salida de clip de video HDMI, tiene la posibilidad de conectarse a un televisor o un monitor y utilizarlo como un ordenador de sobremesa o como un centro multimedia. Este microordenador fue creado en el Reino Unido por Raspberry Pi Foundaion, que quería lanzar un ordenador sencillo y de bajo coste para llevar los conocimientos de informática a todos los colegios de todo el mundo, gracias a su versatilidad y bajo costo. El propósito inicial ha sido superado y se empezó a utilizar en otros espacios como la robótica y la domótica. El modelo Raspberry Pi 4 B apareció en el mercado en junio de 2019. Por primera ocasión, se integran puertos USB 3.0. Además, incluye un procesador Broadcom nuevo más potente que los anteriores y es capaz de conectar 2 pantallas 4k. La Figura 17-1 es la representación física de la Raspberry modelo Pi 4 B.

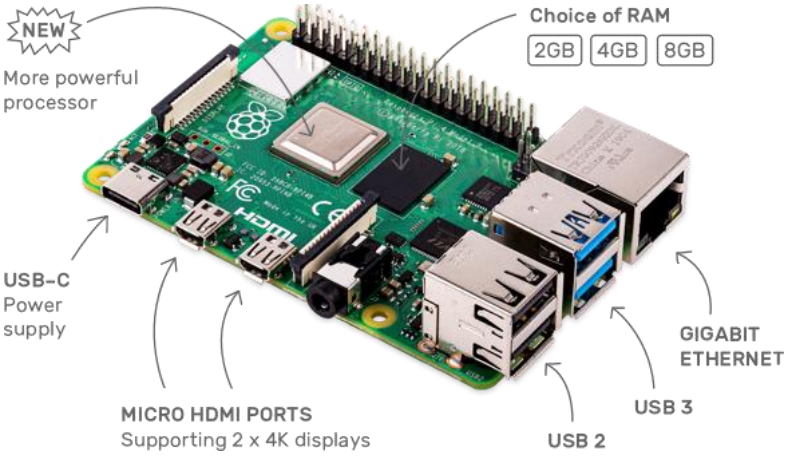

Figura 16-1. Raspberry Pi **4**

Fuente: (Raspberry PI, 2019)

1.9 Arduino

Según (Fernández, 2020) Arduino es una plataforma para usarlos en proyectos de electrónica de código abierto, está basada en hardware y software independiente, flexible y simple de usar para los creadores y desarrolladores. Esta plataforma posibilita generar diversos tipos de microordenadores de una sola placa a los que la sociedad de creadores puede darles diversos tipos de uso, debido a su vivencia de cliente fácil y accesible. Arduino se ha usado en una gran cantidad de proyectos y aplicaciones diferentes. El programa Arduino es simple de utilizar para principiantes, pero lo suficientemente flexible para usuarios avanzados. Funciona en Mac, Windows y Linux. Los docentes y los alumnos lo usan para edificar artefactos científicos de bajo precio, para probar los inicios de la química y la física, o para iniciar con la programación y la robótica.

La figura 18-1 representa la tarjeta Arduino Uno con su distribución de terminales.

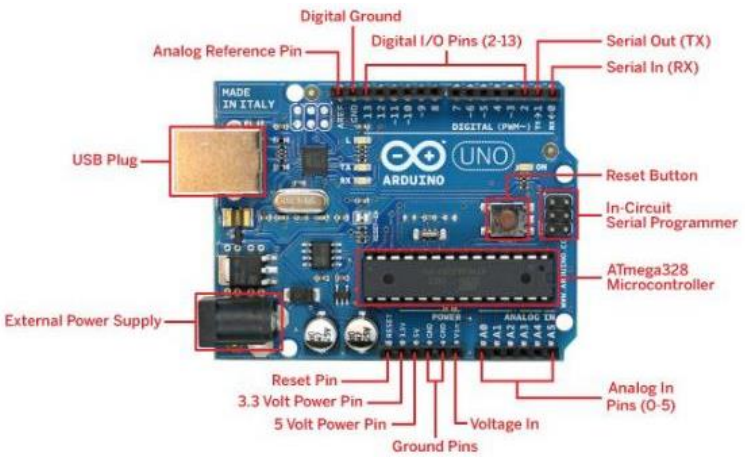

Figura 17-1. Arduino UNO distribución de terminales

Fuente: https://www.robomart.com/image/catalog/RM0058/02.jpg

1.10 NodeMCU EP8266

Según (Xukyo, 2020), el NodeMCU ESP8266 es un microcontrolador con un módulo Wifi incluido. Es bastante simple de utilizar, es ligero y tiene una memoria y capacidad de cálculo más grande que el Arduino Uno. Lo primordial de la IoT es conectar los objetos a una red como el WiFi. El transmisor y la antena Wifi incluidos en el microcontrolador permiten el acceso a internet. Debido a esto, es viable producir un servidor que alberga una página web que posibilita mantener el control del microcontrolador de manera remota. Esta página podría ser usada para demostrar los valores medidos por el NodeMCU o para el control de las entradas y salidas del microcontrolador. La figura 19-1 muestra una imagen del dispositivo NodeMCU.

Figura 18-1. NodeMCU ESP8266

Fuente: https://www.aranacorp.com/wp-content/uploads/nodemcu-v3-esp8266.jpg

CAPÍTULO II

2. METODOLOGÍA

Este capítulo se enfoca en la implementación del prototipo determinando los requerimientos a cumplirse, así como la concepción general del sistema que permitirá la implementación del prototipo de detección de mascarilla usando visión artificial, los componentes, los esquemas de conexión, los instrumentos de programación además se especifican las características y el diseño del hardware y software del prototipo.

2.1 Requerimientos del prototipo

Se planteó la implementación de un prototipo de detección de mascarilla y medición de temperatura para el personal que ingresa a la empresa PUFIT, usando técnicas de visión artificial, para ello se determinó cumplir con los siguientes requerimientos:

- Detectar la presencia de la persona cuando esta se encuentre de frente a la cámara del dispositivo, para realizar la identificación de los usuarios que usen o no mascarilla.
- Ser un dispositivo funcional de bajo costo
- Definir el conjunto de reglas que los usuarios de la empresa PAUFIT deber seguir para poder acceder a su lugar de trabajo.

En este requerimiento se planteó la implementación de un protocolo de comunicación apropiado para controlar el acceso mediante la verificación de usuarios que no cumplan con el uso de mascarilla, mediante 3 tipos de alerta:

- Auditiva
- Visual
- Sistema de Notificación

2.1.1 Requerimientos de Hardware

Para el correcto funcionamiento del prototipo, considerando que la implementación de la etapa de detección de mascarilla se lo realizó mediante visión artificial y que además el prototipo también realiza la medición de temperatura, se procedió a seleccionar el hardware necesario para cumplir sus requerimientos funcionales:

- Precisar el componente hardware adecuado a usar en la etapa de adquisición y procesamiento de imágenes.
- Establecer el componente hardware necesario para la implementación de detección de temperatura.
- Determinar el componente hardware necesario a usar en el protocolo de comunicación para el control de acceso

2.1.2 Requerimientos de Software

Uno de los puntos de vista más relevantes para la ejecución del presente trabajo es el requerimiento de software debido a que todo el sistema es dependiente de la programación e instalación. Los requerimientos con respecto al sistema operativo y de las aplicaciones instaladas van enfocados al uso del Software OpenCV y Arduino Uno, el programa debe ser compatible con las placas de desarrollo y con sus requerimientos. Además de ciertos aspectos relevantes los cuales son:

- Lectura y comunicación correcta de los sensores
- Comunicación Raspberry Pi-Arduino Uno
- Establecer el uso de software libre

2.2 Arquitectura del prototipo

La figura 1-2 representa la arquitectura del sistema detector de mascarilla y medición de temperatura, así como su interacción con los elementos de control y le alerta.

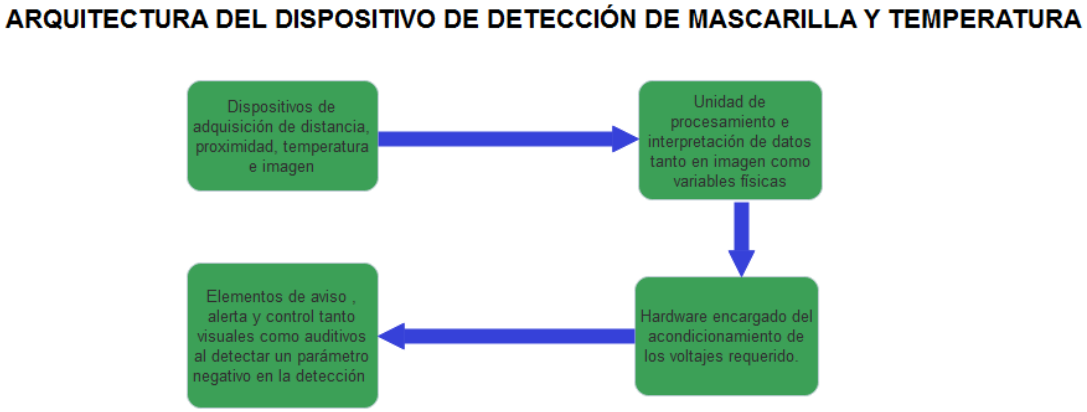

Figura 1-2. Arquitectura del sistema de detección de mascarilla y temperatura

Realizado por: Paullán, A., 2021

2.3 Hardware

En este apartado se establece una descripción detallada de los equipos y accesorios seleccionados en la implementación del prototipo, siendo elegidos aquellos que cumplieron con la mayoría de las características y condiciones requeridas como efectividad de desempeño, eficiencia, facilidad de uso, flexibilidad y economía.

2.3.1 Adquisición de información

Para la adquisición de información se utilizó la cámara para Raspberry Pi. La información adquirida es enviada a través del cable plano al conector CSI dedicado de la Raspberry Pi. En la figura 21-2 se puede observar la cámara que se consideró la adecuada para el prototipo debido a su compatibilidad con el sistema de Raspbian, además de poseer una mejor resolución frente a las cámaras web (USB). En la tabla 1-2 se muestra las características de la cámara para Raspberry Pi.

Tabla 1-2: Características Cámara para Raspberry Pi

Característica	Descripción
Sensor	Óptico Omnivision 5647
Resolución	5 MegaPixeles
Imagen	Resolución de 2592 x 1944
Clip de video	1080p a 30 fps
Dimensiones	25 mm X 20 mm X 9mm.

Fuente: (Lopez, 2019)

Realizado por: Paullán, A., 2021

Figura 2-2. Cámara Raspberry Pi

Fuente: https://tienda.bricogeek.com/3115-thickbox_default/camara-raspberry-pi-v2-8-megapixels.jpg

2.3.2 Unidad de Procesamiento

Para el desarrollo del prototipo propuesto se utilizó una tarjeta de desarrolló Raspberry Pi4, por su bajo costo, bajo consumo de energía y por sus prestaciones mejoradas como su rapidez que es 6 veces mejor en comparación de modelos anteriores, esta se encargó de realizar el procesamiento de las imágenes, así como también de codificar el algoritmo para realizar la detección de mascarilla, en la figura 3-2 se puede observar el modelo de Raspberry Pi4 y la tabla 2-2 muestra las características del dispositivo.

Figura 3-2. Raspberry Modelo Pi4 B

Fuente: https://cdn.alzashop.com/ImgW.ashx?fd=f3&cd=RK100b3

Tabla 2-2: Características de la Raspberry Pi4

CARACTERÍSTICA	DESCRIPCIÓN
Procesador	ARM Cortex-A72 de 64 bits con cuatro núcleos a 1.5 GHz
Almacenamiento	1 GB ,2 GB o 4 GB SDRAM LPDDR4
Conexión	Gigabit Ethernet, Red inalámbrica 802.11ac de doble banda, Bluetooth 5.
Puertos	2 USB 3.0 y 2 USB 2.0

Fuente: (Lopez, 2019)

Realizado por: Paullán, A., 2021

2.3.3 Arduino Uno

Para el proceso de toma de temperatura se determinó incorporar un Arduino Uno, este dispositivo trabaja con un protocolo de comunicación serial I2C resultando ideal para la comunicación con el sensor de temperatura GY906 y la Raspberry Pi, en la figura 4-2 se puede observar el Arduino Uno y en la tabla 3-2 se describe las características técnicas del mismo.

Figura 19-2. Tarjeta Arduino Uno

Fuente: https://i0.wp.com/arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg

Tabla 3-2: Características técnicas de la placa Arduino UNO

CARACTERÍSTICA	DESCRIPCIÓN
Microcontrolador:	ATmega328
Voltaje de Operación	5V
Voltaje de Entrada (recomendado)	7-12V
Entrada/Salida Digitales:	14 (de ellos 6 con salida PWM)
Entradas Analógicas	6
Memoria Flash	32 kb
EEPROM	1KB
Velocidad	16 MHz

Fuente: (Pomare, 2009)

Realizado por: Paullán, A. 2021

2.3.4 Amiplificador de Audio PAM8403

Para el mensaje de alerta determinado para el prototipo, se utilizó el amplificador de audio PAM8403. La tabla 4-2 contiene las características del amplificador.

Tabla 4-2: Características del amplificador PAM8403

CARACTERÍSTICA	DESCRIPCIÓN
Potencia de Salida	3 W
Señal a Ruido	90 db
Voltaje de alimentación mínimo:	2.5 V
Voltaje de alimentación máximo:	5 .5V
Eficiencia	Mayor al 90%

Fuente: (Ovalle, 2019)

Realizado por: Paullán, A. 2021

Se optó por este dispositivo ya que es un amplificador de audio que ofrece una baja distorsión armónica, alta eficiencia y calidad de reproducción, su nueva arquitectura sin filtros permite que el dispositivo accione el altavoz directamente, sin necesidad de filtros de salida de paso bajos, generando ahorro en costos del sistema y en el área de montaje, el proceso de amplificación de la señal de audio se realiza a través de un amplificador operacional, en donde la ganancia es variable. En la figura 5-2 se muestra el dispositivo.

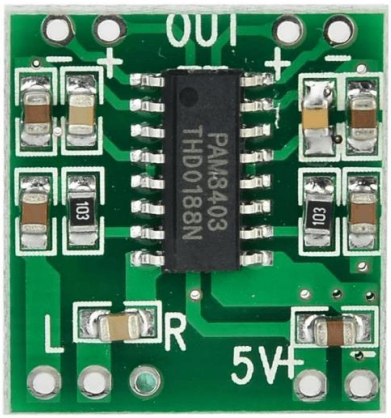

Figura 5-2. Amplificador PAM8403

Fuente: https://n9.cl/p6sx4

2.3.5 Placa de desarrollo NodeMCU ESP8266

La placa de desarrollo NodeMCU fue usada para implementar como medio receptivo, a la cual se incorporó una tarjeta compatible con el protocolo de comunicación MQTT, este protocolo se basa en el envío y recepción de mensajes, el cual está orientado a la comunicación máquina a máquina en el internet de las cosas, el MQTT predomina primordialmente por su sencillez, escalabilidad, bajo consumo de ancho de banda, además la comunicación podría ser de uno a uno o de uno a varios. Hoy en día esta placa es programada en la mayoría de las aplicaciones para internet de las cosas, una ventaja de usar esta tarjeta es el bajo costo que esta tiene y que es compatible con la plataforma de Arduino Uno. La figura 6-2 muestra el dispositivo NodeMCU y la tabla 5-2 describe las características técnicas del dispositivo.

Figura 6-2. Dispositivo NodeMCU.

Fuente: https://www.aranacorp.com/wp-content/uploads/nodemcu-v3-esp8266.jpg

Tabla 5-2: Características técnicas de la placa NodeMCU

CARACTERÍSTICA	DESCRIPCIÓN
Regulador	3 .3 V
Wifi	802.11 b/g/n
Pines GPIO	9
Alimentación externa	20 V Max

Fuente: (BricoGeek, 2021)

Realizado por: Paullán, A. 2021

2.3.6 Sensor Ultrasónico

Estos sensores detectan objetos a distancias que van desde pocos centímetros hasta varios metros. A través del sensor ultrasónico se detecta la presencia de una persona cuando esta se encuentra frente al dispositivo, de esta manera junto con el sensor infrarrojo de temperatura proceden a enviar una señal para que la cámara realice la adquisición de la imagen y posteriormente realice la detección. Además, se eligió este dispositivo por su precisión, reducido tamaño, bajo consumo de energía y un bajo costo. La figura 7-2 representa el sensor ultrasónico utilizado en el prototipo. Además, la tabla 6-2 describe las características del dispositivo.

Figura 7-2. Sensor Ultrasónico.

Fuente: https://n9.cl/cei2v

Tabla 6-2: Características del sensor ultrasónico

CARACTERÍSTICA	DESCRIPCIÓN
Voltaje de Operación	+- 3mm
Exactitud	40 kHz
Corriente de trabajo	15 mA
Rango de medición	2cm a 450cm

Fuente: (Cárdenas, 2015)

Realizado por: Paullán, A. 2021

2.3.7 Sensor de temperatura Infrarrojo

Para la detección de temperatura se ha eligió el sensor de temperatura infrarrojo GY 906, este sensor toma la temperatura sin necesidad de hacer contacto directo sobre el área en cuestión, este sensor tiene una alta exactitud y una sorprendente resolución, su interfaz de comunicación es bastante simple de utilizar debido a que puede comunicarse por medio del protocolo I2C, además es un dispositivo de bajo consumo. La figura 8-2 muestra el sensor de temperatura infrarrojo, además la tabla 8-2 muestra las características del dispositivo.

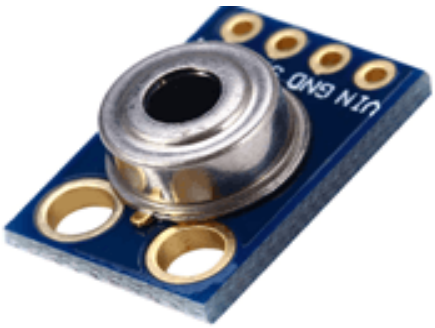

Figura 8-2. Sensor de temperatura

Fuente: (Llamas, 2016)

Tabla 7-2: Características del sensor de temperatura

CARACTERÍSTICA	DESCRIPCIÓN
Voltaje de alimentación	3 V
Exactitud	0.5°C
Corriente nominal	2.5 mA
Rango de medición	-70 a 380 °C

Fuente: (Llamas, 2016)

Realizado por: Paullán, A. 2021

2.3.8 Tarjeta de Acondicionamiento

Para la comunicación de la Raspberry Pi, Arduino Uno y los sensores se utilizó una tarjeta para el acondicionamiento de la señal, debido a que los sensores trabajan a un voltaje diferente de la Raspberry Pi, la figura 9-2 muestra el diseño de la tarjeta de acondicionamiento.

Figura 9-2. Diseño de la tarjeta de acondicionamiento

Realizado por: Paullán, A. 2021

2.3.9 Modulo Rele

Para el control de acceso se optó por utilizar un módulo de relés, el cual estará encargado de activar la puerta mediante un accionamiento eléctrico. Se optó por este módulo debido a la compatibilidad que tiene con los controladores además por facilidad de uso ya que no requiere de librerías específicas para su control. La figura 10-2 muestra el módulo de relés, además la tabla 8-2 muestra las características del dispositivo.

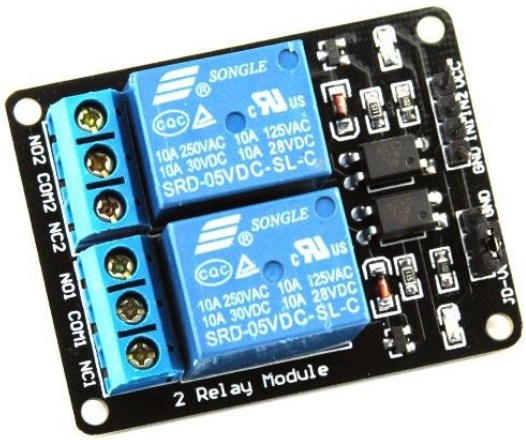

Figura 10-2. Sensor de temperatura

Fuente: (Iberobotics, 2020)

Tabla 8-2: Características del sensor de temperatura

CARACTERÍSTICA	DESCRIPCIÓN
Voltaje de operación	5 V
Canales	2 canales optoacoplador
Corriente de activación	20 mA
Tamaño	44,4 x 32,4 mm

Fuente: (Iberobotics, 2020)

Realizado por: Paullán, A. 2021

2.3.10 Conexiones del Hardware

La figura 11-2 representa el esquema de conexión, el cual esa formado por elementos electrónicos que se usaron para la creación del prototipo, mediante el uso de sensores, tarjetas controladoras, y actuadores mismos que interactuando entre sí realizaron la medición de temperatura e identificaron el uso correcto de la mascarilla.

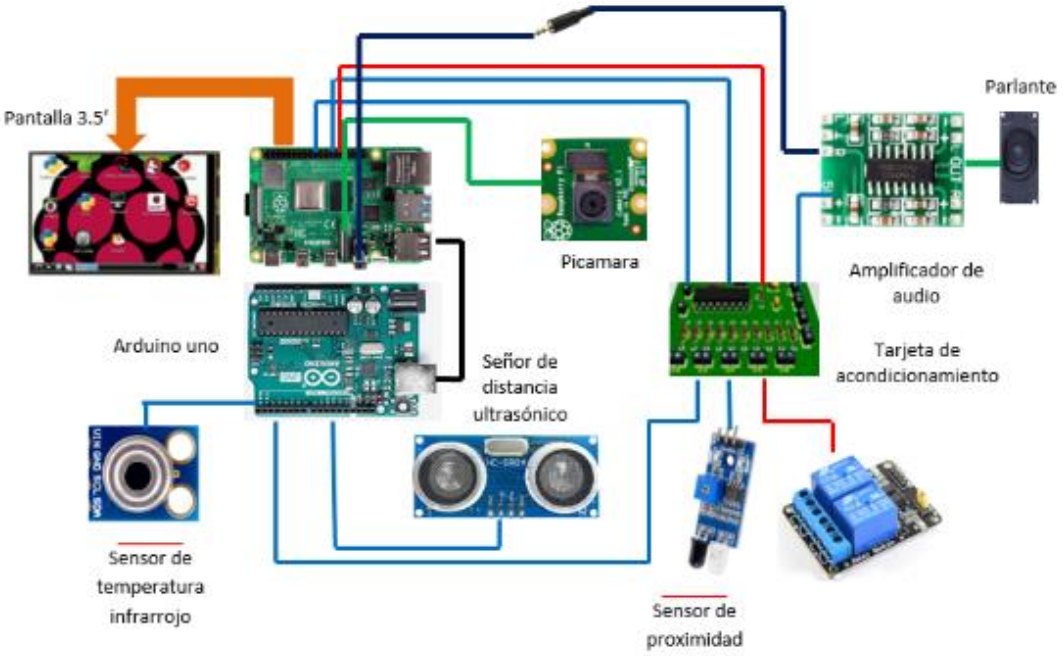

Figura 11-2. Esquema de conexión

Realizado por: Paullán, A. 2021

El esquema de conexión se lo realizó de la siguiente manera, la pantalla de 3.5 pulgadas se conectó a los 40 primeros terminales de la Raspberry Pi, el cual a través de un bus de datos se procedió a enlazar a la cámara. En la tarjeta Arduino Uno se incorporó los sensores tanto de distancia como de temperatura, así como el cable usado para la comunicación serial y por último una tarjeta de acondicionamiento de voltajes y amplificador de audio cabe mencionar que el amplificador se incorpora en la misma tarjeta, pero para mayor compresión de las conexiones, esta se la ha representado por separado como se puede observar en la figura 11-2. Esta tarjeta de acondicionamiento fue diseñada para el prototipo, a la cual se ingresó los voltajes de 5 y 3.3 voltios, así como el sensor de proximidad, la entrada de audio y por último la salida de los parlantes para las alertas auditivas. Además del esquema de conexiones la figura 12-2 muestra un diagrama eléctrico de conexiones implementado en el prototipo.

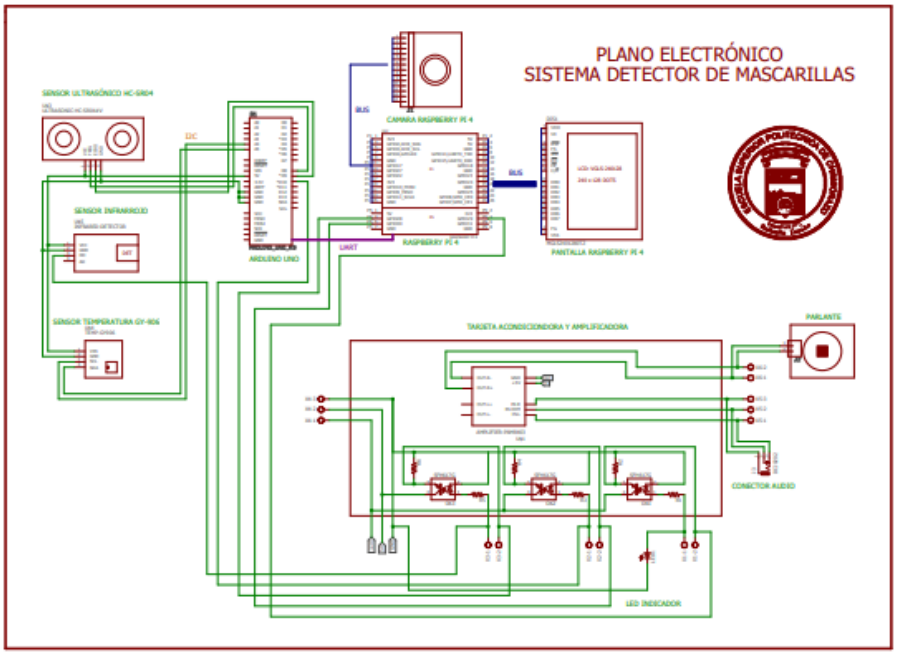

Figura 12-2. Diagrama eléctrico de conexiones.

Realizado por: Paullán, A. 2021

2.4 Software

En este apartado se explica el procedimiento que se llevó a cabo para el desarrollo del algoritmo de visión artificial, así como también los pasos y consideraciones a tomar para realizar el trabajo y cumplir con los objetivos propuestos.

2.4.1 Montaje del sistema Operativo en la Raspberry Pi

Lo primero es obtener una tarjeta Microsd la cual se insertó a la Raspberry Pi, después de insertarla en el ordenador la primera tarea que se realizó fue dar un formateo rápido para afirmar que no hay nada en ella, a continuación, se procedió a cargar la imagen del sistema operativo y a configurar ciertos parámetros como el país y la hora. Dentro del sistema operativo se cuenta con programas predeterminados como por ejemplo edición de texto, edición de código, pero el programa más importante y el que se utilizó en la programación, Python no está instalado por defecto, así que se procedió a instalar mediante una serie de comandos. La figura 13-2 muestra el entorno del sistema operativo Raspbian instalado.

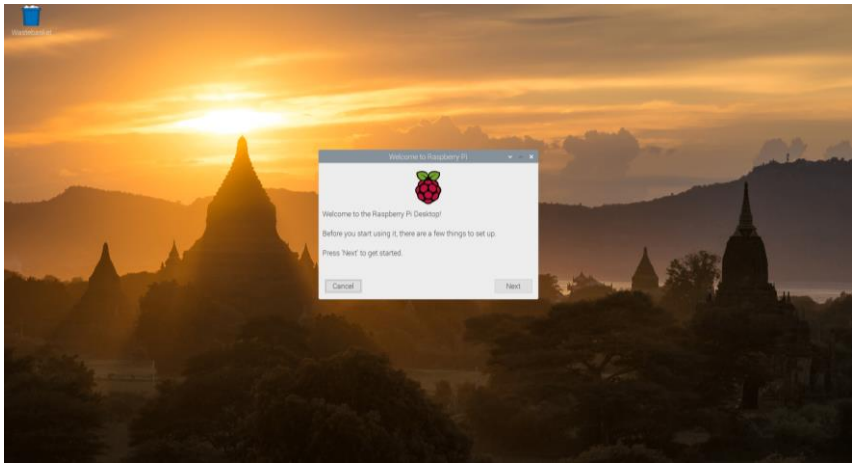

Figura 13-2. Entorno del sistema operativo Raspbian

Realizado por: Paullán, A. 2021

2.4.1.1 Instalacion de Python 3.7

Se instaló la versión 3.7 de Python en Raspbian, para ello la figura 14-2 muestra el código usado para realizar una correcta instalación, estos comandos permiten realizar una actualización de los paquetes instalados por defecto, así como también permite instalar los complementos que Python requiere.

```
sudo apt-get update -y
sudo apt-get install build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-
dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev
zlib1g-dev libffi-dev -y
wget https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tar.xz
tar xf Python-3.7.4.tar.xz
cd Python-3.7.4
./configure
make -j 4
sudo make altinstall
Sudo apt-get install idle3.7
```

Figura 14-2. Comandos para instalar Python en Raspbian

Fuente: (GitHubGist, 2020)

Realizado por: Paullán, A. 2021

2.4.1.2 Instalación de OpenCV

Para la aplicación de las técnicas de visión artificial y el procesamiento de las imágenes se necesita la librería de OpenCV, la cual permite llamar a las funciones para poder realizar capturas de imágenes y procesarlas. Para realizar una correcta instalación se lo hizo a través de los comandos que se muestran en la figura 15-2, por medio de estos comandos se importó todas las herramientas y bibliotecas que se requirieron para poder realizar el trabajo.

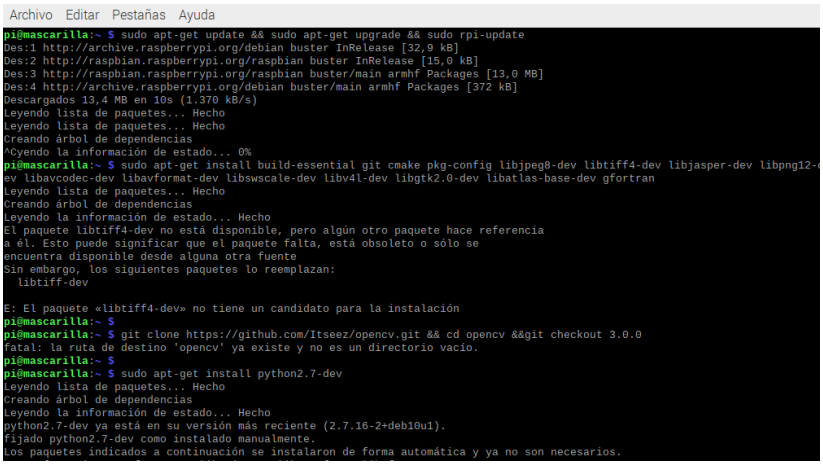

Figura 15-2. Comandos OpenCV

Realizado por: Paullán, A. 2021

La instalación tarda unos minutos ya que se debe instalar todas las herramientas y paquetes para que no exista ningún error en su funcionamiento, una vez realizado esto se procede a compilarlo en la Raspberry Pi, si la compilación fue exitosa se observa los datos que se muestran en la figura 16-2 , dejándolo listo para poder utilizarlo.

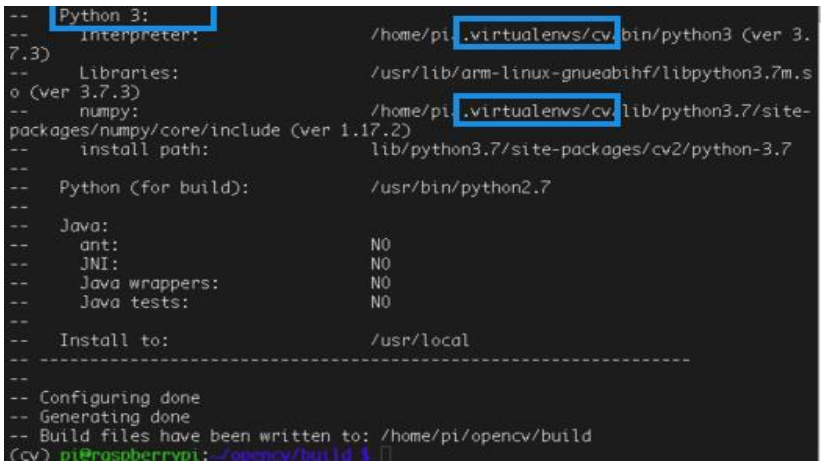

Figura 16-2. Instalación correcta de OpenCV en Python

Realizado por: Paullán, A. 2021

2.4.1.3 Instalación de TensorFlow y librerias complementarias

Para poder hacer uso de la librería TensorFlow se es necesario de ciertas funciones para su correcta compilación, a través de estas funciones se permitió acceder a la última versión de TensorFlow la cual cuenta con la arquitectura del modelo, permitiendo así acceder a los pesos del modelo de entrenamiento, la figura 17-2 muestra el código empleado.

```
$sudo pip install --upgrade pip
$sudo pip3 install --upgrade setuptools
$sudo pip3 install numpy==1.19.0sudo apt-get install -y libhdf5-dev libc-ares-dev
$libeigen3-dev gcc gfortran python-dev libgfortran5 libatlas3-base libatlas-base-dev
libopenblas-dev libopenblas-base libblas-dev liblapack-dev cython libatlas-base-dev
openmpi-bin libopenmpi-dev python3-devsudo pip3 install keras_applications==1.0.8 --no-
deps
$sudo pip3 install keras_preprocessing==1.1.0 --no-deps
$sudo pip3 install h5py==2.9.0
$sudo pip3 install pybind11pip3 install -U --user six wheel mock
$wget https://raw.githubusercontent.com/PINTO0309/Tensorflow-bin/master/tensorflow-
2.3.0-cp37-none-linux_armv7l_download.sh
$sudo chmod +x tensorflow-2.3.0-cp37-none-linux_armv7l_download.sh$ ./tensorflow-2.3.0-
cp37-none-linux_armv7l_download.sh
$sudo pip3 uninstall tensorflow
$sudo -H pip3 install tensorflow-2.3.0-cp37-none-linux_armv7l.whl
```

Figura 17-2. Comandos para instalar Tensorflow

Fuente: (TensorFlow, 2020)

Realizado por: Paullán, A. 2021

2.4.2 Instalación del Protocolo MQTT en la Raspberry Pi

El protocolo MQTT se utilizó como complemento del control de acceso, debido a que si la persona usa mascarilla y su temperatura se encuentra en los rangos normales establecidos se procedió a dar paso para el ingreso. Se usó este método debido a las ventajas con las que cuenta, como su facilidad de funcionamiento y porque requiere un pequeño ancho de banda para permitir la comunicación. Además, es bidireccional, pero en este caso para esta aplicación se lo trabajara de manera unidireccional, ya que solo se requiere enviar una variable de control para el accionamiento de una puerta mediante un control eléctrico.

Antes de instalar el broker MQTT en la Raspberry Pi, se actualizó el sistema operativo, así como las librerías instaladas a través de los siguientes comandos.

sudo apt update

sudo apt full upgrade

Una vez que el sistema ha terminado de actualizarse, se procedió a instalar el software MQTT a través de la siguiente línea de código *sudo apt install mosquitto mosquitto-clients*, luego de ello se instaló la librería *mosquitto-clients* que permitió interactuar y probar que el bróker MQTT se esté ejecutando correctamente en la Raspberry Pi. Durante el proceso de instalación, el administrador de paquetes configuro automáticamente el servidor MQTT para que se inicie en el arranque. Luego de ello se verifico que esté instalado y en ejecución a través del siguiente comando *sudo systemctl status mosquitto*, el cual envió un mensaje de texto diciendo "activo (en ejecución) comprobando así que se inició correctamente.

2.4.3 Configuración de la pantalla de 3.5 pulgadas para la visualización

Para que la persona pueda ver su rostro se procedió a configurar una pantalla de 3.5 pulgadas, esto con el fin de hacer más interactivo el contacto con el usuario, para la configuración de esta pantalla se debe implementar algunos comandos en el terminal del sistema operativo Raspberry Pi. Para la configuración de la pantalla lo primero que se procedió a realizar es descargar los drivers necesarios para el funcionamiento del dispositivo, luego de ello a través del comando cd LCD-show/ se procede a dar los permisos respectivos para acceder la carpeta desde donde se va a ejecutar la configuración de la pantalla. Luego de haber realizado esto se procede a calibrar la pantalla a través del siguiente comando sudo. /LCD35-show, dejando así habilitada la pantalla LCD. La figura 18-3 muestra la configuración de la pantalla.

Figura 18-3. Configuración de la pantalla LCD

Realizado por: Paullán, A. 2021

2.4.4 Programa de recolección de imágenes para el entrenamiento

Para la creación del programa se usó las variables provenientes del sensor de distancia ultrasónico y del sensor de detección de presencia infrarrojo para hacer el procesamiento de recopilación de imágenes. El grafico 1-3 detalla la secuencia para realizar la adquisición de imágenes.

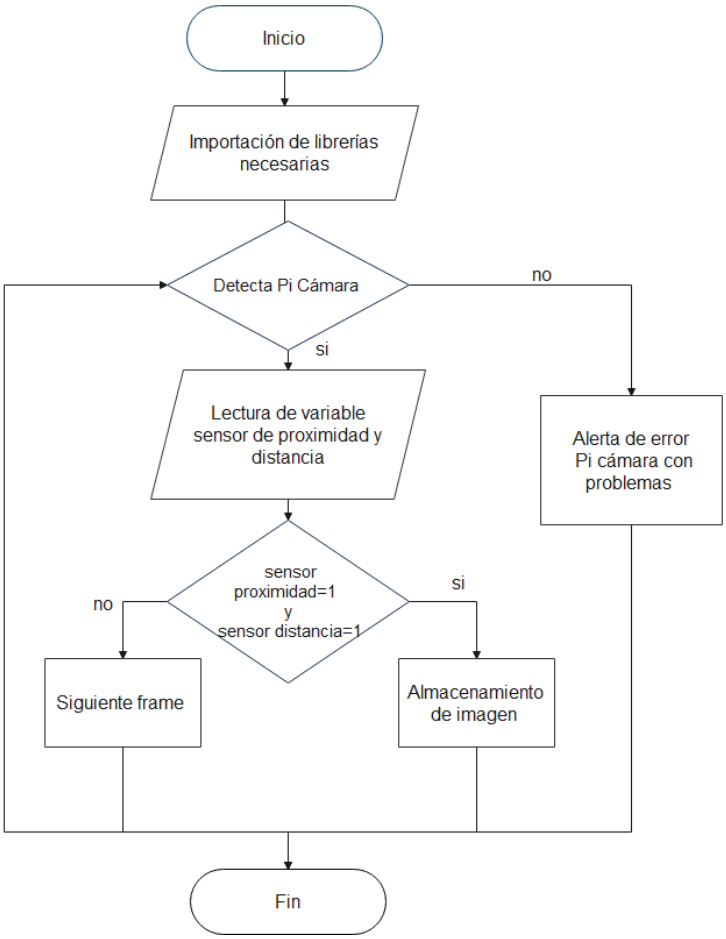

Gráfico 1-3. Diagrama de flujo para recolección de imágenes

Realizado por: Paullán, A. 2021

El programa inicia importando las librerías necesarias para la adquisición de imágenes, como se muestra en la figura 19-2. En la primera línea de la imagen se muestra la función que permitió acceder a la cámara a través de un módulo que se incorpora a la tarjeta por medio de un bus de datos, esta contiene las funciones necesarias para que la cámara realice las capturas correspondientes. En la segunda línea se muestra la librería de visión artificial la cual permite acceder a todas las funciones de la librería OpenCV. Se accede a la librería de Machine Learning a través del comando *numpy*, como se ve en la tercera línea de código de la imagen, la cual permite realizar un análisis vectorial, matricial y estadístico, también permite asignar un número de funciones para trabajar con vectores, transformando así imágenes a vectores, y realizando una comparación de los análisis estadísticos de operaciones con matrices y estadística en general. Y por último la cuarta line de código que se muestra en la imagen, permitió acceder a las entradas y salidas digitales de la Raspberry Pi.

```
from picamara.array import PiRGBArray
import cv2
import numpy as np
import RPi.GPIO as GPIO
```

Figura 19-2. Librerías y funciones importadas para la recolección de imágenes.

Realizado por: Paullán, A. 2021

Además de ello esta última librería permitió configurar los *GPIOS*, *q*ue básicamente son terminales físicos implementados como entradas o salidas digitales, en este caso el dispositivo cuenta con dos entradas, una de ellas para detectar presencia por medio de un sensor de distancia ultrasónico y la otra un sensor de corto alcance infrarrojo para la temperatura. La detección de flanco como lo plantea el algoritmo será acondicionada para la lectura del sensor de distancia y para la lectura del sensor infrarrojo, los cuales se usarán en este caso para hacer una captura de una imagen ya sea de persona con mascarilla o sin mascarilla, las cuales se proceden a guardar en dos carpetas distintas, adaptando así a los sensores para que junto con la cámara sea un dispositivo de captura de imágenes para obtener un data base. La figura 20-2 muestra una ilustración grafica de cómo se realizó la recolección de imágenes.

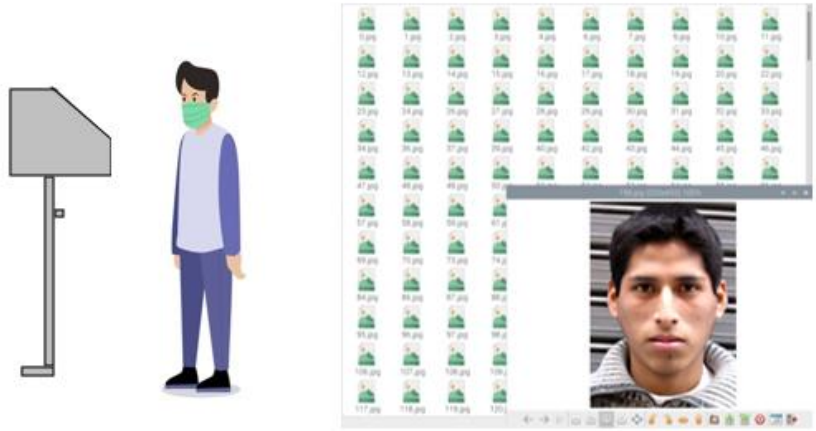

Figura 20-2. Recolección de imágenes para el data base

Realizado por: Paullán, A. 2021

2.4.4.1 Entrenamiento del modelo de Red Neuronal

Una vez creada la base de datos con las imágenes de los rostros de las personas con mascarilla y sin mascarilla se procedió a realizar el entrenamiento de la red neuronal. En este caso para el diseño de la arquitectura MobileNetV2, que es un modelo pre entrenado, eficiente y fácil de implementar en Python, con el uso de librerías como Keras y TensorFlow. Para mejorar esta arquitectura se implementó el algoritmo que se observa en el grafico 2-3, una arquitectura que se puede implementar en la Raspberry Pi debido a su bajo requerimiento por parte del procesador.

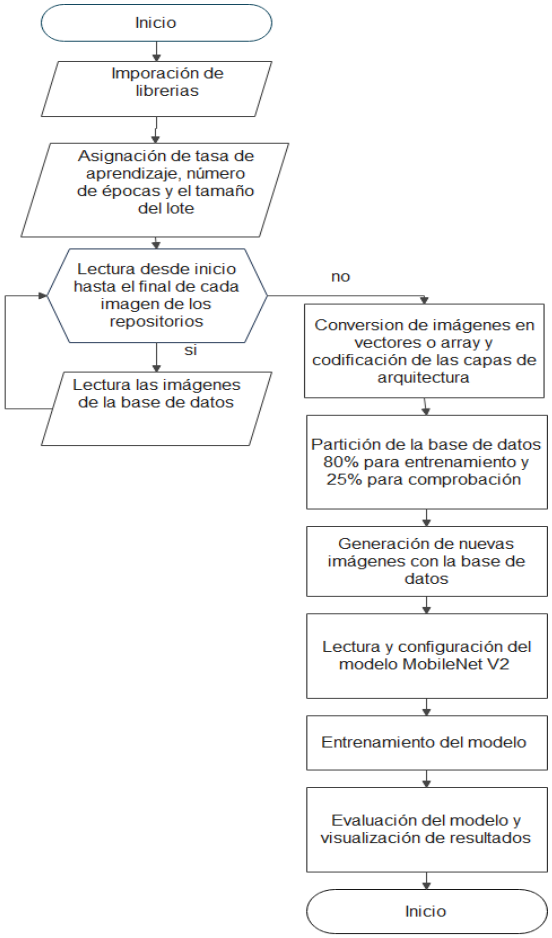

Gráfico 2-3. Diagrama de flujo para entrenamiento

Realizado por: Paullán, A. 2021

2.4.4.2 Importación de librerias usadas para el entrenamiento de la red neuronal

Para empezar, se necesita llamar a las librerías que serán utilizadas dentro del entrenamiento, una de las principales son TensorFlow y Keras, las cuales permiten la conversión y normalización de las imágenes a un vector para que se incorporen en el modelo mediante un método supervisado. La figura 40-2 muestra las librerías y funciones usadas dentro del entrenamiento de la red neuronal, de la cuales se procede a realizar una breve descripción.

```
#librerias necesarias
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
```

Figura 21-2. Librerías necesarias para el entrenamiento de la Red Neuronal

Realizado por: Paullán, A. 2021

A través de la primera línea de código que se puede ver en la figura 21-2 se importó la función I*mageDataGenerator* la cual genera imágenes modificadas a partir de un determinado conjunto de datos en este caso imágenes de personas con y sin mascarilla, estas nuevas imágenes son creadas a partir del cambio de perspectiva, de esta manera no se necesita una base de datos con demasiadas imágenes para poder tener mejores resultados, esta técnica es muy utilizada cuando no se cuenta con una gran cantidad de datos.

La segunda línea muestra la función *MobileNetV2* la cual inicializa la arquitectura a utilizar en el programa, de esta manera mediante el modelo pre entrenado se procedió a realizar un nuevo entrenamiento a partir de la base de datos creada con el programa de recopilación de imágenes.

La siguiente línea muestra la función *AveragePooling2D* la cual realiza una operación de agrupación promedio para los datos espaciales, esta es necesaria para ir recorriendo en la estructura por toda la imagen.

En la siguiente línea de código a través de la función *adams* se importó el algoritmo de optimización Adam que se puede utilizar en lugar del método de descenso de gradiente estocástico tradicional, para actualizar los pesos de red de manera iterativa en funcionalidad de los datos de entrenamiento. Se utilizó este optimizador por su simplicidad al momento de implementarlo, porque es computacionalmente eficiente, tiene pequeños requisitos de memoria y porque los hiperparámetros poseen una interpretación intuitiva y comúnmente necesitan pocos ajustes. (Brownlee, 2017)

La siguiente función *Flatten* establece aleatoriamente las unidades de entrada en 0 con una frecuencia de velocidad en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar

el sobreajuste, las entradas que no permanecen configuradas en 0 se escalan en 1/(1-tasa) de forma que la suma de cada una de las entradas no cambia. Por otra parte, la función *to_categorical* permite tomar la información de datos binarios, esta función se maneja con datos ceros y unos, al momento de pasar por esta función de activación del modelo va aprendiendo mediante un error.

La función *train_test_split* de la librería import *sklearn* puede subdividir un conjunto de datos en subconjuntos que minimizan el potencial de sesgo en su proceso de evaluación y validación, para no generar una sobrecarga en el entrenamiento. Con la función *classification_report* de la librería import *sklearn* crea un informe de texto que muestra las principales métricas de clasificación. La función *paths* de la librería *imutils* crea una lista para abrir las imágenes de los rostros que tienen mascarillas y de los que no la tienen, almacenándolos en una variable diferente y la función *matplotlib.pyplot* sirvió para graficar una o varias señales que son necesarias para mostrar el proceso del aprendizaje y la disminución del error.

2.4.4.3 Analizador de argumentos para la creación de variables

Al existir un conjunto de imágenes en dos carpetas diferentes se dificulta el poder llamarlo a Python desde una dirección específica, para ello se usó la librería *argparse* que permite organizar de mejor manera estos archivos. Lo mismo se debe hacer para almacenar tanto el plot de salida del modelo de aprendizaje, así como la ruta donde se guardará el modelo entrenado. A continuación, se muestra el código empleado.

```
#Construcción del analizador de argumentos para la creación de variables.
ap = argparse. ArgumentParser ()
ap.add_argument ("-d", "--dataset", required =True,
    help= "dirección de la data base")
ap.add_argument ("-p", "--plot", type =str, default= "plot.png",
    help = "dirección en donde se guardará")
ap.add_argument ("-m","--model' 'type=str,
    default ="mascarilla. detector. Model"
    help="dirección de salida una vez que el modelo termine el entrenamiento")
args = vars (ap. parse_args ())
```

Los argumentos de línea de comando incluyen:

- dataset: ruta o dirección de entrada a la base de datos que contiene las imágenes de los rostros de las personas con y sin mascarilla.

- plot: ruta especifica o dirección del grafico en donde se almacena el historial de entrenamiento de salida que se genera usando matplotlib.

- model: ruta o dirección al modelo de clasificación de mascarilla resultante del entrenamiento

También se debe especificar constantes de híperparámetros que incluyen la tasa de aprendizaje inicial, el número de épocas de entrenamiento y el tamaño del lote.

2.4.4.4 Asignación de tasa de aprendizaje, número de épocas y el tamaño del lote

La tasa de aprendizaje es un hiperparámetro que controla cuánto cambia el modelo en respuesta al error estimado, cada vez que se actualizan los pesos del modelo durante el entrenamiento, este valor se encuentra a menudo entre el rango de 0.0 y 1.0. Para este entrenamiento se asignó una taza de aprendizaje de 1e-4, a través de la siguiente línea de código *INIT_LR = 1 e -4*, siendo este valor el recomendado para el manejo de esta arquitectura.

El número de épocas es un hiperparámetro que asigna el número de veces que el programa de aprendizaje funcionará en todo el conjunto de datos de entrenamiento. El número de épocas a menudo es tradicionalmente grande, cientos o miles, de datos, que permite que el algoritmo de aprendizaje se ejecute hasta que el error del modelo se haya minimizado lo suficiente. Para este entrenamiento se asignó 20 épocas a través de la siguiente línea de código *EPOCHS = 20*, por dos razones, la primera es porque el modelo puede dar buenos resultados con menor cantidad de épocas y la otra razón es que al utilizar una cantidad reducida de épocas se optimizo los recursos de hardware.

Cabe mencionar que el tamaño del lote es una cantidad de muestras procesadas antes de que se actualice el modelo, para el entrenamiento se contó con un total de 600 imágenes de personas con mascarillas y 600 imágenes de personas sin mascarilla en la base de datos, al especificar este parámetro se está estableciendo un límite del lote que será entrenado de forma iterativa, para no sobrecargar el entrenamiento, determinando así el tamaño del lote a través de la siguiente línea de código *BS = 32*, el valor de lote asignado será de 32 lo que quiere decir que se entrenara cada nueva época cada 32 lotes de imágenes.

2.4.5 Preprocesamiento y conversión de imágenes de la base de datos.

Una vez creada las variables que permitieron abrir y guardar los archivos que se crearon en el entrenamiento el siguiente paso es establecer las variables implícitas en la programación, donde

se almacenará la lista de imágenes que se encuentran en la base de datos. A través del comando *list* en la variable *dirección_imagen,* con el parámetro *paths.list_imageabre* se crea una ruta específica que almacena en esta variable las imágenes. Luego de ello se asigna un bucle de iteración con la variable *dirección_imagen* con la finalidad de ir separando imagen por imagen, dicha operación se la realiza con el comando *imagePath.slit* y este proceso se almacena en la variable *labels.* Lo siguiente que se realiza es un preprocesamiento de las imágenes.

Los pasos de preprocesamiento incluyen cambiar el tamaño a 224 × 224 píxeles con la función *load_image,* con el parámetro *target_size,* como se observa en la figura 22-2. Luego de esto a través de la función *img_to_array* realiza la conversión de una imagen a formato matriz y la función *preprocess_input* escalará las intensidades de píxeles que van desde 0 a 255 en la imagen de entrada al rango [-1, 1].Además de ello con el comando *appened* se agrega la imagen pre procesada en la variable *image y labels* asociada a la listas de datos y etiquetas, respectivamente, es decir que se formó un array de manera consecutiva dentro del bucle for almacenando así las imágenes producto del pre procesamiento.

Figura 22-2. Normalización de imagen

Realizado por: Paullán, A. 2021

2.4.6 Segmentación de la base de datos

Para la aplicación de este modelo dentro del campo del Deep Learning que trabaja con redes neuronales convolucionales, se procedió a entrenar la red neuronal con el 80% de las imágenes de la base de datos y el 20% restante se usó para realizar una evaluación del proceso de entrenamiento. La función *train_test_split* es la encargada de subdividir el conjunto de datos o imágenes para el entrenamiento. Mientras que para la evaluación como parámetro de entrada se ingresaron las variables *data* y *labels* en las cuales se especificaron el porcentaje utilizado para comprobar la eficacia de modelo. A continuación, el código empleado.

 # Segmentación al 80% de datos para el entrenamiento
(trainX, testX, trainY, testY) = train:test:Split(data, labels , test_size = 0.20, straify = labels, random_state = 42)

2.4.7 Creación de nuevas imágenes a partir de la base de datos

Se podría decir que al contar con 705 imágenes estas serían suficientes para realizar un buen entrenamiento del modelo, la verdad que aun con esa cantidad de información no es suficiente ya que para desarrollar un algoritmo de visión artificial aplicando Deep Learning, usar un lote por debajo de 1000 imágenes no es suficiente para obtener resultados satisfactorios. Para ello se implementó una función que permite modificar cada imagen tanto en su rotación, traslación, cambio de escala, e incluso volteándola, permitiendo así obtener una imagen nueva en algunos aspectos en comparación con la original, para ello se crea un constructor que es una variable que contiene algunos parámetros de entrada para realizar esta operación, A continuación, se muestra el código empleado.

```
#construccion del generador de imágenes con el data base que se tiene
aumento = ImageDataGenerator (
          rotation_range=20,
          zoom_range=0.15,
         width_shift_range=0.2,
        Height_shift_range= 0.2,
        shear_range=0.15,
       horizontal_flip=True,
       fill_mode= ''nearest'')
```

En el entrenamiento, se aplicará mutaciones sobre las imágenes, todo esto en un esfuerzo por mejorar la generalización de las imágenes, a esto se lo conoce como aumento de datos, donde los parámetros de rotación aleatoria, corte, desplazamiento, volteo y zoom usan la función *ImageDataGenerator*, que es muy útil cuando se cuenta con un data base limitado para generar el cambio de perspectiva en las imágenes. La figura 23-2 representa un ejemplo del constructor.

Figura 23-2. Generación de nuevas imágenes.

Realizado por: Paullán, A. 2021

2.4.8 Configuración de parámetros del modelo pre entrenado MobileNetV2

La arquitectura con la que se realizó el entrenamiento es MobileNetV2, para cargar el modelo se lo hizo a través de la función *MobileNetV2,* luego de haber realizado esto se procedió a configurar el tamaño de las imágenes que se normalizaron a los 224 pixeles con sus 3 respectivos canales, por último, se configuro el bloque de capas que tiene la arquitectura dejándolo listo para su entrenamiento, a continuación, se muestra el código empleado.

```
# Lectura del modelo neuronal MobilenetV2, y configuración de las capas ocultas
baseModel = MobilenetV2 (weights=''imagenet'', inclue_top = False,
input_tensor=Input (shape= (224,224,3)))
```

2.4.8.1 Construcción de la Arquitectura en la cabecera del modelo base

Al reentrenar una red neuronal suele surgir ciertos inconvenientes en la parte de su arquitectura, específicamente en las capas iniciales o comúnmente llamada cabecera del modelo, donde se ingresa la imagen ya convertida en vector ocasionando problemas de aprendizaje al pasar por las capas subsiguientes ya que el modelo fue entrenado con otros parámetros como las dimensiones de las imágenes, su función de activación etc. Para evitar estos inconvenientes se acondiciono la cabecera con los requerimientos necesarios para el aprendizaje.

Este método se lo conoce como ajuste fino y requiere que se realice una "cirugía de la red", haciendo una analogía lo que se realiza es que se corta el conjunto final de capas completamente conectadas (es decir, la "cabeza" de la red) donde se devuelven las predicciones a las capas posteriores. Para luego proceder a reemplazar la cabeza con un nuevo conjunto de capas completamente conectadas con los requerimientos necesarios, programados para inicializar el

modelo. A partir de esto, las capas por debajo de la cabeza mantienen sus pesos para que puedan actualizarse.

Para hacer esto lo primero que se realizo es configurar la variable en donde se guardó el modelo pre entrenado como una variable tipo output o salida, a través de la siguiente línea de código *modelo_cabezera = basemodel.output* para de esta manera poder modificar y construir la nueva cabecera del modelo.

El segundo comando que se empleó fue *AveragePooling2D* el cual resume la presencia de características en una imagen de entrada, para extraer de mejor manera estas características se usó el método de pooling o agrupación. La función de esta capa de agrupación es reducir la resolución del mapa de características que se obtiene de la imagen original, pero conservando sus características más relevantes para la clasificación mediante variantes de traslación. El filtro de capa de agrupación o pooling más común es de tamaño 2x2 con un dimensionamiento de 7x7 y la razón por la que se asignó de esta manera es porque es la más adecuada para la detección de características que permitieron interpretar en el modelo rasgos faciales como ojos, nariz, boca, cejas , como se muestra en la figura 24-2 y en base a eso decidir si estas características son un rostro humano con mascarilla o sin mascarilla, para realizar esto se usó la siguiente línea de código *modelo_cabezera = AveragePooling2D(pool_size=(7,7))(modelo_cabezera).*

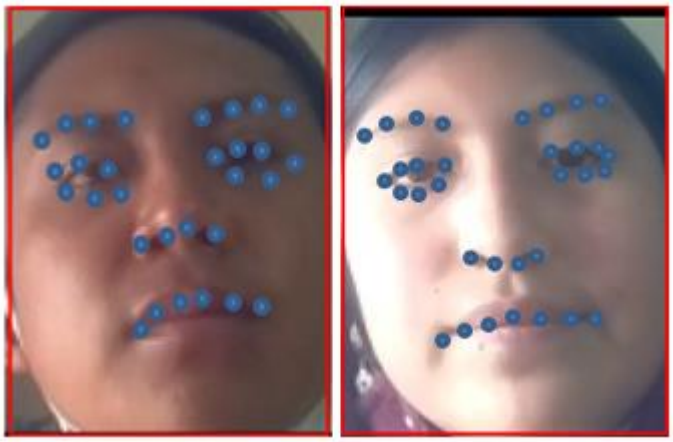

Figura 24-2. Patrones significativos de un rostro.

Realizado por: Paullán, A. 2021

Al hacer un proceso de pooling o agrupación lo que se busco es básicamente reducir el tamaño de los datos capa tras capa del modelo, pero manteniendo la información relevante, este proceso da como resultado una matriz, pero al ingresar a la siguiente capa esa matriz debe sectorizarse debido a que las redes neuronales de las capas subsiguientes en su entrada lo hacen de forma vectorial, para realizar este proceso se lo realiza mediante el comando *flatten*, que no es nada más que un

aplanamiento y que tiene mucho sentido ya que se va a aplanar dicha matriz resultante del pooling. La figura 25-2 muestra la representación gráfica del proceso de aplanamiento o Flatten.

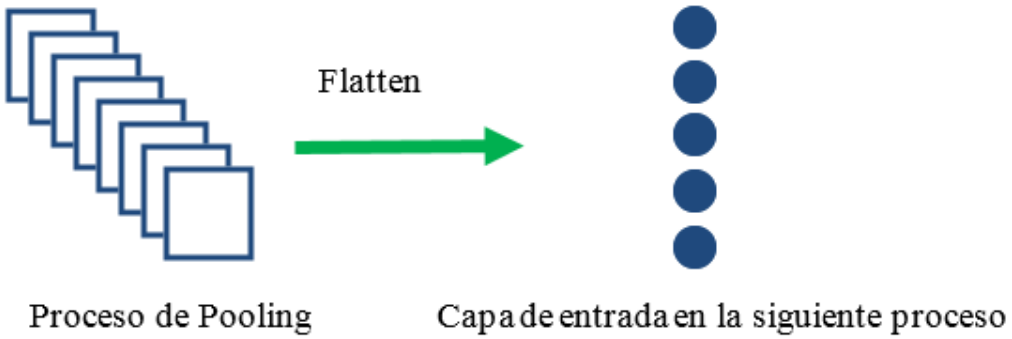

Figura 25-2. Representación gráfica del proceso Flatten.

Realizado por: Paullán, A. 2021

La función que se usó para implementar el *flatten* en el programa tiene el mismo nombre, cuyos parámetros de entrada son el proceso que se desea implantar y la variable en donde se realizara el proceso, para ello se usó la siguiente línea de código *modelo_cabezera= Flatten(name="flatten'‘) (modelo_cabezera)*

Por último, se configuro la función de activación, que no es nada más que aquella que activa los pesos de las neuronas, permitiendo insertar la no linealidad en la red neuronal. Para este modelo de red neuronal convolucional se usó la función de activación *relu* a través de la siguiente línea de código modelo_cabezera=Dense (128, activation= "relu'‘) (modelo_cabezera), logrando un mejor rendimiento.

2.4.8.2 Compilación del modelo de Red Neuronal

Luego de haber acondicionado la cabecera con los requerimientos iniciales necesarios, se procedió a compilar el modelo, pero antes de ello se procedió a especificar ciertos parámetros como disminuir la taza de aprendizaje y entropía cruzada binaria, que es nada más que un valor de diferencia entre dos distribuciones de probabilidad para una determinada variable aleatoria o para un conjunto de eventos. Para optimizar el modelo durante el entrenamiento se usó la función de costos que es la que permitió disminuir el error a la salida del modelo entrenado, mientras la función de costo tienda a cero menor será la perdida y el modelo será mejor.

La pérdida de entropía cruzada es una función de costo importante, que se usa para optimizar modelos de clasificación, en este caso se usó el algoritmo de optimización *Adam,* cuyos parámetros de entrada son la tasa de aprendizaje, el número de épocas y el tamaño del lote,

parámetros que fueron asignados anteriormente, a continuación, la figura 26-2 muestra la compilación del modelo.

```
# compilavcion del modelo
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
        metrics=["accuracy"])

# entrenamiento del modelo
print("[INFO] training head...")
H = model.fit(
        aumento.flow(trainX, trainY, batch_size=BS),
        steps_per_epoch=len(trainX) // BS,
        validation_data=(testX, testY),
        validation_steps=len(testX) // BS,
        epochs=EPOCHS)
```

Figura 26-2. Compilación del modelo de red neuronal.

Realizado por: Paullán, A. 2021

2.4.8.3 Evaluación y validación del entrenamiento

La pérdida en Deep Learning es una penalidad por una predicción incorrecta, es decir es un valor numérico que indica cuan incorrecta fue la predicción del modelo implementado. Si la predicción del modelo es perfecta la perdida es cero, de lo contrario la perdida será mayor a cero, este modelo se entrenó implementando el descenso de gradiente por lo que se requirió usar una función de pérdidas al diseñar y configurar el modelo.

Para la evaluación del modelo se tomó en consideración ciertos parámetros, como la pérdida de entrenamiento, que es nada más que el error en el conjunto de datos de entrenamiento a la cual se le asignó la variable *train_loss*, la pérdida de validación que es el error que se generó luego de haber realizado la validación de datos a través de la red neuronal entrenada asignándole la variable *val_loss.* Otro paramento que se considero fue la precisión de entrenamiento, que se la obtuvo aplicando a los datos de entrenamiento, en este caso al 80% de la base de datos que anteriormente habíamos separado para este proceso, asignándole la variable *train_acc*. Mientras que para realizar la validación de la precisión se realizado con el 20% de las imágenes de la base de datos, asignándole la variable val_acc. Estas variables se usaron para visualizar el resultado del entrenamiento tomando como eje x las épocas y como eje y la precisión y las perdidas. La figura 27-2 muestra la respuesta que se obtuvo de la etapa de entrenamiento, así como también se puede observar la presencia de pequeños sobreajustes con la pérdida de validación menor que la pérdida de entrenamiento. Además, la gráfica de la derecha muestra la precisión que se obtuvo conforme se iba iterando en las épocas respectivas dando como resultado una precisión del 99%.

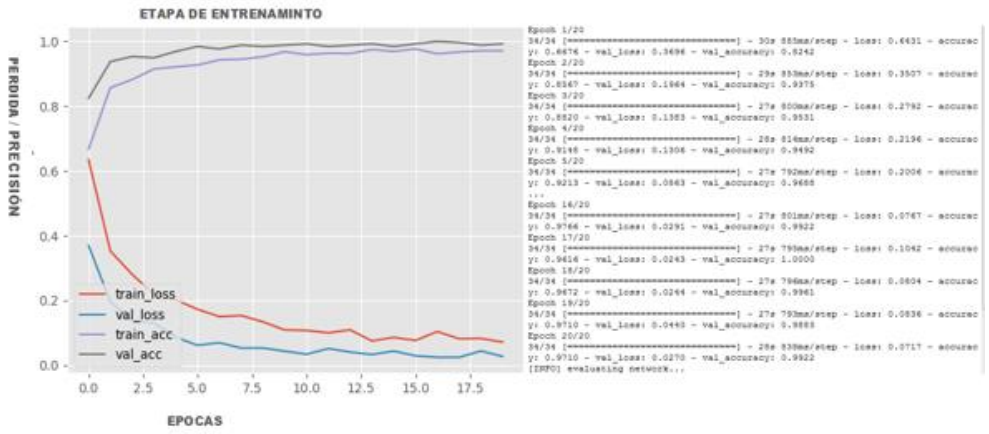

Figura 27-2. Respuesta de precisión en la etapa de entrenamiento

Realizado por: Paullán, A. 2021

2.4.9 Programa principal para deteccción de mascarilla y temperatura

En el grafico 3-3 se muestra el diagrama de flujo en el cual se detalla la secuencia de pasos para el proceso de detección de mascarilla y temperatura.

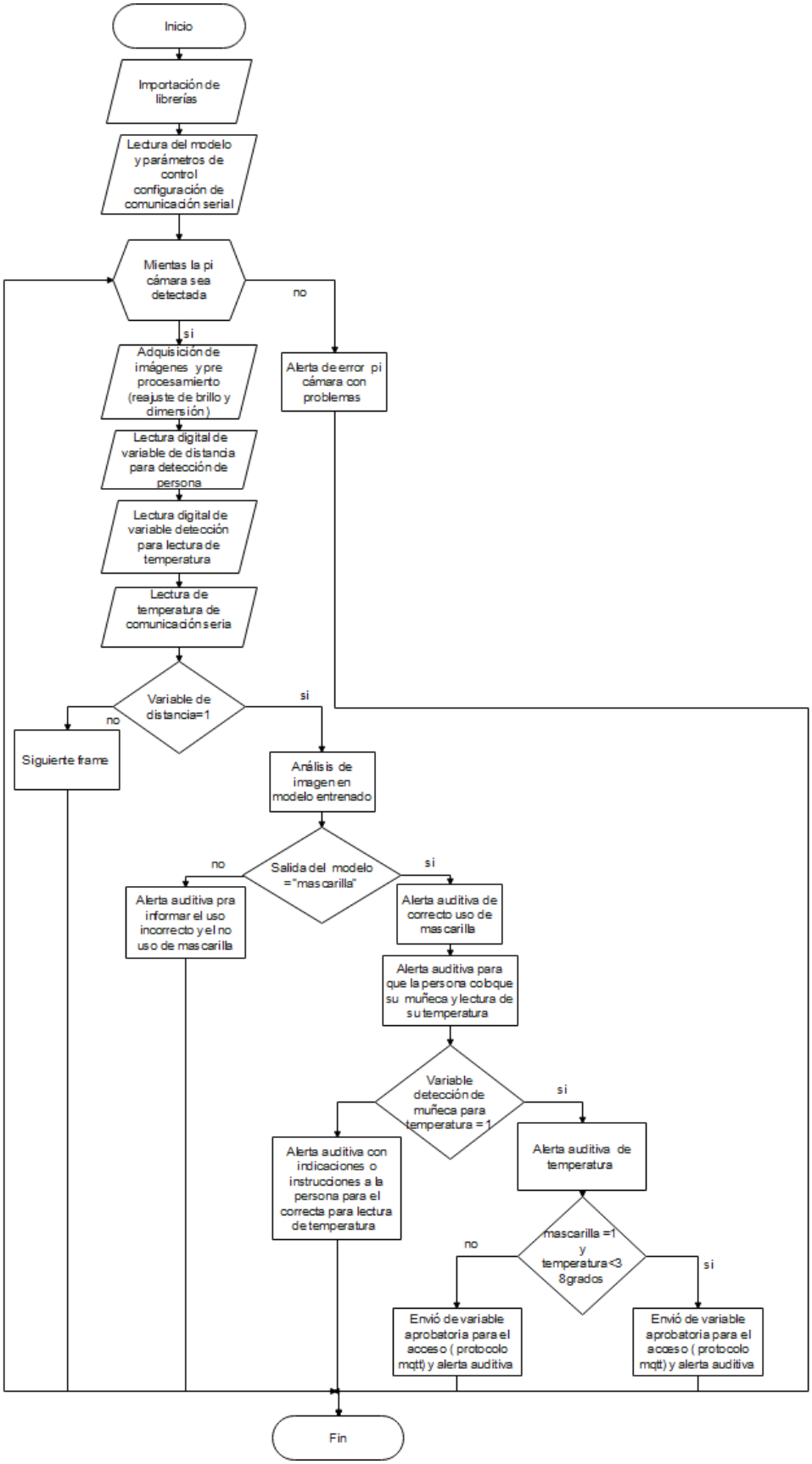

Gráfico 3-3. Diagrama de flujo para detección de mascarilla y temperatura

Realizado por: Paullán, A. 2021

Una vez finalizado el proceso de entrenamiento se procedió a realizar un programa donde se podrá visualizar el resultado del entrenamiento para la detección de mascarilla en diferentes rostros y a diferentes personas, para ello se procedió a importar las librerías necesarias para garantizar el funcionamiento de la red, así como también su correcta lectura e interpretación de los datos de salida. Las librerías que se importaron fueron las mismas que se usaron en el entrenamiento, pero omitiendo algunas de ellas. A continuación, la figura 28-2 muestra las librerías importadas.

```
import cv2
Fromtensorflow.keras. applications. mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import im_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os
import datetime
import time
```

Figura 28-2. Librerías importadas para el funcionamiento de la red

Realizado por: Paullán, A. 2021

También se declararon las variables de control para el sensor de proximidad y el sensor de temperatura, esta última será totalmente independiente a la detección de mascarilla, aunque trabaja de forma secuencial ya que después de haber realizado la detección de mascarilla se procederá a realizar la lectura de temperatura, para ello se hizo uso de un sensor de proximidad de corto alcance para poder detectar la presencia, en este caso se detecta la muñeca de la persona a través del cual el sensor de temperatura hace la lectura, en el programa esta se configuro como una variable de entrada digital, como detección de flanco de subida es decir que se detectó cuando existió una transición de un cero a un uno lógico, las siguientes líneas muestra el código empleado.

```
GPIO.add_event_detect(4, GPIO.RISING, callback=sensor:deeccion_mascarilla)
GPIO.add_event_detect(2, GPIO.RISING, callback=sensor:deeccion_muñeca)
```

Después de configurar los terminales como entradas digitales, lo siguiente es abrir el modelo de entrenamiento, así como también los pesos y la arquitectura de la red. Para ello se hizo usó del comando *cv2dnn.readNet* que es propio de OpenCV y *load_model*, así como también de TensorFlow y Keras, se inician variables auxiliares que controlan el reconocimiento, a continuación, se muestra el código empleado.

```
print ("lectura del modelo")
```

facenet = cv2.dnn.readNet("deploy. prototxt","res10_300x300_ssd_iter_140000.caffemodel')
maskNet = load_model ("mask_detector.model")
cont = 0
value = 0
deteccion_flanco = 0
variable_detecion_mascarilla = 0
detección_rostro = 0
conmutador_de_camara = 0
diferenciat = 1

Al tener las variables listas para abrir el modelo lo siguiente es acceder al dispositivo de captura en este caso el de la Raspberry Pi para ello se inicializo el dispositivo y se lo hace de una forma sencilla accediendo y tomando capturas con solo una estructura de iteración *for* que accede y toma una imagen en una constante de tiempo para ser analizada y si el sensor de proximidad se activa será procesado dentro del modelo de entrenamiento, a continuación se muestra el código empleado.

for frame in camera. capture_continuous (rawCapture,Format ="bgr", use_video_port=True):

variable_detecion_mascarilla=0

Button=GPIO.input(7)

Estado_actual=button

GPIO,output(11,1)

El programa empieza condicionando el correcto reconocimiento del dispositivo de captura de imágenes, ingresando al modelo de entrenamiento como una entrada digital proveniente de la tarjeta Arduino Uno como dispositivo de adquisición de datos del sensor ultrasónico, programado en un rango efectivo de detección de 65 a 80 cm. Luego de ello el Arduino Uno envía una salida digital que pasa por la tarjeta de acondicionamiento de voltaje para finalmente ser leída como entrada digital por la Raspberry Pi y el cambio de estado de 0 a 1 es decir un flanco de subida hará el proceso de detección de mascarilla. Una vez que el modelo ya tiene el resultado del proceso de detección de mascarilla, este resultado es expresado como una variable tipo string o texto tiendo dos valores “no mascarilla” y “mascarilla” en base a esto si la variable es “no mascarilla” se alerta mediante un mensaje de audio, que comunica que la persona no tiene mascarilla o la tiene colocada de manera incorrecta.

En cambio cuando el modelo de como resultado “mascarilla” quiere decir que la persona tiene correctamente colocada la mascarilla en su rostro, procediendo a realizar el proceso de control de

temperatura, esta lectura siempre se la hará desde el principio del programa es decir que la lectura de la temperatura mediante la comunicación serial no estará condicionada a ningún parámetro, la razón por la que realiza esto es porque dentro de la programación de Arduino Uno existe un retardo de 200 ms cuando se realiza la lectura del sensor de temperatura para establecer una correcta comunicación serial con la tarjeta. Al realizar la lectura de temperatura en la programación de Python en la Raspberry Pi ese retardo produce un falso positivo, motivo por el cual la lectura de la temperatura se la realiza fuera de toda condición. Cuando el sensor de proximidad detecta presencia realiza un cambiado de un cero lógico a uno lógico para realizar una correcta detección de temperatura, en caso de que el sensor no haya realizado la detección se reproduce una alerta auditiva con instrucciones para que la persona coloque correctamente su muñeca en la ranura del dispositivo. La figura 29-2 muestra el resultado final del algoritmo, dando como resultado la correcta detección de mascarilla.

Figura 29-2. Resultados respecto a la detección del rostro y mascarilla

Realizado por: Paullán, A. 2021

2.4.10 Algoritmo para la etapa de procesamiento de imágenes.

Antes de ingresar las imágenes para que estas sean procesadas por el modelo de Deep Learning mediante la arquitectura MobileNetV2 se debe realizar el algoritmo de procesamiento de imagen, para ello se realizó dos pasos previos que se pueden observar en el grafico 4-3 a través de un diagrama de flujo.

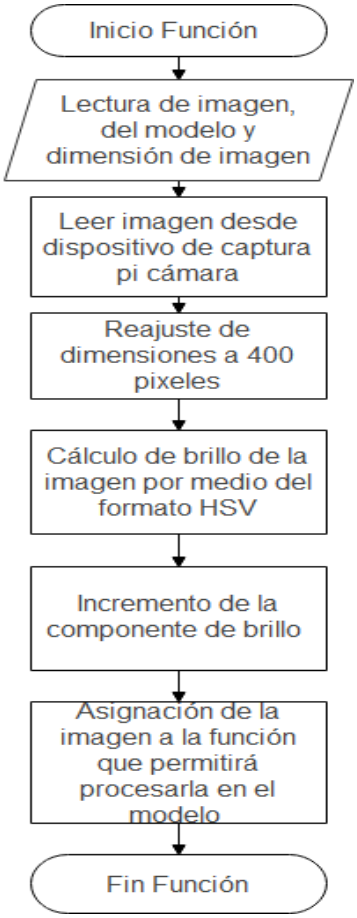

Gráfico 4-3. Diagrama de flujo para el procesamiento de imágenes

Realizado por: Paullán, A. 2021

En primer lugar, se realiza la adquisición de la imagen, estas imágenes son adquiridas por medio de la Cámara para Raspberry Pi, una vez realizado esto se procede a realizar un *resize* o un ajuste de dimensiones a una escala de 400 pixeles, esto se lo realizo usando una de las funciones de la librería *imutils* la cual permite realizar un escalonamiento de la imagen a la dimensión que se necesite. En la figura 30-2 se puede observar la imagen procesada con el comando *resize* y la imagen adquirida. Una de las desventajas de este proceso es que al reducir las dimensiones se pierden ciertos detalles en patrones que conforman un rostro humano cuando se tiene una imagen completa, pero esto es necesario debido a que el modelo debe tener como entrada una imagen con una dimensión especifica.

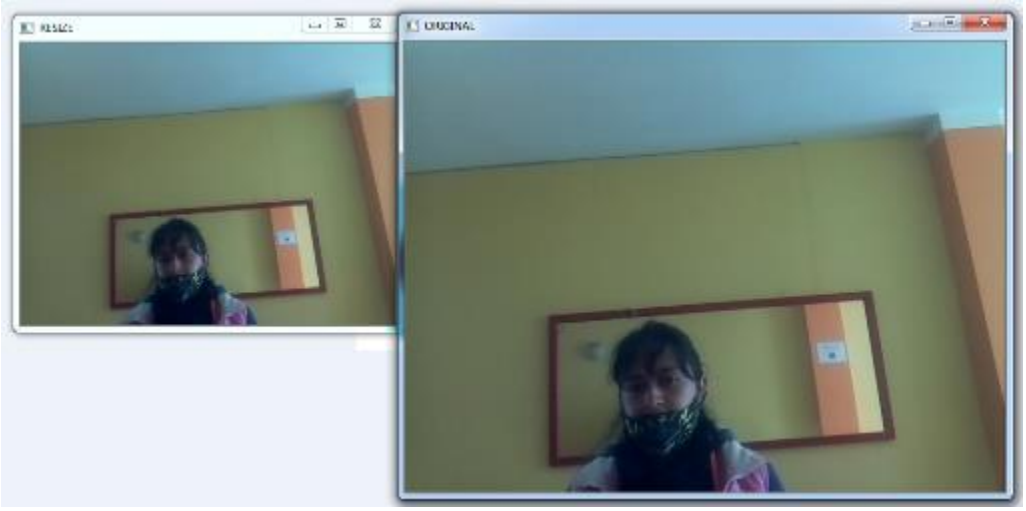

Figura 30-2. Imagen adquirida

Realizado por: Paullán, A. 2021

Al tener las imágenes con las dimensiones necesarias, el siguiente paso que se realizo fue compensar la falta o exceso de iluminación en las imágenes adquiridas, para ello se procedió a ajustar la componente de brillo mediante el uso de la técnica de segmentación de color HSV, que a diferencia del RGB que se basa en los colores primarios, este formato se basa en el tono, saturación y brillo y así de esta manera aclarar la imagen en caso de que existiera poca iluminación para así poder realizar una mejor detección mediante la identificación de los patrones.

Para poder obtener una imagen más clara se procedió a calcular la componente de brillo usando el formato HSV, esta componente es una matriz que está compuesta por valores que empiezan desde 0 a 255, de este rango se requiere tan solo un valor, para lo cual se procedió a calcular la media de todos los valores que la componen. Para luego compararla en base a una constante acondicionada como un valor aceptable de iluminación, en base a este valor se realizó un incremento o decremento de la componente de brillo para luego volverse a unir y regresar a su formato original RGB, obteniendo una imagen más clara y así poder realizar una correcta detección de mascarilla. La figura 31-2 muestra la imagen adquirida luego de haber aplicado la compensación de brillo.

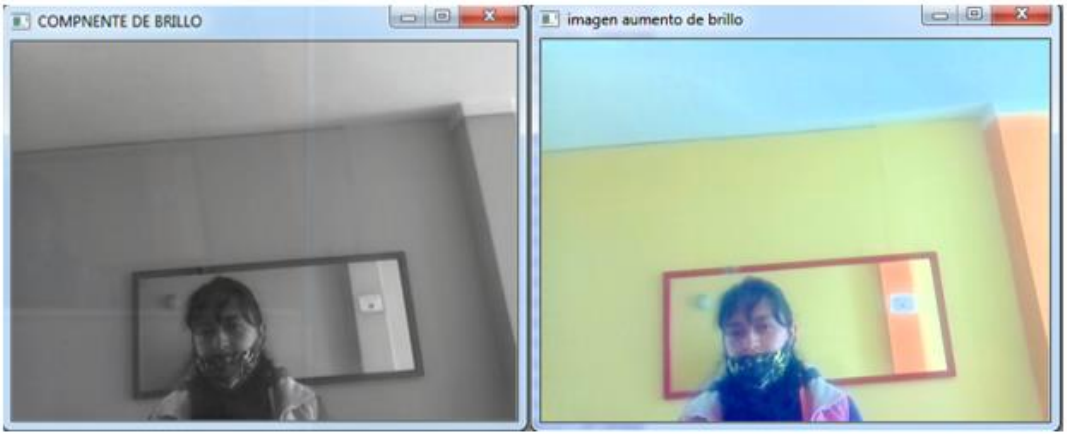

Figura 31-2. Imagen adquirida aplicando la compensación de brillo.

Realizado por: Paullán, A. 2021

2.4.11 Programa de adquisición de temperatura y distancia en Arduino

Para la adquisición de temperatura fue necesario utilizar una señal que permita realizar un procesamiento por cada frame de captura con el propósito de no sobrecargar el procesador de la tarjeta(Raspberry Pi), es decir que se realizó un procesamiento de reconocimiento, cada vez que el sensor ultrasónico detecta el parámetro de distancia el cual está dentro de un rango especifico y cuando esto sucede, manda una señal de control digital proveniente de la tarjeta Arduino Uno la cual pasa a una tarjeta de acondicionamiento y posteriormente a la Raspberry Pi. El grafico 5-3 muestra el diagrama de flujo para el control de temperatura.

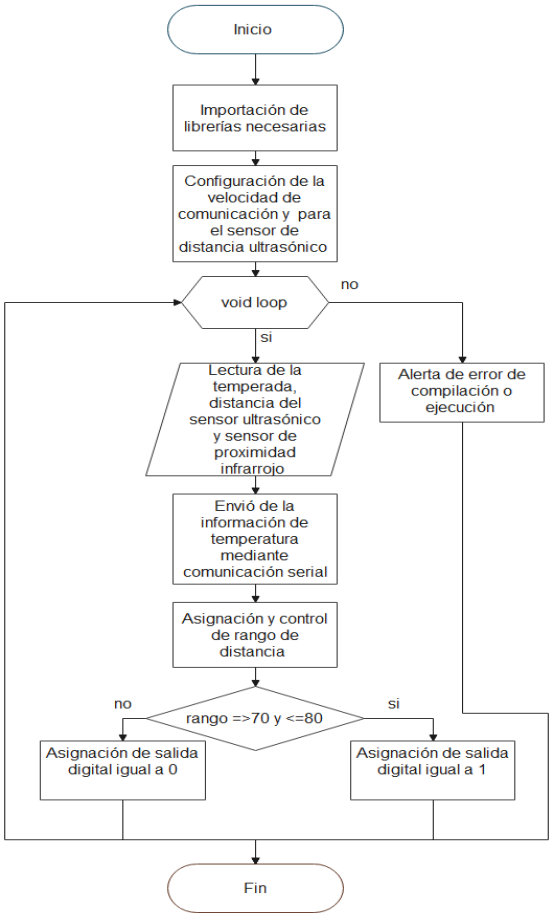

Gráfico 5-3. Diagrama de flujo para control de temperatura

Realizado por: Paullán, A., 2021

El programa inicia importando las librerías necesarias para realizar la lectura del sensor infrarrojo, esto se lo realiza través del comando *Adafruit_MLX90614*, luego se importa la librería *Wire.h* que es la que sirve para configurar la comunicación serial necesaria para enviar el dato de temperatura a la Raspberry Pi. La comunicación serial se asigna declarando la velocidad en baudios en este caso será de 9600 baudios, no es necesario una mayor velocidad ya que solo se enviará un dato de forma consecutiva, y esto se lo realizo dentro de la función *setup*, a través de las siguientes líneas de código.

Serial. Begin (9600).

pin Mode (10, OUTPUT).

pinMode(Trigger, OUTPUT);

PinMode(Echo, INPUT);

DigitalWrite(Trigger, LOW);

Luego de haber realizado la importación de las librerías se procedió a realizar la lectura de la temperatura con la función *termometroIR.readAmbientTempC*, la cual hace la adquisición del sensor y posteriormente su acondicionamiento en grados centígrados. Una vez realizado esto la temperatura es leída por el Arduino Uno a través del sensor de temperatura y enviada a la Raspberry Pi por medio de una comunicación serial de forma constante, además de una señal de control que será proporcionada por un sensor infrarrojo de presencia cuando esta señal pase de 0 a 1 en una transición de flanco de subida activará una señal digital configurada previamente en el Arduino Uno. Al colocar la persona su muñeca bajo la cavidad correspondiente el sensor de presencia hará una transición de cero lógico (cero voltios) a uno lógico (cinco voltios) esa señal es enviada a la tarjeta de acondicionamiento de voltajes que pasará esos 5 voltios a 3.3 voltios que es el voltaje al que trabaja la Raspberry Pi, en ese momento se activa una salida digital para que la Raspberry Pi procese el dato de temperatura y lo analice. La figura 32-2 se puede observar los sensores de temperatura y presencia acondicionados al prototipo.

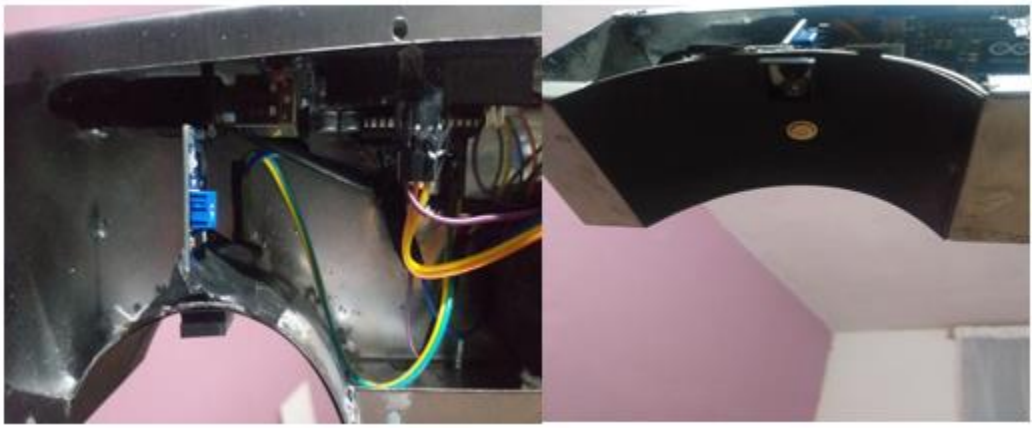

Figura 32-2. Identificación del sensor de temperatura y el de presencia

Realizado por: Paullán, A. 2021

2.4.12 Protocolo de comunicación para control de acceso mediante verificación de usuarios que no cumplan con el uso de mascarilla.

La normativa a la que se debe sujetar el personal de la empresa PAUFIT, considera que cada empleado debe cumplir el uso obligatoria de la mascarilla y que los niveles de temperatura corporal no deben pasar los 37.5 grados centígrados, donde para el cumplimento de estos requerimiento se implanto un conjunto de reglas que tienen como fin el controlar el ingreso de los usuarios mediante el uso correcto de mascarilla, donde se realiza un proceso secuencial el cual le impide el acceso al usuario que no cumpla con él uso de mascarilla, el cual lo hará por medio de mensajes. Para ello en el diseño del protocoló de comunicación se consideró la

implementación de 3 canales los cuales transcurren en la comunicación entre el dispositivo y el usuario.

- Uno de los primeros canales de comunicación entre el dispositivo y el usuario, fue la emisión de una alerta de tipo auditiva en función de un audio que alerta del no uso de la mascarilla y la negación del acceso, así mismo se generó una alerta auditiva que muestra la detección correcta de la mascarilla y el paso al siguiente requerimiento para poder acceder a su lugar de trabajo, es decir la orden para la lectura de la temperatura y finalmente una alerta del uso incorrecto de mascarilla.

- Se implementó una alerta de tipo visual, por medio de una luz led que toma un color rojo para alertar del no uso de mascarilla y cuando la temperatura es alta, toma un color verde cuando se detecta la mascarilla y cuando la temperatura se encuentra entre los rangos normales y una luz azul cuando se hace un mal uso de la mascarilla tal como se puede observar en la figura 33-2.

- También se implementó un protocolo de comunicación a través de una notificación al encargado del ingreso de los trabajadores, el cual consiste en la captura de la imagen de la persona que no usa mascarilla, la cual la envía mediante un mensaje para que el encargado tenga conocimiento del no cumplimento de la normativa.

Para él envió de la notificación se lo realizó mediante la aplicación de whatsapp, para lo cual se importó la librería pip install pywhatkit, además para poder realizar la comunicación se usó la función *sendwhatmsg* en la cual se debe colocar el número telefónico de la persona a quien va dirigido (incluyendo el código de país) y para finalmente enviar la imagen de la persona sin mascarilla. También se consideró enviar la imagen a través del uso de correo electrónico, lo cual consistió en crear un objeto SMTP para la conexión, luego de ello se procedió a ingresar a la cuenta, definir los encabezados de los mensajes y las credenciales de inicio de sesión, luego se creó un objeto de mensaje a través del comando *MIMEMultipart*, para así adjuntar la imagen al mensaje a través del comandoMIMEMultipart y poder finalmente enviar el mensaje.

Figura 33-2. Comunicación con el usuario - Alerta visual

Realizado por: Paullán, A. 2021

Una vez que los usuarios realizaron todos los pasos y habiendo pasado todos los filtros se procedió a otorgar a los usuarios el derecho para poder acceder a su lugar de trabajo, en tanto que se impide el acceso a los usuarios que no cumplen con los requerimientos. Para ello se determinó el uso de la comunicación M2M, la cual por medio del protocolo MQTT se realizó la apertura de la puerta y así poder precautelar la salud de todas las personas que trabajan en la empresa PAUFIT.

2.4.11.1 Protocolo MQQT

Una vez ejecutada la lectura de temperatura y realizada la verificación de uso de mascarilla se envió estos datos por medio del protocolo MQTT al módulo ESP8266, el cual acciona un módulo de relés para poder darle acceso mediante la verificación de temperatura y uso de mascarilla. La comunicación es simple, si el resultado del modelo de entrenamiento de mascarilla y la temperatura está por debajo de los 38 grados centígrados se enviará una "A" como mensaje al módulo para comunicarle que puede darle acceso a la persona, esto acompañado de la reproducción de un mensaje de audio indicando que paso los controles requeridos.

2.4.11.2 Implementación del protocolo MQTT en el proceso de trasmisión

En primer lugar se importó todas las librerías necesarias a través del código *import paho.mqtt.client as mqtt* el cual permite acceder a todas las funciones de MQTT, para la configuración de la dirección ip, el id y el password de la conexión se lo hizo a través de la función *void setup*, también se configuró el puerto del servidor que será el 1883 que es el puerto del protocolo MQTT, el siguiente paso es crear un objeto que será el cliente MQTT. Este cliente tiene una identificación única, esto se lo hizo usando el método *mqtt. Client ()*, a través del siguiente código *cliente= mqtt. Client ("makerio_mqtt")*

Lo siguiente que se hizo fue que el cliente se conecte a un broker (o servidor) en MQTT, configurando el puerto de conexión e inicializándolo con la función *start*, todo esto se lo realiza a través de las siguientes líneas de código.

host: test.mosquitto.org ip: 1883

cliente.connect ("test.mosquitto.org", 1883)

mqttc.loop_start()

El método *publish* envía dos variables a través del protocolo MQTT, en el primer caso se envía una variable tipo char o texto que tendrá el valor "A", lo que significa que se debe dar acceso a la

persona que haya pasado el control por detección de mascarilla y temperatura. El segundo caso cuando se envía una variable que tendrá el valor de "N", significa que la persona no paso los controles mencionados y no se le podrá dar acceso, a continuación, se muestra el código empleado.

client.publish ("outTopic", "A")

client.publish ("outTopic", "N")

Luego de haber realizado esto se procedió a configurar el módulo ESP8266, a través del comando *PubSubClient,* que básicamente hace que el módulo sea un cliente del protocolo MQTT. El cual tiene la función de receptor y establece la conexión con la Raspberry Pi recibiendo la variable de control que será interpretada por este módulo, la cual activa una señal de control configurando una salida digital que sirvió para activar la tarjeta que contiene 2 relays los cuales accionan el actuador eléctrico de la puerta. Para realizar esto se declararon las variables digitales necesarias para el control de acceso en este caso será una que controlara la tarjeta de relays y estas cambiaran de estado de un 0 a 1 lógico en función de la condición que compara el mensaje recibido, si este mensaje tipo texto es una A se asigna un 1 lógico (5 voltios), y en caso que el mensaje sea una N se envía un 0 lógico (0 voltios) a la salida digital. Luego envía este voltaje a las tarjetas de relays para activar o no la puerta mediante un accionamiento eléctrico. La figura 33-2 muestra el proceso para el control de acceso mediante el protocolo MQTT.

Para finalizar en el programa se creó la función *void loop*, dentro de ella se llamó a tres funciones respectivamente. La primera es la función conexión que permite monitorear la conexión de forma continua en caso de darse algún inconveniente, la segunda función es reconexión que imprime un mensaje de error por medio de la comunicación serial, que fue previamente configurada y la cual reintentara restablecerse después de 5 segundos. Y por último la función callback, la cual recepta los datos para en base a estos realizar los cambios de estado de las variables digitales asignando un nivel alto o bajo.

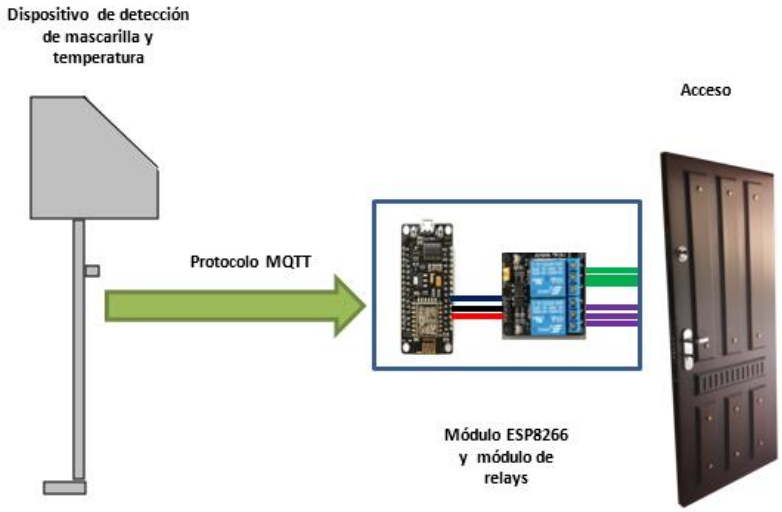

Figura 34-2. Representación gráfica de acceso

Realizado por: Paullán, A. 2021

2.4.11.3 Circuito de acondicionamiento de voltaje y amplificción de audio

Se diseñó un circuito de acondicionamiento para la conversión de las señales de 5 voltios a 3.3 voltios, para el funcionamiento de la Raspberry Pi y los sensores, para esto se utilizó optoacopladores, resistencias de valor alto para evitar que exista un mal funcionamiento en la tarjeta causado por la corriente. La figura 34-2 muestra el diseño del circuito de acondicionamiento de señal y amplificador de audio.

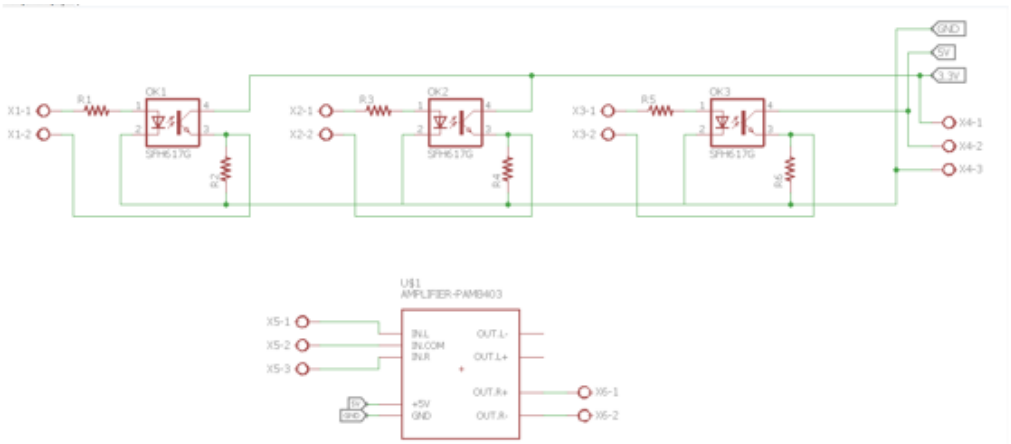

Figura 35-2. Circuito de acondicionamiento de señal

Realizado por: Paullán, A. 2021

Como se puede observar en la figura 34-2 existe 3 optoacopladores el primero se encarga de convertir la salida digital que tiene el Arduino Uno acondicionándolo a un rango de distancia de

3.3 voltios, la cual pasa a la Raspberry Pi como entrada digital para que haga el procesamiento de una sola captura de imagen, pre procesarla e ingresarla al modelo.

El segundo optoacoplador se encargó de la misma conversión de voltaje, pero a su entrada se asignó la salida del sensor de proximidad en donde la persona coloca su muñeca para realizar la lectura de temperatura corporal y el ultimo optoacoplador es un indicativo visual que mediante el uso de un led de alto brillo indica si la persona paso tanto la detección de mascarilla y temperatura de forma exitosa.

En esta tarjeta también se incorporó el amplificador de audio mediante el módulo PAM8403, que es un amplificador de 3 watts que trabajara en conjunto con un parlante. Este módulo tiene como entrada un voltaje de alimentación de 5 voltios y una entrada de audio proveniente de la Raspberry Pi.

2.4.12 Diseño de la estructura del prototipo en SolidWorks

Para el diseño estructural del prototipo, se realizó previamente en SolidWork con una leve inclinación en el plano de la cámara para aumentar el rango de visión, el diseño de esta estructura será lo más funcional y práctica posible para una correcta instalación, para la lectura de la temperatura corporal se dispone de un brazo extensor en donde la persona coloca su muñeca para la toma de temperatura, siendo un dispositivo no invasivo para evitar cualquier tipo de contacto y contagio. La estructura tiene una distribución de cavidades para la cámara, un orificio rectangular el cual servirá para la pantalla, otro para el sensor de proximidad, adicionalmente se tiene 2 orificios en el brazo extensor los cuales será para la lectura de la temperatura. La figura 35-2 muestra el diseño del prototipo en 3D

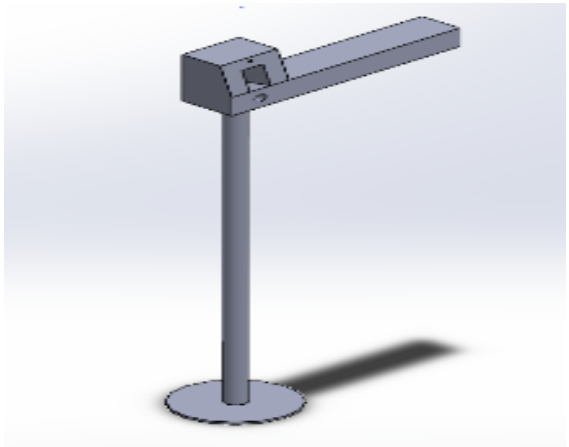

Figura 36-2. Diseño en SolidWork del Prototipo

Realizado por: Paullán, A. 2021

CAPÍTULO III

3. MARCO DE RESULTADOS

Se muestran las pruebas y resultados conseguidos del prototipo de detección de mascarilla y toma de temperatura con el personal que labora en la empresa PAUFIT

3.1 Validación del sistema

Luego de haberse realizado la implementación del prototipo, para la validación de este se establecen las pruebas y los resultados llevados a cabo con el prototipo para la detección de mascarilla y medición de temperatura.

3.1.1 Pruebas y resultados de funcionamiento para detectar de uso de mascarilla

Para realizar las pruebas de validación, el dispositivo se ubicó antes de la puerta de ingreso al taller de la empresa PAUFIT, este lugar posee una cubierta que protege al dispositivo, además para realzar las pruebas se tomaron en cuenta las siguientes observaciones.

- No pasar corriendo por el área de detección.
- Los usuarios deben ubicarse al frente de la cámara y mirar a la pantalla del prototipo.

Para las pruebas de detección de mascarilla se tomó en consideración cuatro escenarios experimentales, se realizó las pruebas con un grupo de 12 personas en diferentes horas del día, para las pruebas se consideró a personas con mascarilla, sin mascarilla y personas con distractores, estos distractores se consideraron a personas que estén usando una gorra o una bufanda en lugar de la mascarilla y también consideramos distractores cuando la persona tiene colocada de manera incorrecta la mascarilla.

Para ello se determinó el número de pruebas a realizar, donde se consideró la población el número de pruebas desde el punto de vista estadístico como infinito y a través de aplicar la fórmula para el tamaño de la muestra con un margen de error del 6% se obtuvo los siguientes datos. Aplicamos la fórmula:

$$\dot{n} = \frac{z^2 \text{pq}}{ME^2} \qquad \text{Ecu. 2}$$

Con ME = 0,06; p=0,5: q=0,5 y de acuerdo al margen de error le corresponde el valor Z=1,88, reemplazando tenemos:

$n = 245$, buscamos un múltiplo de 12 (pues son doce personas quienes participan en este estudio), más próximo a 245 y obtenemos 240, 240/12 = 20, como los momentos más adecuados

para hacer las pruebas son la mañana, medio día, media tarde y la noche (4 en total), se divide 20/4 =5 lo que me indica el número de veces que se debe realizar las pruebas. Las tablas 1-3, 2-3, 3-3 y 4 -3 muestran los datos obtenidos de las pruebas de detección de mascarilla realizadas en cada escenario.

Tabla 1-3: Pruebas de detección de mascarilla realizadas en la mañana

Escenario	No de pruebas	No de mascarillas	No de distractores	No sin mascarilla	Aciertos	Falla	Eficiencia (%)
Mañana	1	12	0	0	12	0	100
	2	11	1	0	12	0	100
	3	10	2	0	12	0	100
	4	8	1	3	11	1	92
	5	5	2	5	10	2	83
Total		46	6	8	57	3	475%
Eficiencia en Detección							95%

Realizado por: Paullán, A. 2021

La tabla 1-3 contiene los resultados de las pruebas realizadas respecto a la detección de mascarilla, donde se puede apreciar que existió un total de 57 aciertos, en las pruebas realizadas en la mañana, determinando así que el dispositivo fue eficiente en un 95%. La figura 1-3 muestra los resultados de las pruebas realizadas.

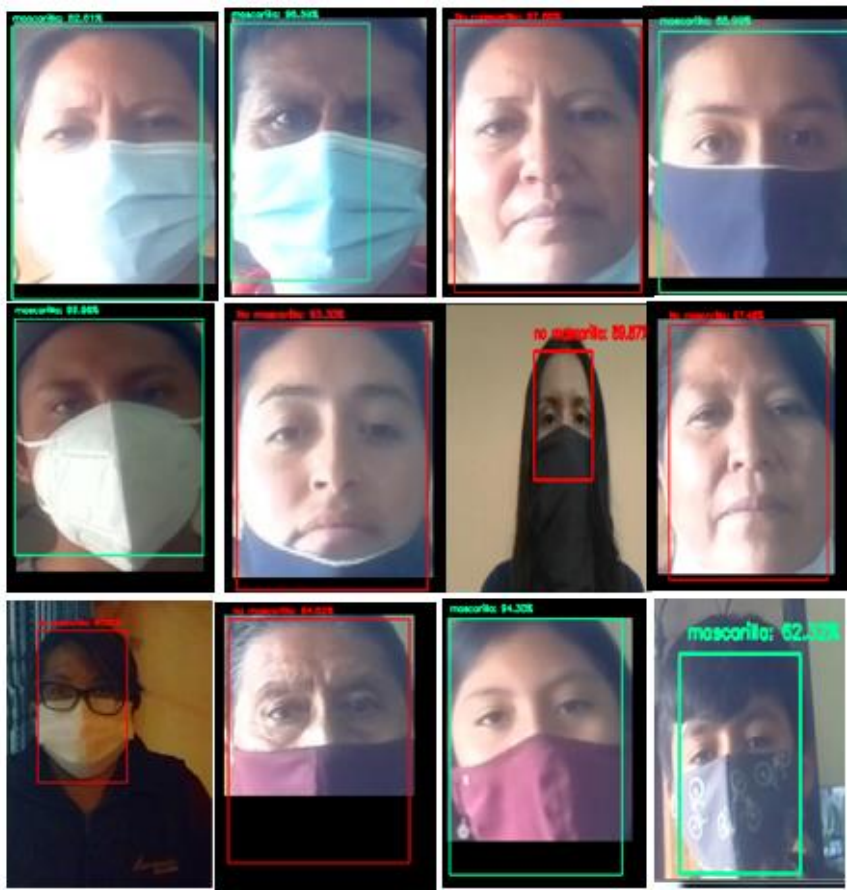

Figura 1-3. Resultados de la pruebas realizadas en la mañana

Realizado por: Paullán, A. 2021

Tabla 2-3: Pruebas de detección de mascarilla realizadas al medio día

Escenario	No de pruebas	No de mascarillas	No de distractores	No sin mascarilla	Aciertos	Falla	Eficiencia (%)
Medio Día	1	12	0	0	12	0	100
	2	8	0	4	12	0	100
	3	6	0	6	11	1	92
	4	10	2	0	10	2	83
	5	5	2	5	11	1	92
Total		41	4	15	56	3	467%
Eficiencia en Detección							93%

Realizado por: Paullán, A. 2021

La tabla 2-3 contiene las pruebas realizadas al medio día respecto a la detección de mascarilla, se evidencia un total de 56 aciertos, determinando así que el dispositivo es eficiente en un 93%. La figura 2-3 muestra los resultados de las pruebas realizadas.

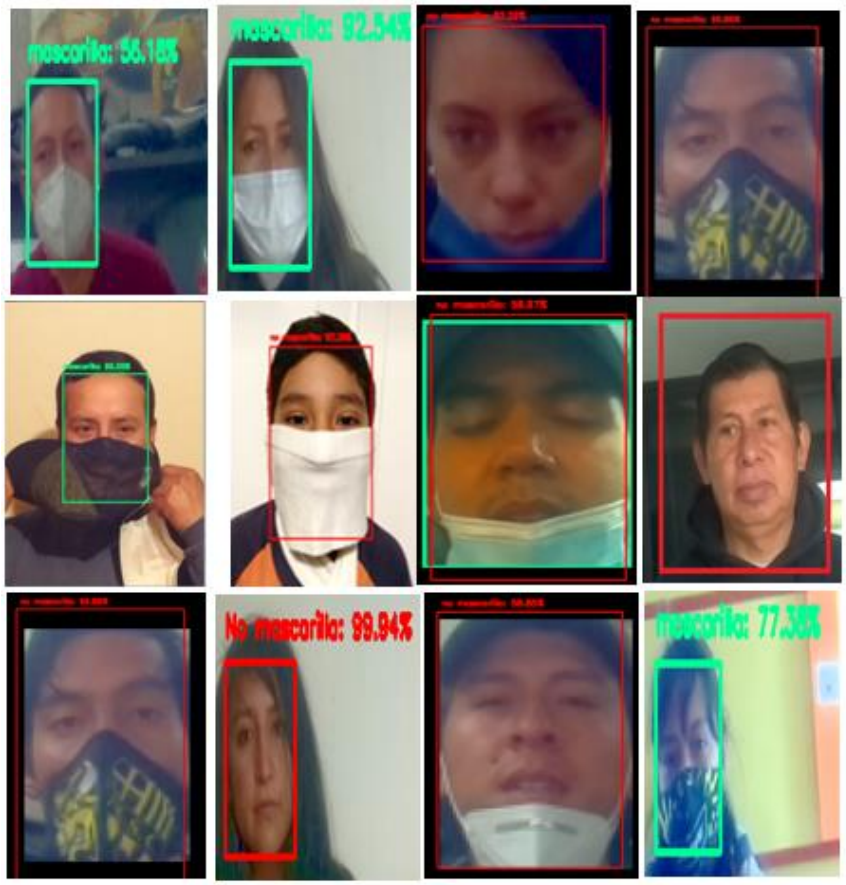

Figura 2-3. Resultados de pruebas realizadas al medio día

Realizado por: Paullán, A. 2021

Tabla 3-3: Pruebas de detección de mascarilla realizadas a la media tarde

Escenario	No de pruebas	No de mascarillas	No de distractores	No sin mascarilla	Aciertos	Falla	Eficiencia (%)
Media Tarde	1	12	0	0	12	0	100
	2	2	4	6	11	1	92
	3	4	0	8	11	1	92
	4	6	1	5	12	0	100
	5	7	0	5	12	0	100
Total		31	5	24	58	2	484%
Eficiencia en Detección							97%

Realizado por: Paullán, A. 2021

La tabla 3-3 contiene las pruebas realizadas a la media tarde respecto a la detección de mascarilla, se evidencia un total de 58 aciertos, determinando así que el dispositivo es eficiente en un 97%. La figura 3-3 muestra los resultados de las pruebas realizadas.

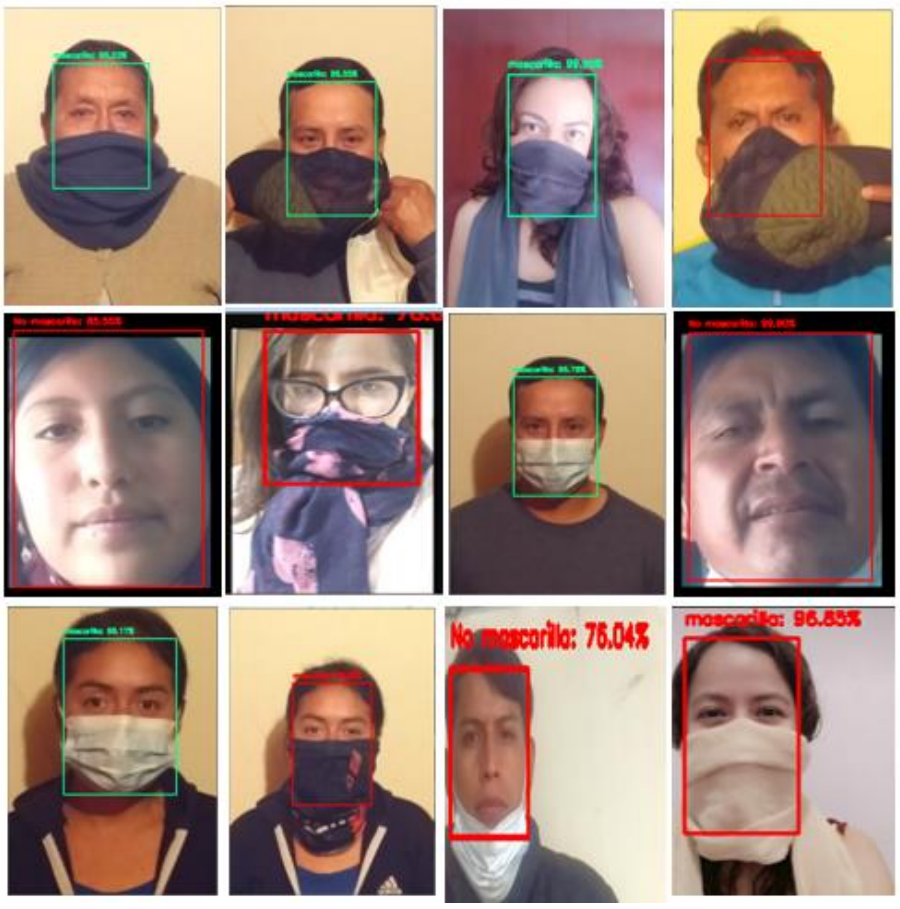

Figura 3-3. Resultado de pruebas realizadas a media tarde

Realizado por: Paullán, A. 2021

Tabla 4-3: Pruebas de detección de mascarilla realizadas en la noche

Escenario	No de pruebas	No de mascarillas	No de distractores	No sin mascarilla	Aciertos	Falla	Eficiencia (%)
Noche	1	12	0	0	11	1	92
	2	4	5	3	10	2	83
	3	7	0	5	11	1	92
	4	8	2	2	12	0	100
	5	9	0	3	11	1	92
Total		40	7	13	55	5	459%
Eficiencia en Detección							92%

Realizado por: Paullán, A. 2021

La tabla 4-3 contiene las pruebas realizadas en la noche respecto a la detección de mascarilla, se evidencia un total de 55 aciertos, determinando así que el dispositivo es eficiente en un 92%. La figura 4-3 muestra los resultados de las pruebas realizadas.

Figura 4-3. Resultado de pruebas realizadas en la noche

Realizado por: Paullán, A. 2021

La tabla 5-3 muestra el total de aciertos de detección que se obtuvo de las pruebas realizadas en diferentes escenarios, los cuales se muestra en el gráfico 6-3, determinando así un porcentaje general de eficiencia promedio del 94,25%.

Tabla 5-3: Resultados de las pruebas realizadas en los diferentes escenarios

	Aciertos	**% De eficiencia**	**% Error**
Pruebas en la mañana	57	95	5
Pruebas al medio día	56	93	7
Pruebas a media tarde	58	97	3
Pruebas en la noche	55	92	8
Total	226	377%	23%
Media Total		94,25%	5,75%

Realizado por: Paullán, A. 2021

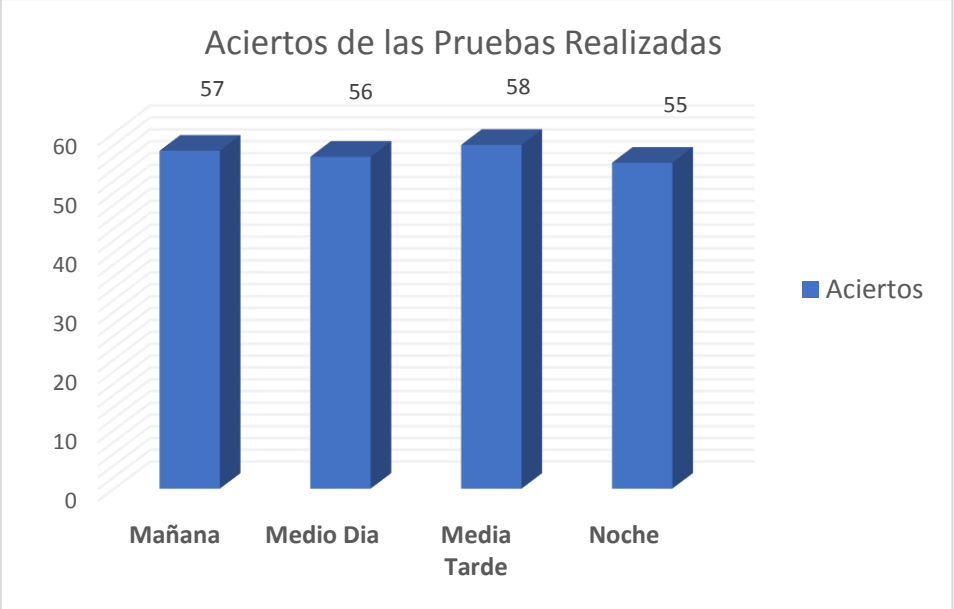

Gráfico 6-3. Número de aciertos de detección de mascarilla

Realizado por: Paullán, A., 2021

El gráfico 7-3 muestra la eficiencia promedio del dispositivo de detección de mascarilla

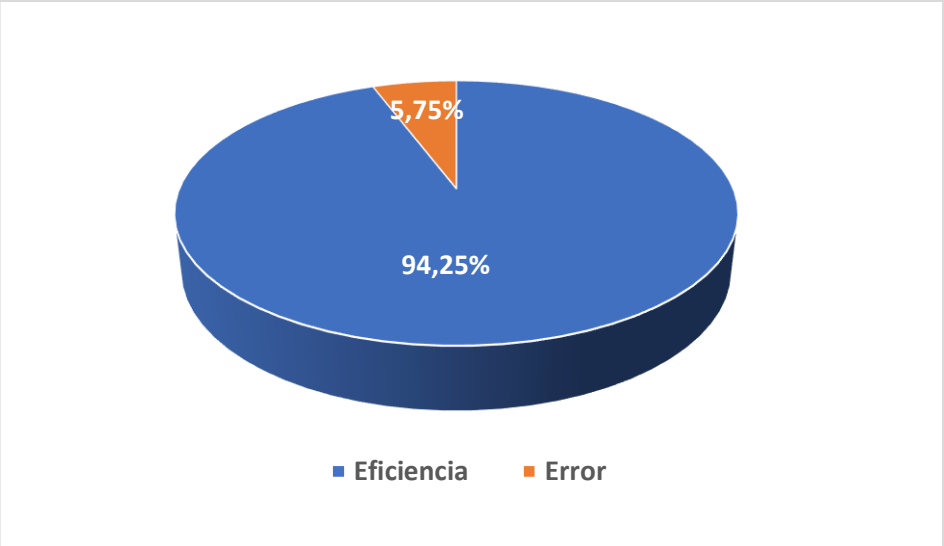

Gráfico 7-3. Eficiencia promedio del dispositivo

Realizado por: Paullán, A., 2021

Luego de haber realizado las pruebas de detección de mascarilla se procedió a realizar las pruebas para determinar desde que distancia el dispositivo empieza a realizar una detección efectiva, para ellos se tomó a un grupo de 5 personas para realizar pruebas a diferentes distancias como se puede observar en la tabla 6-3.

Tabla 6-3: Resultados de las pruebas realizadas para determinar la distancia adecuada

Pruebas a una distancia de 30 cm con respecto a la cámara						
No de pruebas	**No de mascarillas**	**No de no mascarillas**	**Detección mascarilla**	**Detección no mascarilla**	**% Eficiencia**	**% Error**
1	3	2	0	1	20	80
2	4	1	1	1	40	60
3	2	3	1	2	60	40
4	1	4	0	2	40	60
5	5	0	1	0	20	80
Total	15	10	3	6	180	320
Eficiencia de detección					36%	64%
Pruebas a una distancia de 50 cm con respecto a la cámara						
No de pruebas	**No de mascarillas**	**No de no mascarillas**	**Detección mascarilla**	**Detección no mascarilla**	**% Eficiencia**	**% Error**
1	3	2	3	2	100	0
2	4	1	3	1	80	20
3	1	4	1	4	100	0

4	2	3	2	3	100	0
5	0	5	0	5	100	0
Total	10	15	5	11	480	20
Eficiencia de detección					96%	4%
Pruebas a una distancia de 100 cm con respecto a la cámara						
No de pruebas	**No de mascarillas**	**No de no mascarillas**	**Detección mascarilla**	**Detección no mascarilla**	**% Eficiencia**	**% Error**
1	3	2	3	2	100	0
2	4	1	3	1	80	20
3	2	3	1	3	80	20
4	1	4	1	4	100	0
5	5	0	5	0	100	0
Total					460	40
Eficiencia de detección					92%	8%
Pruebas a una distancia de 150 cm con respecto a la cámara						
No de pruebas	**No de mascarillas**	**No de no mascarillas**	**Detección mascarilla**	**Detección no mascarilla**	**% Eficiencia**	**% Error**
1	3	2	3	2	100	0
2	4	1	4	1	100	0
3	1	4	1	4	100	0
4	2	3	1	3	80	20
5	0	5	0	5	100	0
Total					480	20
Eficiencia de detección					96%	4%

Realizado por: Paullán, A. 2021

Luego de haber realizado las pruebas de detección de mascarilla a 4 diferentes distancias, con un porcentaje del 94.6% de eficiencia, se determinó que la distancia de reconocimiento con respecto a la cámara es desde los 50cm a los 150cm. El gráfico 8-3 muestra los datos obtenidos.

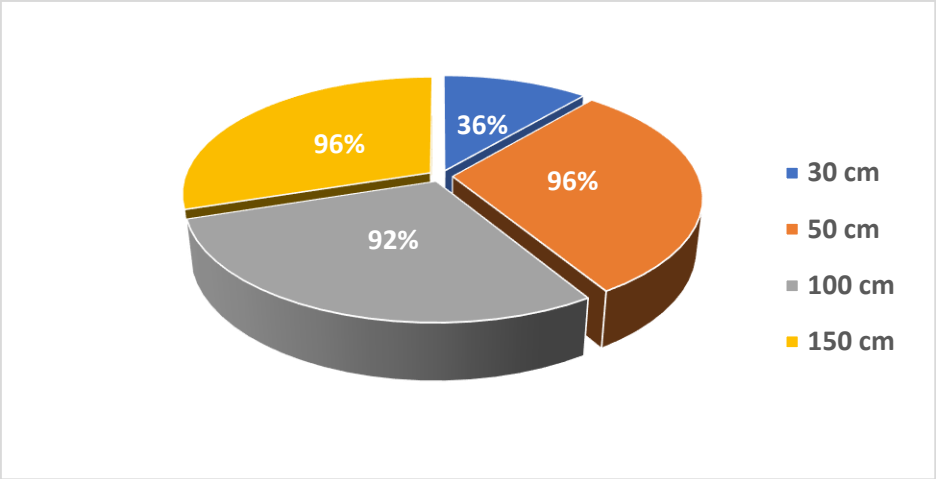

Gráfico 8-3. Eficiencia de detección para determinar la distancia de detección

Realizado por: Paullán, A., 2021

3.1.2 Pruebas y resultados de la medición de temperatura corporal

Para las pruebas de temperatura la persona coloco su muñeca bajo el brazo extensor del dispositivo sin tocar la estructura, ya que su diseño es precisamente para evitar cualquier contacto y contagio, se realizó 5 pruebas de toma de temperatura a las personas en 2 días distintos y dos escenarios diferentes, se realizó 2 mediciones, una de ellas fue tomar la temperatura corporal con el prototipo y la otra medición se realizó con un termómetro infrarrojo comercial de la marca Runfengte modelo SKU-178, el cual posee una resolución de 0.2[ºC]

Tabla 7-3: Resultados obtenidos por medición de temperatura día 1

Escenario	No de pruebas	Temperatura tomada con el dispositivo en grados centígrados(°C)	Temperatura tomada con el instrumento de medición en grados centígrados(°C)	Error absoluto
Mañana	1	36.9	36.8	0.10
	2	36.8	36.7	0.12
	3	36.5	36.7	0.16
	4	36.2	36.0	0.30
	5	37.3	37.0	0.38
Tarde	1	37.0	36.9	0.16
	2	36.9	36.7	0.18
	3	36.4	36.6	0.16
	4	37.3	37.0	0.38
	5	35.9	36.2	0.30
Media total				0.24

Realizado por: Paullán, A. 2021

La tabla 7-3 muestra los datos que se obtuvieron de la medición de temperatura, tanto las tomadas con el dispositivo propuesto en el presente trabajo, como las que se obtuvieron a través del instrumento de medición de temperatura corporal (termómetro digital infrarrojo). Dando como resultado un error de 0.24.

Tabla 8-3: Resultados obtenidos por medición de temperatura día 2

Escenario	**No de pruebas**	**Temperatura tomada con el dispositivo en grados centígrados(**°C)	**Temperatura tomada con el instrumento de medición en grados centígrados(**°C**)**	**Error absoluto**
Mañana	**1**	36,4	36,6	0,20
	2	36,3	36,6	0,34
	3	37,2	36,5	0,43
	4	36,9	37,0	0,16
	5	36,1	36,5	0,14
Tarde	**1**	36,12	36,78	0,36
	2	36,6	36,7	0,10
	3	37,3	37,0	0,38
	4	36,9	36,9	0,14
	5	37,0	36,9	0,16
Media Total				0,24

Realizado por: Paullán, A. 2021

La tabla 8-3 muestra los datos que se obtuvieron el segundo día que se realizó las pruebas de medición de temperatura, manteniendo los mismos parámetros mencionados anteriormente. Obteniendo como resultado que el equipo es capaz de estimar los valores de temperatura dentro del rango correspondiente de temperatura corporal y que el error en las medidas es menor a los 0.5 grados.

3.1.3 Análisis de tiempo de respuesta

El tiempo de respuesta es el tiempo que el dispositivo se tardó en realizar el proceso de detección, en este caso el tiempo que el dispositivo se demoró en detectar o no la mascarilla el cual se estimó en un promedio de 0.93 segundos por Frame de captura contando con la detección del sensor ultrasónico y en función del sensor de proximidad, este tiempo se establece en función de la Raspberry Pi que tiene un procesador con una velocidad mayor, además de una memoria RAM con más capacidad. . En cuanto a la detección de temperatura el tiempo promedio fue de 1,32

segundos con respecto a cambios de detección. Haciendo un promedio de tiempo de respuesta del dispositivo de 2.25 segundos en cuanto a detección de mascarilla y medición de temperatura.

3.1.3.1 Tiempo de respuesta para la detección de mascarilla y toma de temperatura

Para el análisis del tiempo de respuesta, se realizó 5 pruebas, en cada prueba se tomó 4 muestras, para determinar el tiempo que se demoró el dispositivo en detectar si la persona está o no usando la mascarilla, así como también el tiempo que se tardó en tomar la temperatura, los datos obtenidos se muestran en la tabla 9-3 y la figura 5-3 muestra lo datos recolectados de Python referente a los tiempos de respuesta.

Tabla 9-3: Resultados obtenidos por tiempos de respuesta

No de Pruebas	**Tiempo en segundos(s)**	
	Detección de mascarilla	**Toma de temperatura**
1	0,85	1,45
	0,78	1,43
	0,96	1,41
	0,85	1,37
2	0,82	1,12
	0,96	1,38
	0,93	1,12
	0,89	1,43
3	1,08	1,32
	1,09	1,44
	0,87	1,52
	1,02	1,41
4	0,92	1,42
	0,89	1,36
	1,07	1,12
	1,04	1,31
5	0,95	1,12
	0,88	1,4
	0,96	1,12
	0,98	1,28
Tiempo Promedio	0,9395	1,3265

Realizado por: Paullán, A. 2021

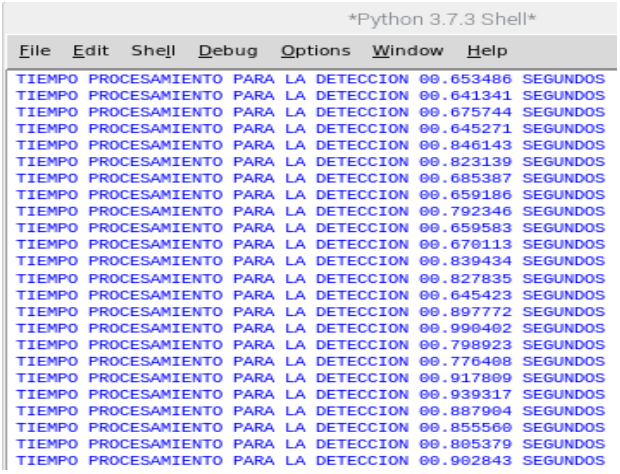

Figura 5-3. Datos obtenidos de Python respecto al tiempo de respuesta

Realizado por: Paullán, A. 2021

3.2 Consumo de energía del dispositivo

Se consideró realizar un análisis de consumo de energía del dispositivo para ello se tomaron lecturas del consumo de corriente de forma modular como se muestra en la tabla 10-3.

Tabla 20-3: Consumo en corriente del dispositivo

Dispositivo	Consumo(A)
Raspberry Pi con pantalla de 3.5 (Etapa de procesamiento)	0.95
Circuito amplificador y Acondicionamiento de señal. (Etapa de acondicionamiento)	0.123
Arduino Uno y circuito de detección de temperatura (Etapa de adquisición)	0.54
Total, Consumo (A)	1.61

Realizado por: Paullán, A. 2021

De las medidas realizadas para determinar el consumo total de energía del prototipo, el cual se realizó por etapas se obtuvo un consumo total de 1,61 A. El grafico 9-3 muestra los datos obtenidos del consumo de corriente, mediante un gráfico de barras.

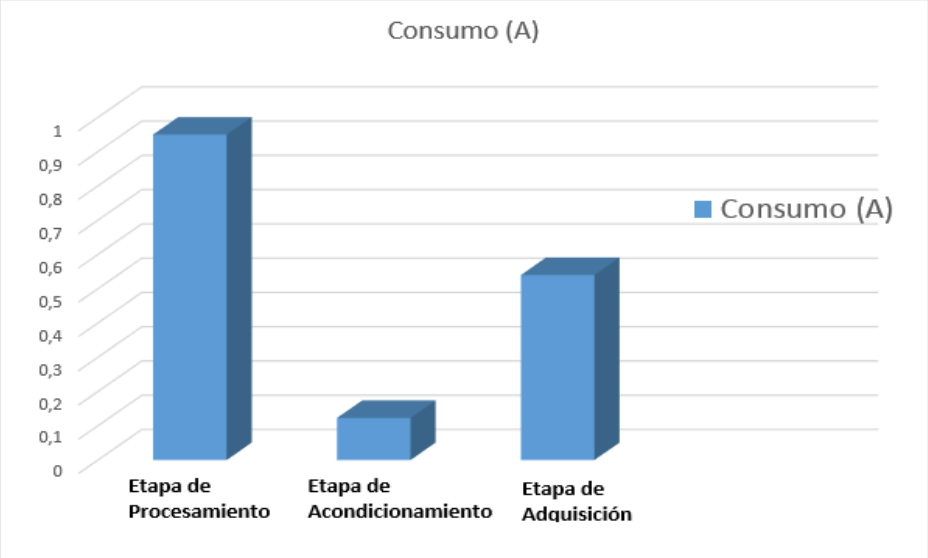

Gráfico 9-3. Consumo de corriente del dispositivo

Realizado por: Paullán, A., 2021

CAPÍTULO IV

4. ANÁLISIS DE COSTOS DEL PROTOTIPO IMPLEMENTADO

En el siguiente apartado se indica el precio de los dispositivos utilizados para la implementación del prototipo de detección de uso de mascarilla y medición de temperatura, el cual tiene un costo de 385 dólares americanos (USD)

4.1 Costos Directos

En la tabla 1-4 se detallan los costos que influyeron directamente para la implementación del prototipo de detección de mascarilla y medición de temperatura para el personal que labora en la empresa PAUFIT.

Tabla 1-4: Costo directos de implementación del prototipo

CANTIDAD	DESCRIPCIÓN	V. UNITARIO (USD)	V. TOTAL (USD)
1	Kit de Raspberry Pi4	125	125
1	Disipador de calor	30	30
1	Pantalla de 3.5 ‘	25	25
1	Cámara de Rspberry Pi	30	30
1	Arduino Uno	20	20

1	Sensor ultrasónico	5	5
1	Sensor de proximidad	5	5
1	Sensor de temperatura	15	15
1	Modulo amplificador	4	4
1	Parlante de 3w	5	5
1	Tarjeta electrónica	15	15
1	Plug de audio	3	3
1	Estructura acero inoxidable	100	100
1	Conector de alimentación dc	3	3
		Total	385

Realizado por: Paullán, A. 2021

Se pudo adquirir los materiales e implementos elegidos para el prototipo sin ningún inconveniente, en función al análisis de costos el prototipo puede ser implementado sin una costosa inversión.

4.2 Comparación de costos entre dispositivos.

Comercialmente existen dispositivos similares en el mercado, para realizar una comparación de costos y de funciones se eligió el dispositivo que la empresa Hikvision comercializa, esta empresa es una de las empresas líderes proveedoras de productos y soluciones de seguridad, luego de analizar las características y funciones del dispositivo que la empresa Hikvision ofrece se puede comparar y comprobar la similitud con el dispositivo desarrollado para este trabajo.

Tabla 2-4: Comparación entre dispositivos de detección de mascarilla y temperatura

Dispositivo Empresa Hikvision	**Dispositivo**
Modelo DS-K5604A3XFV.	Prototipo para detección de mascarilla y temperatura
Características y Funciones	**Características y Funciones**
Posee Cámara Realiza reconocimiento Facial Realiza control de presencia Detecta el uso de mascarilla Detecta temperatura	Posee Cámara No realiza reconocimiento facial Realiza control de presencia Detecta el uso de mascarilla Detecta temperatura
Precio total ($)	**Precio total ($)**
1200	**385**

Realizado por: Paullán, A. 2021

Analizando las funciones que realizan los 2 dispositivos como muestra la tabla 2-4, se demuestra que el prototipo de detección de mascarilla y temperatura construido tiene un ahorro del 68 % respecto al modelo comercializado por la empresa Hikvision, por lo que se considera un porcentaje aceptable de inversión, teniendo en cuenta que el prototipo desarrollado en este trabajo dispone de características similares a el dispositivo comercial.

CONCLUSIONES

De la indagación sobre la existencia de dispositivos para detectar el uso de mascarilla y temperatura se pudo definir que en el mercado existen algunos dispositivos tales como los que las empresas Hikvision, Automatic Systems, By Demes Group disponen, estos modelos basados en Inteligencia Artificial para el control de acceso con control de temperatura, ofrecen soluciones tecnológicas para mejorar los procedimientos de verificación de uso de mascarilla y control de temperatura.

Mediante revisión bibliográfica se logró determinar los elementos hardware y software necesarios y adecuados para llevar a cabo la construcción del dispositivo de detección de mascarilla y medición de temperatura.

Con el propósito de crear un protocolo de comunicación apropiado para controlar el acceso de los usuarios que no cumplen con el uso de mascarilla, se desarrolló tres tipos de comunicación (visual, auditivo y por notificación a través de un mensaje), los cuales sirvieron para controlar que el personal de la empresa PAUFIT haga el uso correcto de la mascarilla, precautelando así la salud de todos los que laboran en ella.

Se realizó diferentes pruebas de detección de temperatura y mascarilla en tiempo real con el personal que labora en la empresa PAUFIT, determinando así la funcionabilidad del prototipo. Además, a través de las pruebas realizadas en la noche respecto a la detección de mascarilla se determinó que el dispositivo es eficiente en un 92%, demostrando que el dispositivo no tiene inconvenientes para realizar la detección en este tipo de ambiente.

Mediante revisión bibliográfica de artículos, revistas, tesis se logró adquirir cierto conocimiento sobre el entrenamiento de redes neuronales basado en el modelo de Deep Learnig, permitiendo incorporar modelos destinados a la clasificación, detección de objetos o personas, permitiendo ser implementados en tarjetas de bajo costo como la Raspberry Pi, logrando así desarrollar un algoritmo eficiente para detección de mascarilla, que cumpla con los objetivos propuestos.

A través de las pruebas realizadas se determinó un porcentaje del 94,25% de eficiencia general del dispositivo, además también se determinó que la distancia de reconocimiento es desde los 50cm a 150cm con respecto a la estación de censado.

RECOMENDACIONES

El prototipo fue diseñado para interiores, en caso de replicar el dispositivo se debe tomar en cuenta algunos factores como la lluvia, el polvo, la humedad, condiciones que pueden afectar los componentes del dispositivo.

La utilización de programas y hardware libre ha posibilitado el diseño y creación de un primer modelo de bajo precio; no obstante, el funcionamiento del mismo se puede mejorar llevando a cabo mejoras referentes al hardware, pero sufriendo una alteración en su precio final.

La estructura del prototipo se puede fabricar de otro material para disminuir el precio final del dispositivo.

Se podría incorporar una cámara de mayor resolución para de esa manera obtener imágenes de mejor calidad, optimizando la repuesta del dispositivo en la detección de personas con mascarillas.

Para poder incrementar el porcentaje de eficiencia del dispositivo se recomienda ampliar el entrenamiento de la red neuronal.

Si se desea minimizar los precios de construcción del prototipo, se debe tomar en cuenta que los dispositivos no deben perder la calidad y que estos deben ser los óptimos, para la construcción del algoritmo de detección.

BIBLIOGRAFÍA

ALEJANDRO, V. *Sistemas de Visión Artificial*. [En línea] 2019. Disponible en: https://docplayer.es/55281847-Introduccion-al-diseno-de-microrobots-moviles.html

ANDREEA, P. *Aplicación para Detección y Reconocimiento Facial*. [En línea], Sevilla**.** 2014; Disponible en: https://www.researchgate.net/publication/280445703.

BARRIOS, J. *Algoritmo para la detección de mascarilla.* [En línea] 2020. Disponible en: https://www.juanbarrios.com/algoritmo-para-la-deteccion-del-uso-de-la-mascarilla/.

BOOTCAMP. *Intro a las redes neuronales convolucionales*. [En línea] 2019. Disponible en: https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8.

BORRMART. *SafeFlow automatiza la medición de temperatura y la deteccion de mascarilla*. [En línea] 2020. https://www.seguritecnia.es/revistas/seg/478/index.html#zoom=z.

BRICOGEEK. *NodeMCU V3 Wifi - ESP8266, CH340.* [En línea] 2021. Disponible en: https://tienda.bricogeek.com/wifi/1033-nodemcu-v3-wifi-esp8266-ch340.html.

BROWNLEE, J. *Introducción suave al algoritmo de optimización de Adam para el aprendizaje profundo.* [En línea] 2017. Disponible en: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

BUPA. *Covid 19 centro de informacion*. [En línea] 2020. Disponible en: https://www.bupasalud.com/salud/coronavirus.

CAMPOS, P. *Soluciones para el control de accesos*. [En línea] 2020. Disponible en: https://www.interempresas.net/Seguridad/Articulos/308020-Soluciones-Hikvision-para-el-control-de-accesos.html.

CÁRDENAS, A. *Sensor Ultrasónico* . [En línea] 2015. Disponible en: https://electrocrea.com/blogs/tutoriales/33306499-sensor-ultrasonico.

CDC. *Centros para el control y la prevención de enfermedades*. [En línea] 2021. Disponible en: https://espanol.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/cloth-face-cover-guidance.html.

ELSEVIER. Fiebre . [En línea] 2017. Disponible en: https://www.elsevier.es/es-revista-farmacia-profesional-3-articulo-fiebre-X0213932417620584.

FERNÁNDEZ, Y. *Qué es Arduino, cómo funciona y qué puedes hacer con uno.* [En línea] 2020. Disponible en: https://n9.cl/hl53e

FITER, M. *El reconocimiento facial contra el covid:asi arende la inteligencia facial a reconocer mascarrillas.* [En línea] 2020. Disponible en: https://www.elindependiente.com/futuro/2020/06/07/la-lucha-del-reconocimiento-facial-contra-el-covid-asi-aprende-a-reconocer-mascarillas.

GAMA, A. *Aplicación de percepción mejorada para personas con problemas visuales usando deep learning y dispositivos vestibles.* [En línea] 2020. Disponible en: https://rua.ua.es/dspace/bitstream/10045/107576/1/Aplicacion_de_percepcion_mejorada_para_personas_con_Gama_Garcia_Angel_Manuel.pdf.

GITHUBGIST. *Instalación de Python 3.7.4 en Raspbian.rst.* [En línea] 2020. Disponible en: https://gist.github.com/SeppPenner/6a5a30ebc8f79936fa136c524417761d.

GONZALES, C Y ZARAMA, D. *Aplicación de sistemas de visión por computadora para la inspección visual de Uchuvas.* [En línea] 2014. Disponible en: https://core.ac.uk/download/pdf/71419539.pdf.

HEALTHWISE. *Temperatura Corporal*. [En línea] 2020. Disponible en: https://www.cigna.com/individuals-families/health-wellness/hw-en-espanol/pruebas-medicas/temperatura-corporal-hw198785.

ANDRADE, H, HIIDALGO, P Y SINCHE, S. *Modelo para detectar el uso correcto de mascarillas en tiempo real.* [En línea] 2021. Disponible en: https://www.riti.es/ojs2018/inicio/index.php/riti/article/view/281.

IBEROBOTICS. *Módulo Relé 5V 2 canales optoacoplados, salida 10A/250VAC.* [En línea] 2020. Disponible en: https://www.iberobotics.com/producto/modulo-rele-5v-2-canales-optoacoplado/.

JIMÉNEZ, O Y MAGALY G. "*Desarrollo de un sistema de visión artificial para la deteccion de personas en un semáforo.* [En línea] 2015. Disponible en: https://dspace.unl.edu.ec/jspui/bitstream/123456789/11225/1/Jim%C3%A9nez%20Ochoa%2C%20Magaly%20Gabriela.pdf.

KRUTY, G. *Face detection and tracking: Using OpenCV*. [En línea] 2017. Disponible en: https://ieeexplore.ieee.org/document/8203730.

LEYVA, C. *Termometro Digital* . [En línea] 2019. Disponible en: https://tesis.ipn.mx/bitstream/handle/123456789/28142/Term%C3%B3metro%20Digital.pdf?sequence=1&isAllowed=y.

LLAMAS, L. *Termómetro a distancia infrarrojo.* [En línea] 2016. Disponible en: https://www.luisllamas.es/arduino-y-el-termometro-infrarrojo-a-distancia-mlx90614/.

LOGRONO, C. "*DESARROLLO DE UN SISTEMA AUTOMÁTICO PARA EL.* [En línea] 2016. Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/6010/1/108T0164.pdf.

LOPEZ, J. *Xataca. Raspberry Pi 4 ya a la venta: características y precio oficial de sus 3 versiones.* [En línea] 2019. Disponible en: https://hardzone.es/2019/06/24/raspberry-pi-4-caracteristicas-precio/.

LUCAS, J. *Que es Raspbery PI.* [En línea] 2019. Disponible en: https://openwebinars.net/blog/que-es-raspberry-pi/.

MALIHA, K. *Detección y reconocimiento de rostros mediante Open Cv*. [En línea] 2019. Disponible en: https://ieeexplore.ieee.org/document/8974493.

MORENO, A. *Reconocimiento Facial Automático mediante Técnicas de.* [En línea] 2004. Disponible en: http://oa.upm.es/625/1/10200408.pdf..

OMS. *Preguntas y respuestas sobre la enfermedad por coronavirus (COVID-19).* [En línea] 2020. Disponible en: https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/advice-for-public/q-a-coronaviruses.

OVALLE, H. *DESCRIPCIÓN DEL AMPLIFICADOR PAM8403*. [En línea] 2019. Disponible en: https://hetpro-store.com/TUTORIALES/amplificador-pam8403/.

OZAN, C. *Diseño de un detector facial profundo por Mask R-CNN*. [En línea] 2019. Disponible en: https://ieeexplore.ieee.org/document/8806447.

PAGE, B. *Kinect, un accesorio para jugar que se ha convertido en herramienta para artistas*. [En línea] 2019. Disponible en: https://www.lavanguardia.com/tecnologia/20190428/461843755133/kinect-arte-microsoft-videojuegos-sensor-hackeada.html.

PALACIOS, R. *Reconocimiento de imagenes mediante Raspberry Pi*. [En línea] 2018. .

POMARE, J. *Manual de Arduino.* [En línea] 2009. Disponible en: https://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf.

RASPBERRY PI, FOUNDATION. *Raspberry Pi*. [En línea] 2019. [Citado el: 2 de Diciembre de 2019.]. Disponible en: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

RIOS, C. "*Comparación y uso de modelos de redes neuronales convolucionales aplicado al desarrollo de tecnología asistiva para identificación de productos.* [En línea] 2020.

SCHNEIDER, F. *Componentes electronicos.* 2020. Disponible en: https://www.fierceelectronics.com/components/advances-ir-temperature-measurement.

SEGURIDAD, C. *Nuevas soluciones tecnológicas con detección de temperatura de Hikvision.* [En línea] 2020. Disponible en: https://cuadernosdeseguridad.com/2020/04/deteccion-de-temperatura-solucion-hikvision/.

BLANCO, J. *Soluciones de control de accesos con detección de temperatura y mascarilla.* [En línea] 2020. Disponible en: https://cuadernosdeseguridad.com/2020/05/by-demes-soluciones-control-accesos/.

TENSORFLOW. *Instala TensorFlow* . [En línea] 2020. Disponible en: https://www.tensorflow.org/install/source_rpi?hl=es.

VALLEJO, G. *Sistema de Monitoreo de Signos Vitales y Alerta de Accidentes para Personas.* [En línea] 2015 Disponible en: https://repositorio.uta.edu.ec/bitstream/123456789/15108/1/Tesis_t1071ec.pdf.

VÁZQUEZ, M. *Reconocimiento de Objetos usando Deep Learning*. [En línea] 2016. Disponible en:http://bibing.us.es/proyectos/abreproy/91070/fichero/Marina+Vazquez+Rull+Reconocimiento+de+Objetos+usando+Deep+Learning+TFG.pdf.

XUKYO. *Crear una interfaz web con NodeMCU ESP8266.* [En línea] 2020. Disponible en: https://www.aranacorp.com/es/crear-una-interfaz-web-con-nodemcu-esp8266/.

ANEXOS

ANEXO A: Programación Código de Entrenamiento

```
# importacion de librerias necesarias
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
fom tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_i
nput
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os
from google.colab import drive

from google.colab import drive as cdrive
import tensorflow as tf
```

abrir directorio de ubicación

```
DIRECTORIO = '/content/gdrive/MyDrive/deteccion_mascarrilla/datase'

CATEGORIAS      =      ["sin_mascarrilla","incorrecta_mascarilla",
"correcta_mascarilla"]
Ubicacion_imagenes = list(paths.list_images(DIRECTORIO))
print(imagePaths)

## leer y pre procesar  cada imagen
data = []
labels = []

for category in CATEGORIAS:
    path = os.path.join(DIRECTORIO, category)
    for img in os.listdir(path):
     img_directorio = os.path.join (path, img) ## abrir el directorio
```

```
        # Redimensionamiento de  imagen
        imagen = load_img (img_path, target_size= (224, 224))
        # conversion a matriz
        imagen = img_to_array(imagen)
        # preprocesamiento de imagen
        imagen = preprocess_input(imagen)
        # alamacenamiento  de imagen
        data.append(image)
        # alamacenamiento  de label
        labels.append(category)

print("compilo")

#categorizacion de las clases
lb = LabelBinarizer ()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

### conversion a array

datos = np.array(data, dtype="float32")
labels = np.array(labels)
print("compilado")

# construcción de  generardo de imagen
aug = ImageDataGenerator(
  rotation_range=20,
  zoom_range=0.15,
  width_shift_range=0.2,
  height_shift_range=0.2,
  shear_range=0.15,
  horizontal_flip=True,
  fill_mode="nearest")

#   creacion   de      del   modelo   MobileNetV2   preentrenado
baseModel = MobileNetV2(weights="imagenet", include_top=False,
  input_tensor=Input(shape=(224, 224, 3)))
BaseModel. Summary ()
# contruccion de  la cabezera del modelo
cabecera = baseModel.output
cabecera = AveragePooling2D (pool_size=(7, 7))( cabecera)
cabecera = Flatten(name="flatten")(headModel)
cabecera  = Dense(128, activation="relu")(cabecera)
cabecera = Dropout(0.5)(headModel)
cabecera = Dense(2, activation="softmax")(cabecera)
print("compilado")
```

```
# unión de la cabezera con el modelo preentrenado
model = Model(inputs=baseModel.input, outputs=headModel)

# control de en la fase de entrenamiento solo para que la cabezera y
capas ayacentes sean entrenadas
for layer in baseModel.layers:
  layer.trainable = False

# Compilación de optimizador para el entrenamiento
print ("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile (loss="binary_crossentropy", optimizer=opt,
  metrics=["accuracy"])
# Entrenamiento de la cabezera
print ("[INFO] training head...")
H = model.fit(
  aug.flow(trainX, trainY, batch_size=BS),
  steps_per_epoch=len(trainX) // BS,
  validation_data=(testX, testY),
  validation_steps=len(testX) // BS,
  epochs=EPOCHS)
print (" Almacenamiento  del modelo ")
model.save("modelo.model", save_format="h5")

# Graficar la pérdida y la precisión del entrenamiento
N = EPOCHS
plt.style.use("grafica de tasa de aprendiaje ")
plt.figure()
plt.plot(np.arange(0,             N),             H.history["perdida
"], label="train_perdida")
plt.plot(np.arange(0,     N),     H.history["validacion     perdiad
"], label="val_perdida")
plt.plot(np.arange(0, N), H.history["preciccion"], label="train_pre
ci")
plt.plot(np.arange(0, N), H.history["val_presicion"], label="val_pr
e")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epocas #")
plt.ylabel("perdida/precisión")
plt.savefig("plot.png")


programa principal

# import the necessary packages

import time
import cv2
```

```
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os
import datetime

import time
# initialize the camera and grab a reference to the raw camera capture
#camera = PiCamera()
#camera.resolution = (640, 480)
#camera.framerate = 32
time.sleep(0.1)
tiempo_1=datetime.datetime.now()
tiempo_1=tiempo_1.second
tiempo_2=datetime.datetime.now()
tiempo_2=tiempo_2.second
tiempo_1_f=datetime.datetime.now()
tiempo_2_f=datetime.datetime.now()
import time
# Infinite loop
estado_actual_p=0
estado_anterior_p=0
estado_actual_p=0
estado_anterior_p=0
detccion_persona=0
estado_actual_m=0
estado_anterior_m=0
detccion_mano=0
detccion_persona=0
detccion_mano=0
def funcio_deteccion(frame, modelo_rostro, modelo_mascarilla):

 (h, w) = frame.shape[:2]
 blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),(104.0, 177.0,
123.0))
 faceNet.setInput(blob)
 detections = faceNet.forward()


 faces = []
 locs = []
```

```
 preds = []
      # loop over the detections
 for i in range(0, detections.shape[2]):

  confidence = detections[0, 0, i, 2]
           # filter out weak detections by ensuring the confidence
is
           # greater than the minimum confidence
  if confidence > 0.15:
                # compute the (x, y)-coordinates of the bounding
box for
                # the object
   box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
   (startX, startY, endX, endY) = box.astype("int")

                #  ensure  the  bounding  boxes  fall  within  the
dimensions of
                # the frame
   (startX, startY) = (max(0, startX), max(0, startY))
   (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                # extract the face ROI, convert it from BGR to RGB
channel
                # ordering, resize it to 224x224, and preprocess it
   face = frame[startY: endY, startX:endX]
   face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
   face = cv2.resize(face, (224, 224))
   face = img_to_array(face)
   face = preprocess_input(face)
   face = np.expand_dims(face, axis=0)
   faces.append(face)
   locs.append((startX, startY, endX, endY))
 if len(faces) > 0:
  preds = maskNet.predict(faces)
 return (locs, preds)
print("[INFO] loading rostro detector model...")
modelo_rostro                                                    =
cv2.dnn.readNet("deploy.prototxt","res10_300x300_ssd_iter_140000.ca
ffemodel")
modelo mascarilla = load_model("modelo.model")
cont=0
value=10
persona_mascarilla=1;
while True:

 negro= np.zeros((800, 800, 3), dtype = "uint8")
 detccion_persona=0
 detccion_mano=0
```

```
 #ersona_mascarilla=0
 #print("detccion_persona",persona mascarilla)

image=cv2.imread("C:\\Users\\Gabriel\\Pictures\\pruebas_n1\\noche\\
mas carillas\\fonfo.jpg",1)
 #image = frame.array
 frame = imutils.resize(image, width=400)
 frame1=frame.copy()
 hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
 h, s, v = cv2.split(hsv)

 lim = 255 - value
 v[v > lim] = 255
 v[v <= lim] += value
 end_hsv = cv2.merge((h, s, v))
 frame = cv2.cvtColor(end_hsv, cv2.COLOR_HSV2BGR)

 cv2.imshow("imagen aumento de brillo",frame)

 cont=cont+1
 if cont%2==0:
  tiempo_1=datetime.datetime.now()
  tiempo_1=tiempo_1.second
 elif cont%3==0:
  tiempo_2=datetime.datetime.now()
  tiempo_2=tiempo_2.second
 diferencia=abs(tiempo_1-tiempo_2)
 variable_casacda=0
 print("diferencia     ",diferencia)
 if diferencia==1:
  tiempo_1_f=datetime.datetime.now()

  (locs, preds) = detect_and_predict_mask (frame, faceNet, maskNet)
  for (box, pred) in zip(locs, preds):
   (startX,startY, endX, endY) = box
   (mask, withoutMask) = pred
   label = "mascarilla" if mask > withoutMask else "No mascarilla"
   color = (150, 255, 0) if label == "mascarilla" else (0, 0, 255)
#(150, 255, 0)
   label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
   cv2.putText(frame, label, (startX, startY - 10),
   cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
   cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
   #cv2.imwrite('r_s_m.jpg',frame)
   #print(" label ",label)
   print('mascarilla',mask)
   print(' sin mascarilla',withoutMask)
   if mascarilla > sin_mascarrilla :
```

```
    persona_mascarilla==0
    #engine.say("MARCARILLA CORCTAMENTE COLOCADA  ")

    #engine.runAndWait()

    crop              =              frame[startY:startY+abs(startY-
endY),startX:startX+abs(startX-endX)]
    crop = imutils.resize(crop , width=300)
    h1, w1 = crop.shape[:2]
    negro[100:100+h1,300:300+w1] = crop
    cv2.imshow("Frame",frame)
    variable_casacda=1

   else:
    persona_mascarilla==0
    #engine.say('COLOQUESE UNA MASCARILLA PORFAVOR  ')
    #engine.runAndWait()
    crop         =         frame[startY:         startY+abs(startY-
endY),startX:startX+abs(startX-endX)]
    crop = imutils.resize(crop , width=300)
    h1, w1 = crop.shape[:2]
    negro[100:100+h1,300:300+w1] = crop
    cv2.imshow("Frame",frame)

    #cv2.imshow("Frame",crop )
    variable_casacda=1



 if variable_casacda==1:
  print("ENTRANDOOOO")
  tiempo_1_f=datetime.datetime.now ()
  #print ("diferencia     ",diferencia)
  (locs, preds) = detect_and_predict_mask(negro, faceNet, maskNet)
  for (box, pred) in zip(locs, preds):
   (startX, startY, endX, endY) = box
   (mask, withoutMask) = pred
   label = "  mascarilla" if mask > withoutMask else " no mascarilla"
   color = (125, 255, 0) if label == "mascarilla" else (0, 0, 255)
   label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
   cv2.putText(negro, label, (startX, startY - 10),
   cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
   cv2.rectangle(negro, (startX, startY), (endX, endY), color, 2)
   #cv2.imwrite('r_s_m.jpg',frame)
   #print(" label ",label)
   print('mascarilla1',mask)
   print(' sin mascarilla1',withoutMask)
   if mask > withoutMask:
```

```
    bb=1
   # cv2.imshow("Frame",negro)
   else:

    bb=0

   #cv2.imshow ("Frame",negro)



 tiempo_f=datetime.datetime.now()
 tiempo_f=str(tiempo_f)
 print(tiempo_f)
 tiempo_f=tiempo_f+".jpg"
 #cv2.imwrite(tiempo_f,frame)


 if diferencia==1:
  tiempo_2_f=datetime.datetime.now ()
  print("diferencia",abs (tiempo_1_f-tiempo_2_f))
 if cont==100:
  cont=0
 tiempo_1=datetime.datetime.now()
 tiempo_1=tiempo_1.second
 tiempo_2=datetime.datetime.now()
 tiempo_2=tiempo_2.second

 #cv2.putText(frame,s , (200, 300 - 10),
 #cv2.FONT_HERSHEY_SIMPLEX, 0.85, (55,155,0), 2)
 rotatacion= imutils.rotate(negro, 90)
 rotatacion = imutils.resize( rotatacion, width = 480)
 key = cv2.waitKey(1) & 0xFF
  if key == ord("q"):
  break
cv2.destroyAllWindows()
```

Código de Arduino lectura de temperatura

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"
// Instanciar objeto
Adafruit_MLX90614 termometroIR = Adafruit_MLX90614();
const int Trigger = 9;   //Pin digital 2 para el Trigger del sensor
const int Echo = 8;   //Pin digital 3 para el Echo del sensor
//Inicia a serial por software nos pinos 10 e 11
```

```
SoftwareSerial mySoftwareSerial(3,2); // RX, TX
DFRobotDFPlayerMini myDFPlayer;
char command;
int pausa = 0;
int r = 1;
int ultimo_digito;
unsigned long Now = millis()/1000;
// we use modulo to split the parts
int Seconds = Now%60;
int audio_deciaml=18;
String inString = "";
int numero=900;
int entero_1=28;
int entero_2=28;
int parte_entera;
int decimales=0;
int cont=0;
float estado_ant=0;
float estado_act=0;
float resta=0;
float resta_ant=0;
float resta_act=0;
float resta_abs=0;
float estado_diferencia_anterior=0;
float estado_diferencia_actual=0;
float inicio_lectura=0;
int cont_lectura=0;
float lectura=0;
float resta_temperatura;
unsigned long tiempo_lectura_inicio = millis()/1000;
unsigned long tiempo_lectura_final = millis()/1000;
float temperatura_inicial=0;
int control_temperatura=0;
int cont_temperatura=0;
int conversion_temp=20;
float k=0.0;
void setup()
{
  // Iniciar comunicación serie
  Serial.begin(9600);
  pinMode(10, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(Trigger, OUTPUT); //pin como salida
  pinMode(Echo, INPUT);  //pin como entrada
  digitalWrite(Trigger, LOW);//Inicializamos el pin con 0
  // Iniciar termómetro infrarrojo con Arduino
  termometroIR.begin();
```

```
mySoftwareSerial.begin(9600);
//Inicializa a serial do Arduino
Serial.begin(9600);
//Verifica se o modulo esta respondendo e se o
//cartao SD foi encontrado
//Serial.println();
//Serial.println(F("DFRobot DFPlayer Mini"));
//Serial.println(F("Initializing DFPlayer module ... Wait!"));

if (!myDFPlayer.begin(mySoftwareSerial))
{
//Serial.println(F("Not initialized:"));
//Serial.println(F("1. Check the DFPlayer Mini connections"));
//Serial.println(F("2. Insert an SD card"));
while (true);
}
//Serial.println();
//Serial.println(F("DFPlayer Mini module initialized!"));

//Definicoes iniciais
myDFPlayer.setTimeOut(500); //Timeout serial 500ms
myDFPlayer.volume(50); //Volume 5
myDFPlayer.EQ(0); //Equalizacao normal

// myDFPlayer.play(1);  //Play the first mp3
 digitalWrite(4,LOW);
 digitalWrite(5,LOW);
 }
void loop()
{
     while (Serial.available() > 0)
{

 // r =  (Serial.parseInt() );  //conveting the value of chars to
integer
 r=1;
   inString = Serial.readString();

   numero= inString.toInt();

   Serial.println(numero);
    if (numero==700)
  {
   // Serial.println(r);
    myDFPlayer.play(1);
  //  Serial.println("BBSITAAA");
  }
    if (numero==400)
  {
   // Serial.println(r);
    myDFPlayer.play(28);
```

```
  //  Serial.println("BBSITAAA");
  }
     if (numero==200)
  {
   // Serial.println(r);
    //myDFPlayer.play(1);
  //  Serial.println("BBSITAAA");

   digitalWrite(4,LOW);
   digitalWrite(5,LOW);
  }
    if (numero==800)
  {
    myDFPlayer.play(2);
  }
  if (numero==900)
  {
    myDFPlayer.play(3);

  }
}
 k=0.0;
 //Serial.println(k);
  cont++;
  // Obtener temperaturas grados Celsius
  float temperaturaAmbiente = termometroIR.readAmbientTempC();
  float temperaturaObjeto = termometroIR.readObjectTempC();

  // Mostrar información
  //Serial.print("Temp. ambiente => ");
  //Serial.print(temperaturaAmbiente);
 // Serial.println("°C");

 long t; //timepo que demora en llegar el eco
  long d; //distancia en centimetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
  d = t/59;             //escalamos el tiempo a una distancia en cm
  if (d>20 && d<80)
  {
    digitalWrite(10, HIGH);
  }
  else
  {
    digitalWrite(10, LOW);
  }
        //Serial.print("Distancia: ");
// Serial.println(d);

  //Enviamos serialmente el valor de la distancia
  //Serial.print("cm");
  //Serial.println();

 // Serial.print("Temp. objeto => ");
```

```
 //Serial.println(temperaturaObjeto+3.7);

 if (cont%2==0)
  {
 estado_ant=(temperaturaObjeto+3.7);

  }
  else
{
 estado_act=(temperaturaObjeto+3.7);

}
  resta_abs=int(abs(estado_ant-estado_act));

 estado_diferencia_actual= resta_abs;
inicio_lectura=0;
lectura=0;
if(estado_diferencia_anterior<=0 && estado_diferencia_actual<=0)
{
  cont_lectura++;
}
 if(estado_diferencia_anterior<=0 && estado_diferencia_actual>0)
{
  inicio_lectura=1;
  }
 if (inicio_lectura==1 && cont_lectura>1)
{

// Serial.println("LECTURAAAAA INCICIOOOOO");
 cont_lectura=0;
 lectura=1;
 temperatura_inicial=temperaturaObjeto+3.7;
 control_temperatura=1;
  //tiempo_lectura_inicio = millis()/1000;
}

// tiempo_lectura_final = millis()/1000;
resta_temperatura=temperaturaObjeto+3.7-temperatura_inicial;
if   (control_temperatura==1   &&      resta_temperatura>0.75   &&
numero==700 )
{
  cont_temperatura++;

  if (cont_temperatura==4)
  {

   // Serial.println("ENTRANDOOOOOO");
   // Serial.println(temperaturaObjeto+3.7);
    conversion_temp=(temperaturaObjeto+5.3)*10;
    if (conversion_temp>371)
    {
      k=2.0;

       digitalWrite(4,LOW);
       digitalWrite(5,HIGH);
       numero=200;
      // myDFPlayer.play(28);
     // Serial.println(k);
```

```
   }
   else
   {
      digitalWrite(4,HIGH);
      digitalWrite(5,LOW);
      numero=200;
     k=1.0;
   // Serial.println(k);
   }
 //  Serial.println(conversion_temp);
   entero_1=(conversion_temp/100)%10;
  // Serial.println(entero_1);

   entero_2=(conversion_temp/10)%10;
  // Serial.println(entero_2);
   parte_entera=entero_1*10+entero_2;
  // Serial.println(parte_entera);
   decimales=(conversion_temp/1)%10;
 //  Serial.println(decimales);

  if (parte_entera==37)
{
  myDFPlayer.play(13);
   delay(2300);
  myDFPlayer.play(audio_deciaml);
  if (audio_deciaml>19)
  {
     delay(2300);
      myDFPlayer.play(28);
  }
}
      if (parte_entera==38)
{
  myDFPlayer.play(14);
   delay(2300);
  myDFPlayer.play(audio_deciaml);
  delay(2300);
   myDFPlayer.play(28);
}
      if (parte_entera==39)
{
  myDFPlayer.play(15);
   delay(2300);
  myDFPlayer.play(audio_deciaml);
  delay(2300);
   myDFPlayer.play(28);
}
      if (parte_entera==40)
{
  myDFPlayer.play(16);
   delay(2300);
  myDFPlayer.play(audio_deciaml);
  delay(2300);
   myDFPlayer.play(28);
}
    if (parte_entera==41)
{
  myDFPlayer.play(17);
```

```
     delay(2300);
    myDFPlayer.play(audio_deciaml);
    delay(2300);
     myDFPlayer.play(28);
    }
     control_temperatura=0;
     resta_temperatura=0;
     cont_temperatura=0;
   }
 }
 else
 {
   digitalWrite(4,LOW);
   digitalWrite(5,LOW);
 }
  //Serial.println(temperaturaObjeto+3.7);
  //Serial.println(temperatura_inicial);
   estado_diferencia_anterior=estado_diferencia_actual;


 //Serial.println(cont_lectura);
  if (cont==103)
  {
    cont=0;
  }

   // Serial.println("°C");

  delay(100);

  // Si quieres mostrar la información en grados Fahrenheit utiliza
las funciones
  // readAmbientTempF() para temperatura ambiente
  // readObjectTempF() para temperatura del objeto

}
```

ANEXO B: Protocolo frente a la exposición del covid-19

LINEAMIENTO	GRÁFICO
Lineamientos de seguridad para el retorno al lugar de trabajo	
Uso de mascarilla obligatoria	
Medición de temperatura corporal Obligatoria	
Guardar distancia (al menos 2 metros) en la entrada y salida del lugar de trabajo como durante la permanecía en el mismo	
Evitar el saludo con contacto físico incluido dar la mano o beso	
Usar alcohol o gel de manos con una concentración del 70%	

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 2 / 12 / 2021

INFORMACIÓN DE LA AUTORA
Nombres – Apellidos: ALEXANDRA ELIZABETH PAULLÁN PUNGUIL
INFORMACIÓN INSTITUCIONAL
Facultad: INFORMÁTICA Y ELECTRÓNICA
Carrera: ELECTRÓNICA Y AUTOMATIZACIÓN
Título a optar: INGENIERA EN ELECTRÓNICA Y AUTOMATIZACIÓN
f. Analista de Biblioteca responsable: