UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

TESIS DE GRADO

TÍTULO: DETECTOR DE RESTRICCIONES PARA LA PREVENCIÓN DE CASOS DE CONTAGIO EN LA EMERGENCIA SANITARIA BASADO EN ARDUINO Y REDES NEURONALES

Tesis de Grado para obtener el Título de Licenciatura en Informática

Mención Ingeniería de Sistemas Informáticos

POR: MARÍA ALEJANDRA MAMANI CORDERO

TUTOR METODOLÓGICO: PhD. FÁTIMA C. DOLZ SALVADOR

ASESOR: LIC. CELIA ELENA TARQUINO PERALTA

LA PAZ – BOLIVIA

2021

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

Dedico este trabajo a Dios, mi familia y seres queridos quienes durante los años de estudio me apoyaron incondicionalmente, especialmente a mi mamá María Cordero Ticona y mi abuelo Manuel Ticona Canaza quien está en el cielo, alentándome siempre a seguir adelante, enseñarme a luchar por mis metas y ser feliz.

alesisima777@gmail.com

AGRADECIMIENTOS

Agradezco a Dios por darme la fortaleza en esos momentos de debilidad que se me presentaron, por acompañarme, darme sabiduría, y ser un refugio en el que me puedo proteger.

A mi mamita María, quien ha estado ahí cuando más la necesitaba, me enseñó a valorar todo lo que la vida me puede ofrecer en base a su esfuerzo, ha logrado que alcance mis metas.

A mi papá abuelo Manuel, quien en vida me brindó su apoyo y comprensión a lo largo de mi vida y darme su apoyo incondicional.

A mi asesora Ph.D. Fátima Consuelo Dolz Salvador por su gran colaboración y aportes, especialmente en estos tiempos difíciles para todos.

A mi tutor metodológico Lic. Celia Tarquino Peralta quien me ayudó a fortalecer mis conocimientos como estudiante, y brindó total apoyo en este proceso.

Estoy agradecida con los resultados de esta tesis, a la Universidad Mayor de San Andrés, y a todas aquellas personas que de alguna forma han invertido en mi desarrollo como estudiante dentro de esta prestigiosa Universidad.

También agradecer a mis compañeros con los cuales compartí muchas ideas, metas y sueños que hoy realizamos.

RESUMEN

Debido al incremento de casos de contagio durante la emergencia sanitaria en nuestro país Bolivia, se propone diseñar e implementar un detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, el cual permite detectar la falta de uso de barbijo y la aglomeración entre personas, se emplea el módulo electrónico Arduino Uno y las redes neuronales, las cuales realizan un control de barbijos y previenen las multitudes en las calles.

En base a la metodología de investigación del modelo en V, el detector es implementado en Arduino mediante el sensor Buzzer y componentes electrónicos, del mismo modo se tienen las redes neuronales del tipo Convolucional, las cuales generan un modelo para la detección de restricciones, tomando en cuenta las actividades de prueba (validación y verificación).

Se realiza un detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria y el cual emite un sonido de alarma para la prevención mediante la detección de cubre bocas y el distanciamiento físico entre las personas.

Palabras Clave

Arduino, redes neuronales, sensor, aglomeración, barbijo, contagio, emergencia, modelo.

ABSTRACT

Due to the increase in cases of contagion during the health emergency in our country Bolivia, it is proposed to design and implement a restriction detector for the prevention of cases of contagion in the health emergency based on Arduino and neural networks, which allow detecting the lack of use of chinstrap and crowding between people, the Arduino Uno electronic module and neural networks are used, which perform chinstrap control and prevent crowds in the streets.

Based on the research methodology of the V model, the detector is implemented in Arduino by means of the Buzzer sensor and electronic components, in the same way there are Convolutional neural networks, which generate a model for the detection of restrictions, taking account for testing activities (validation and verification).

A restriction detector is made for the prevention of contagion cases in the health emergency and which emits an alarm sound for prevention by detecting mouth covers and physical distancing between people.

Keywords

Arduino, neural networks, sensor, crowd, chinstrap, contagion, emergency, model.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

MARCO INTRODUCTORIO

CAPÍTULO I: MARCO INTRODUCTORIO

1.1. INTRODUCCIÓN

El transcurso de este año es el reflejo de una nueva enfermedad, la cual es el COVID-19, afecta a la población con manifestaciones clínicas graves. Por tanto, se debe proporcionar una educación en base a los resultados de diferentes estudios sobre ambientes y contaminación. Es recomendable fomentar programas de prevención y promoción haciendo uso de herramientas que sean una aportación para la sociedad. Es adecuado para los ciudadanos evitar y controlar la propagación del virus, esto implica implementar la compra de vestimenta de protección, gafas anti-salpicadura, más guantes, así como capacitaciones al personal de salud en todos los aspectos relacionados, incluso el uso apropiado de tapabocas o mascarillas.

En la actualidad, el incremento de la tecnología ha hecho que el uso de sistemas y dispositivos sean implementados en los barrios de la ciudad, para alcanzar altos niveles de eficacia y eficiencia, que cuente con la información precisa y confiable; ahora tenemos como objetivo la prevención de casos de contagio en distintos lugares de la ciudad. La información es un recurso muy importante para las autoridades, debido a que administrar la información es situarlo como un recurso principal para la generación de información oportuna y apoyo en la toma de decisiones

En cuanto a la tecnología, los sistemas automatizados son una tecnología en ascenso y parte de las investigaciones actuales en ingeniería; ya que se enfoca en la implementación de la cibernética, el software y las telecomunicaciones, lo cual ha ido evolucionando, hoy podemos encontrar dispositivos de alarmas, de riego, de información, de recolección de datos, de electricidad, donde se hace el uso de componentes, como ser: Arduino, técnicas de análisis, redes neuronales, algoritmos neurocomputacionales, procesamiento de imágenes entre otros.

El presente trabajo es una propuesta para resolver el problema del distanciamiento en las aglomeraciones y la falta de uso de barbijos, de tal modo prevenir el contagio del coronavirus en la emergencia sanitaria, mediante redes neuronales y una plaqueta Arduino con su respectivo sensor y de esta manera captar las imágenes para procesar, analizar y detectar la distancia entre las personas y el uso de barbijos.

1.2. ANTECEDENTES

1.2.1. ANTECEDENTES TEÓRICOS

La COVID-19 es la enfermedad infecciosa causada por el coronavirus que se ha descubierto recientemente. Tanto este nuevo virus como la enfermedad que provoca eran desconocidos antes de que estallara el brote en Wuhan (China) en diciembre de 2019. Actualmente la COVID-19 es una pandemia que afecta a muchos países de todo el mundo.

En Bolivia, la salud y seguridad es una preocupación central de todos los bolivianos en sus diferentes estratos sociales. Estudios nos muestran los casos de contagio que son alrededor de 40.000, nos señalan que la cifra de infectados aumenta por día. Para la emergencia sanitaria, se realizó la promulgación de varios decretos debido al COVID-19, como ser:

- DECRETO 4245 (28 de mayo,2020): Tiene por objeto:
    a) Continuar con la cuarentena nacional, condicionada y dinámica hasta el 30 de junio de 2020, según las condiciones de riesgo en las jurisdicciones de las Entidades Territoriales Autónomas – ETAs.
    b) Iniciar las tareas de mitigación para la ejecución de los Planes de Contingencia por la Pandemia del Coronavirus (COVID-19) de las ETAs en el marco de la Ley No 602, de 14 de noviembre de 2014, de Gestión de Riesgos.

- DECRETO 4276 (26 de junio,2020): Ante el incremento del contagio comunitario y aumento de casos positivos del Coronavirus (COVID-19) en el territorio boliviano, el presente Decreto Supremo tiene por objeto ampliar el plazo de la cuarentena nacional, condicionada y dinámica.

Se tiene muchos factores para el contagio de coronavirus como ser: La falta de control de aglomeraciones en las calles, concientización en el uso apropiado del tapabocas (barbijo) de las personas y el gran contagio de la población al realizar su labor en esta emergencia sanitaria.

Un detector es aquel dispositivo capaz de detectar o percibir cierto fenómeno físico, tal como la presencia de humo proveniente de un incendio, la existencia de un gas en el aire y la presencia de un intruso en una vivienda.

El procesamiento digital de imágenes (PDI) consiste en la manipulación de los datos contenidos en la imagen para convertirlos en información útil. Dicho procesamiento puede realizarse en forma óptica, analógica o digital. El procesamiento de imágenes mediante medios de procesamiento digital de la información, tales como PC, FPGA (Field Programmable Gate Array), aplicaciones específicas sobre DSP (Digital Signals Processor); constituye lo que se denomina Procesamiento Digital de Imágenes (PDI) (González, y Woods, 2002).

Arduino es una placa hardware que está conformada por un microcontrolador reprogramable y pines que están unidos internamente a puerto E/S tanto digitales como análogos, los cuales se pueden conectar. La placa Arduino UNO es la más utilizada en el mercado, porque fue la primera en aparecer y una de las más económicas de todas.

1.2.2. TRABAJOS SIMILARES

Para la presente investigación se ha realizado un análisis de proyectos desarrollados en diferentes países, obteniendo información que se relaciona con el tema y con el objeto de investigación.

- Alarmas comunitarias, proyecto que fue realizado por el Gobierno Autónomo de La Paz (2014). Las alarmas comunitarias podrán ser activadas por los vecinos y usuarios titulares que tendrán un pulsador inalámbrico y por los usuarios secundarios, desde sus celulares. En ambos casos los policías de los módulos policiales y de las estaciones integrales policiales recibirán un mensaje de alerta. Cada alarma comunitaria cuenta con una sirena, un destellador (luz intermitente) y un altavoz.
- La tesis titulada "Sistema domótica de seguridad perimetral basado en Arduino", en este documento se explica la implementación del sistema domótico de seguridad que permita detectar la intrusión de personas ajenas al hogar, a lo largo del perímetro establecido, en el cual se emplea el módulo electrónico "ARDUINO UNO" que realiza un control de acceso para ingresar al interior del recinto, y utilizando la red celular mediante un Módulo GSM (SIM900) se envía alertas (Condori, 2016).
- La tesis titulada "Sistema de seguridad perimetral programable inteligente", en este documento explica cómo fue desarrollado el sistema de seguridad perimetral empleando tecnología de bajo coste, como es el microcontrolador Arduino. El producto final permite disponer de un sistema de seguridad instalable en multitud de lugares que ofrece una muy buena relación de prestaciones y costes (Brun, 2014).
- La tesis titulada "Sistema para la detección de engaños basado en redes neuronales y Arduino", el proyecto se basa en una placa "Arduino Uno", un sensor que mida el pulso y

ritmo cardiaco, un dispositivo de Bluetooth, todos estos implementados en una placa "Arduino Uno" controlados bajo el sistema operativo Android el cual interpreta los datos obtenidos a través de redes neuronales analizando los datos enviados desde la placa "Arduino Uno" y mostrarlos a través de la aplicación (Romero, 2016).

- El proyecto Fin de Máster en Ingeniería Informática para la Industria titulado "Reconocimiento de imágenes utilizando redes neuronales artificiales". Este trabajo describe el proceso de extracción de patrones característicos de imágenes, mediante la ayuda de Redes Neuronales Artificiales. La información de la Red Neuronal junto con datos adicionales de las imágenes, serán almacenados en una base de datos y consumidos por un servicio web (García, 2013).
- La tesis doctoral titulada "Algoritmos de aprendizaje neurocomputacionales para su implementación hardware", tiene como objetivo avanzar en el conocimiento científico y tecnológico necesario para el diseño y desarrollo de modelos neurocomputacionales en dispositivos hardware específicos (microcontroladores y FPGAs), con el fin de permitir su utilización en aplicaciones de sistemas en tiempo real y redes de sensores (Ortega, 2015).

1.3. PLANTEAMIENTO DEL PROBLEMA

1.3.1. PROBLEMA CENTRAL

Tomando en cuenta los antecedentes y realizando un análisis de los mismos, el problema de investigación es el siguiente:

¿Es posible prevenir la aglomeración de personas y la falta de uso de barbijos mediante el procesamiento de imágenes y un dispositivo en esta emergencia sanitaria?

1.3.2. PROBLEMAS SECUNDARIOS

A continuación, se muestran los problemas secundarios.

- Los casos de COVID-19 incrementan por día, debido a las aglomeraciones en las calles sin tomar en cuenta el distanciamiento físico en la emergencia sanitaria.
- Los dispositivos de grabación de la policía boliviana en diferentes lugares de la ciudad simplemente graban las aglomeraciones, peleas, riñas, marchas, sin realizar la prevención de contagios en la emergencia sanitaria.
- Mayor cantidad de contagios de coronavirus en la ciudadanía debido a que la población realiza su labor en la emergencia sanitaria.
- La discrepancia y la falta de información para la población ocasionan el desuso del cubre bocas (barbijo) en las actividades que se realizan, aumentando el riesgo de infección del virus en la emergencia sanitaria.

1.4. DEFINICIÓN DE OBJETIVOS

1.4.1. OBJETIVO GENERAL

Implementar un detector de aglomeraciones y uso de barbijo a través del procesamiento de imágenes y un dispositivo en base a Arduino para prevenir mayores casos de contagio en esta emergencia sanitaria.

1.4.2. OBJETIVOS ESPECÍFICOS

- Desarrollar un detector de restricciones para el control de aglomeraciones en esta emergencia sanitaria.
- Capturar imágenes mediante la cámara para remitir señales al dispositivo en plaqueta Arduino y disminuir la probabilidad de contagio del virus.
- Prevenir mayores casos de contagio en la emergencia sanitaria con el dispositivo en Arduino, el cual enviará una alarma sonora tanto para la prevención de multitudes y la falta de uso de barbijo.

- Modelar un detector de restricciones para la obtención de datos en un perímetro, con el dispositivo en base a Arduino y procesamiento de imágenes en la emergencia sanitaria.
- Realizar el procesamiento de imágenes mediante el entrenamiento de las redes neuronales, para obtener datos correctos en el dispositivo en Arduino.

1.5. HIPÓTESIS

Mediante el detector de aglomeraciones y uso de barbijos, elaborado en base al procesamiento de imágenes, Arduino y redes neuronales, se podrá prevenir la concurrencia de personas y la falta de uso de barbijos por medio de alertas en la emergencia sanitaria con una precisión del 90%.

1.5.1. OPERACIONALIZACIÓN DE VARIABLES

- **Variable independiente:** Detector de aglomeraciones y uso de barbijos, elaborado en base al procesamiento de imágenes, Arduino y redes neuronales.
- **Variables intervinientes:** Redes neuronales, procesamiento de imágenes, Arduino.
- **Variable dependiente:** Alertas para prevenir la concurrencia de personas y la falta de uso de barbijos en la emergencia sanitaria con una precisión del 90%

1.6. JUSTIFICACIÓN

1.6.1. JUSTIFICACIÓN ECONÓMICA

Sobre el punto de vista económico, el beneficio que tenemos es salvaguardar la seguridad y la salud, la cual es una condición sobresaliente para alcanzar el desarrollo de los países en la pandemia, en esta perspectiva, el gasto para la prevención de los casos de contagio de COVID-19 en La Paz viene a ser una inversión, que involucra a los ciudadanos en general y lograr la disminución de los casos de contagio mediante el detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales.

Los costos que involucran el desarrollo del dispositivo en Arduino y componentes, son los siguientes: Módulo electrónico ARDUINO UNO 60Bs, Sensor Buzzer Bs.10, Componentes para conexión Bs 50.

1.6.2. JUSTIFICACIÓN SOCIAL

La implementación del detector de restricciones para la prevención de casos de contagio en emergencia sanitaria basado en Arduino y redes neuronales; con un análisis de involucrados se determinó que este trabajo de investigación va dirigido a los siguientes participantes:

- Grupos: Brigada vecinal, brigada policial, sistema de salud.
- Personas: Policía, vecinos, funcionarios de la alcaldía, médicos.
- Instituciones: Ministerio de Gobierno, Policía Boliviana, Ministerio de Educación, y Hospitales.

Donde se realizó la clasificación de dichos participantes:

- Beneficiarios: Médicos, policía, personas de distinta edad.
- Implementadores: Brigada policial, alcaldía.
- Decisores: Ministerio de gobierno, vecinos, ministerio de Educación.
- Financiadores: Ministerio de gobierno, alcaldía.

1.6.3. JUSTIFICACIÓN CIENTÍFICA

Se tiene la expectativa que con el desarrollo de este proyecto de tesis se pueda contribuir a la investigación científica, para disminuir la propagación del virus con el detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, con las siguientes tecnologías:

- El procesamiento de imágenes: Es un proceso para la adquisición de la imagen digital sobre las aglomeraciones y la ausencia del barbijo, la cual será percibida mediante una

cámara u otro dispositivo como una detección de imágenes en escala de grises, el uso de la técnica de división y difusión, la cual subdivide la imagen inicialmente en un conjunto de regiones disjuntas, dentro de las cuales, se volverá a realizar una subdivisión o bien una fusión entre ellas.

- Redes neuronales: Son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. Después del procesamiento de imágenes se procede a el entrenamiento y predicción o clasificación mediante un modelo generado en base a las imágenes recolectadas.
- El diseño y elaboración del dispositivo en base a Arduino: Se efectuará con el Arduino UNO, un sensor, protoboard y componentes; los cuales tendrán una conexión Arduino con Python, empleando el puerto serie y la librería PySerial. La comunicación por puerto serie se realizará por cable, de esta manera recibir y enviar información; así prevenir las aglomeraciones y la falta del uso de la tapa bocas (barbijo).

1.7. ALCANCES Y LÍMITES

1.7.1. ALCANCES

Tenemos los siguientes límites:

- La tesis en desarrollo será un software funcional.
- El detector de restricciones para casos de contagio en la emergencia sanitaria realizará el procesamiento de imágenes, para prevenir aglomeraciones y la falta de uso de barbijos.
- El detector de restricciones para casos de contagio efectuará el monitoreo para el control de aglomeraciones y el uso de barbijo en esta emergencia sanitaria.
- Emitirá una alarma sonora para la prevención de multitudes y la falta de uso de barbijo.

- Se realizará la prevención de casos de contagio por medio del detector de restricciones, en un determinado perímetro en la emergencia sanitaria.
- Procesamiento de imágenes mediante el entrenamiento de las redes neuronales.

1.7.2. LÍMITES

A continuación, se presentan lo que el software no hace o no considera hacer:

- El detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales no disminuirá los casos de COVID-19, solo facilitará la detección para una futura prevención.
- Solo se realizarán experimentos o pruebas con el dispositivo en un perímetro determinado.
- El dispositivo para la prevención de casos de contagio en la emergencia sanitaria será un detector de restricciones basado en Arduino y redes neuronales, por tanto, no se realizará la instalación en los postes.
- No se realizará ninguna prueba con cámaras de una estación policial u otro establecimiento.

1.8. APORTES

1.8.1. TEÓRICO

La investigación tendrá como resultado impulsar al estudio de problemas en la ciudadanía con ayuda de dispositivos autómatas y en tiempo real que trabajan mediante el procesamiento de imágenes, redes neuronales y dispositivos en base a Arduino, los cuales hoy en día no son muy utilizados.

1.8.2. PRÁCTICO

El aporte práctico será el desarrollo e implementación de un detector de aglomeraciones y uso del barbijo con la ayuda del procesamiento de imágenes, redes neuronales y el Arduino, los cuales estarán conectados mediante el puerto serie y la librería PySerial, para prevenir mayores casos de contagio en esta emergencia sanitaria.

1.9. METODOLOGÍA

El presente trabajo se basará en el tipo de investigación exploratorio-experimental utilizando el método de investigación inductiva. Para la parte de desarrollo, en la elaboración manejaremos el Modelo en V o de cuatro niveles.

Para la ejecución del presente proyecto se aplica la metodología inductiva, ya que desde esta nos permite obtener conclusiones generales a partir de premisas particulares. En esta metodología pueden distinguirse cuatro pasos esenciales: la observación de los hechos para su registro; la clasificación y el estudio de los hechos; la derivación inductiva que parte y permite llegar a una generalización, y la contrastación (Sampieri, 2003).

El modelo en V es un proceso que representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto. Describe las actividades y resultados que han de ser producidos durante el desarrollo del producto. La parte izquierda del modelo en V representa la descomposición de los requisitos y la creación de las especificaciones del sistema. El lado derecho de la metodología en V representa la integración de partes y su verificación (Inteco, 2009).

Figura 1.1: Modelo en V.
Fuente: (García, 2018).

CAPÍTULO II

MARCO TEÓRICO

CAPÍTULO II: MARCO TEÓRICO

En este punto se realiza la descripción de distintos métodos y técnicas en la implementación del detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales.

2.1. EMERGENCIA SANITARIA

2.1.1. COVID-19

Durante todo este tiempo, conocemos la pandemia de COVID-19 en Bolivia, la cual es parte de la pandemia mundial debido a la enfermedad del coronavirus 2019 (COVID-19). Se debe tomar las precauciones adecuadas y estar bien informado para protegerte y cuidar de quienes te rodean. Según los organismos de salud pública para evitar la propagación de la COVID-19, se debe seguir los siguientes pasos.

- Lavarse las manos con frecuencia. Usa agua y jabón o un desinfectante de manos a base de alcohol.
- Mantener una distancia de seguridad con personas que tosen o estornudan.
- Utilizar mascarilla cuando no sea posible mantener el distanciamiento físico.
- No tocarse los ojos, la nariz ni la boca.
- Cuando tosas o estornudes, cubrirse la nariz y la boca con el codo flexionado o con un pañuelo.

2.1.2. USO DE LA MASCARILLA

Se llama mascarillas médicas a las mascarillas quirúrgicas o para procedimientos que son planas o llevan pliegues; se fijan a la cabeza mediante cintas que se sostienen de las orejas o rodean la cabeza. Su desempeño se pone a prueba mediante un conjunto normalizado de

métodos para comprobar el equilibrio entre una gran capacidad de filtración, la respirabilidad adecuada y, a veces, la resistencia a la penetración por líquidos corporales (OMS, 2020).

El uso adecuado de las mascarillas es el siguiente:

- Ajustarse la mascarilla quirúrgica lo más pegado a la cara.
- Colocar las cintas detrás de las orejas firmemente.
- Moldear la banda metálica alrededor del tabique nasal.

No toque la mascarilla durante su uso. Si debiera hacerlo, se debe lavar las manos antes y después de su manipulación.

El uso de mascarillas forma parte de un conjunto integral de medidas de prevención y control que pueden limitar la propagación de determinadas enfermedades respiratorias causadas por virus, en particular la COVID-19. Sirven también para proteger a las personas sanas (cuando estas las emplean al entrar en contacto con una persona infectada) o para el control de fuentes (si una persona infectada la utiliza para no contagiar a otros) (OMS, 2020).

2.1.3. DISTANCIAMIENTO FÍSICO

La medida más eficaz contra la propagación del SARS-CoV-2 tiene un nombre: distanciamiento físico, que, según los expertos, es un término más apropiado que el comúnmente utilizado distanciamiento social. Este término lo engloba todo, desde mantener la distancia física hasta no salir de casa. Con un poco de paciencia, buena voluntad y cooperación, todos podemos aportar nuestro granito de arena (Braun, 2020).

El distanciamiento físico significa estar físicamente separado. La OMS recomienda mantener una distancia de al menos un metro con los demás. Es una medida general que todas las personas deberían adoptar incluso si se encuentran bien y no han tenido una exposición conocida a la COVID-19 (OMS, 2020).

2.2. CIUDADES INTELIGENTES E INTERNET DE LAS COSAS

2.2.1. CIUDADES INTELIGENTES

Ante este panorama, en los años 90 toma fuerza la noción de Smart Cities o Ciudades Inteligentes se brindaban oportunidades a los gobiernos para la resiliencia de los espacios urbanos a partir de inversiones tecnológicas que contribuyan al crecimiento económico, a prevenir, mitigar y reparar las intervenciones que rompen el equilibrio con el medio ambiente y por supuesto, tendientes a mejorar la calidad de vida de sus pobladores (Fernández Güel, 2015).

Figura 2.1: Algunos elementos de una ciudad inteligente y sostenible.
Fuente: Elaboración propia a partir de Alvarado (2018).

Las Ciudades Inteligentes se definen como sistemas de personas que interactúan y que utilizan la energía, materiales y financiación para generar el desarrollo económico sostenible, flexible y una mayor calidad de vida (Cámara de Comercio de Medellín para Antioquia, 2012).

2.2.2. INTERNET DE LAS COSAS

Cisco en el 2013 define al IoT como la interacción entre los seres humanos y los objetos en diversas aplicaciones, tales como la fabricación, la logística, el sector de los servicios, la

agricultura, el reciclaje, gestión ambiental, casas inteligentes y otros, añadiendo una nueva forma de conectividad de los "objetos" a Internet (Alvear et al., 2017).

El IoT pretende que los datos que se obtengan por medio de sensores puedan ser monitoreados de forma remota, de esta manera generar bases de datos con las cuales se pueda realizar estadísticas, tendencias, probabilidades, todo esto en función de comportamientos y actividades de personas o ambientes donde se puede aplicar sistemas electrónicos completos y obtener estos datos en busca de mejorar procesos o determinar puntos débiles (Alvear et al., 2017).

El Internet de las cosas es la tecnología que permite la interconexión de dispositivos físicos, vehículos, edificios y otros elementos combinados con electrónica, software, sensores, actuadores y redes de conectividad para recopilar e intercambiar datos para su análisis.

2.3. APRENDIZAJE PROFUNDO

El aprendizaje profundo o Deep Learning es un subcampo del aprendizaje automático o Machine Learning, que, a su vez, es un subcampo de la inteligencia artificial. El principal objetivo de la IA es ofrecer algoritmos y técnicas para resolver problemas que las personas pueden resolver de forma intuitiva o automatizada. Un ejemplo es como la inteligencia artificial puede interpretar el contenido de una imagen y extraer conclusiones sobre esta. El aprendizaje profundo se divide en diferentes ramas como redes neuronales, redes neuronales convolucionales o redes neuronales recurrentes (Oliva, 2018).

2.3.1. NEURONA ARTIFICIAL

La neurona artificial es una unidad procesadora con cuatro elementos funcionales (Lara, 2012).

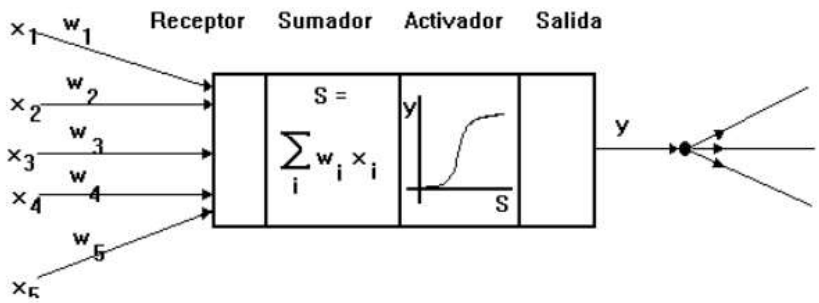


Figura 2.2: Esquema de una neurona artificial.
Fuente: (Lara, 2012).

El elemento receptor, a donde llegan una o varias señales de entrada x_i, que generalmente provienen de otras neuronas y que son atenuadas o amplificadas cada una de ellas con arreglo a un factor de peso w_i que constituye la conectividad entre la neurona fuente de donde provienen y la neurona de destino en cuestión.

El elemento sumador, que efectúa la suma algebraica ponderada de las señales de entrada, ponderándolas de acuerdo con su peso, aplicando la siguiente expresión:

$$S = \sum w_i x_i \quad \text{Ecuación 2.1.}$$

El elemento de función de activación aplica una función no lineal de umbral (que frecuentemente es una función escalón o una curva logística) a la salida del sumador para decidir si la neurona se activa, disparando una salida o no.

El elemento de salida es el que produce la señal, de acuerdo con el elemento anterior, que constituye la salida de la neurona.

2.3.2. REDES NEURONALES

La idea de las Redes Neuronales es, como su propio nombre indica, tratar de imitar el cerebro humano. A pesar de que no se conoce exactamente cómo funciona el cerebro humano, sí que se saben ciertas características como la estructura que tiene e inspirándose vagamente en estas observaciones surgen estos algoritmos. Concretamente el algoritmo consiste en un

conjunto de unidades denominadas neuronas conectadas entre sí para transmitir información de unas a otras. Cuando los datos/información de entrada atraviesan esta red de neuronas, se acaba produciendo un resultado o unos valores de salida que son los que nos interesan. De este modo se puede entender una red neuronal como una gran función que produce una salida dependiente de unos datos de entrada, pero dicha función para ello se apoya en multitud de subfunciones que la componen. Es decir, se puede ver una red neuronal como una función de funciones que permiten a la función principal computar y capturar potentes relaciones en los datos (Erroz, 2019).

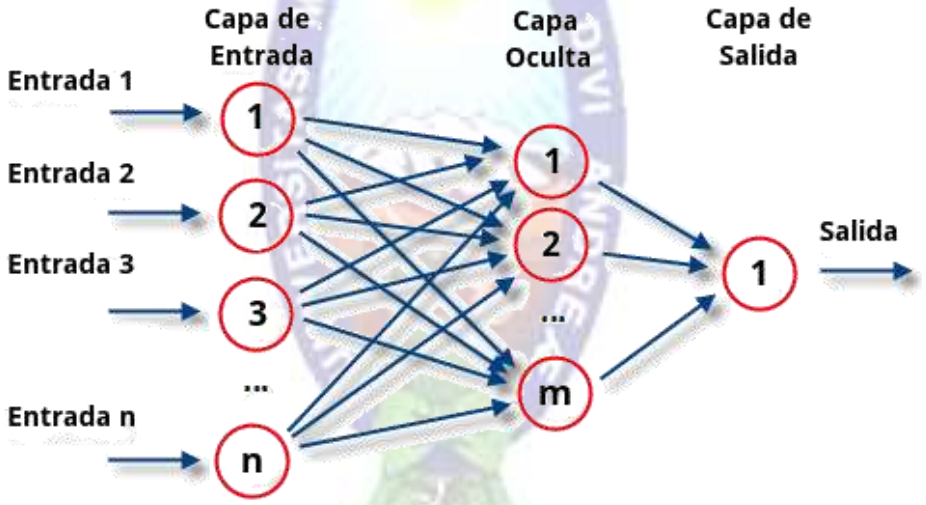


Figura 2.3: Red neuronal de una única capa oculta.
Fuente: (Erroz, 2019).

Las neuronas tienen una estructura simple, ya que cada una tiene un vector multicapa que recibe el vector de datos de entrada $\vec{x}$, un vector de pesos sinápticos $\vec{w}$ y la función de activación $f(x)$. Entonces la salida de una neurona es: $y = f(\vec{w}, \vec{x})$.

En las Redes Neuronales cada neurona actúa como un clasificador con sus propios parámetros y su función de activación. Lo que cambia es que, al haber una interconexión de

neuronas, los clasificadores ahora toman como entrada la salida de otros. Además, ahora debemos entrenar la red en su conjunto y no cada clasificador por separado (Erroz, 2019).

2.3.3. FUNCIONES DE ACTIVACIÓN

Función Escalón: La función de activación escalón se asocia a neuronas binarias en las cuales, cuando la suma de las entradas es mayor o igual que el umbral de la neurona. La activación es 1; si es menor, la activación es 0 (o −1) (Ponce, 2010).

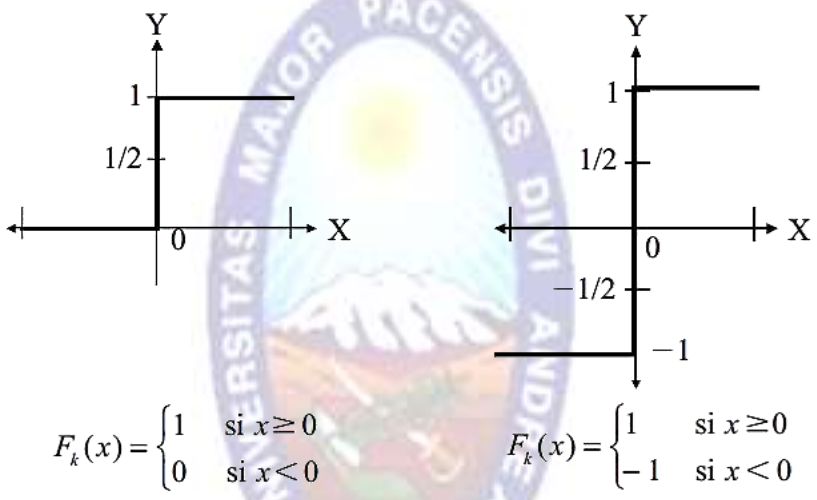


Figura 2.4: Función de activación escalón.
Fuente: (Ponce, 2010).

Función Lineal y mixta: La función lineal o identidad responde a la expresión $Fk\ (u) = u$. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la función se define como 0 (o −1). Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de entrada está comprendida entre los dos límites, entonces la activación es 1. Si la suma de entrada está comprendida entre ambos límites, superior e inferior, entonces la activación se define como una función lineal de la suma de las señales de entrada (Ponce, 2010).

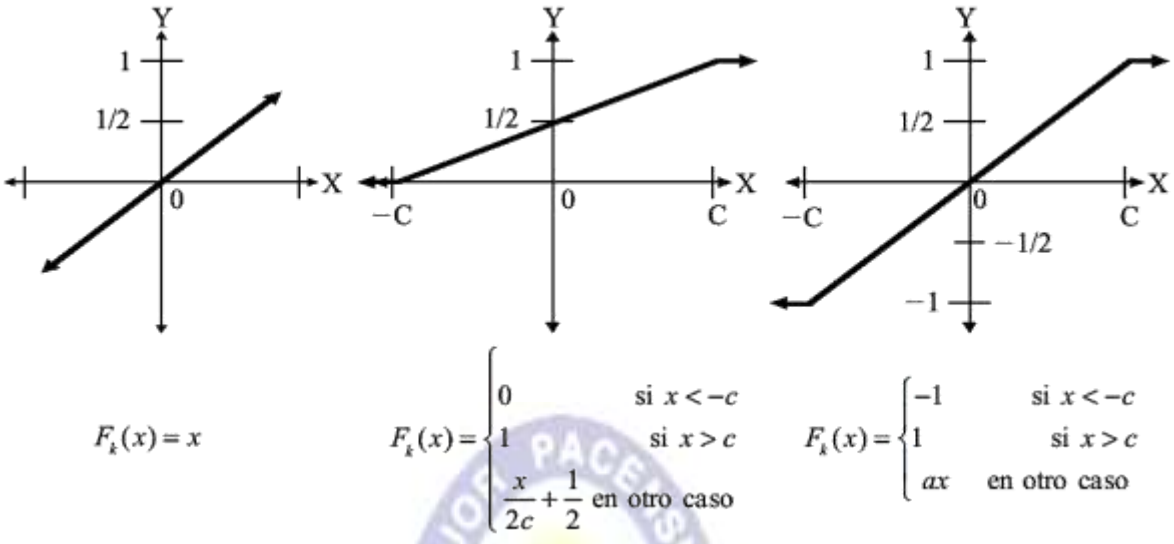


Figura 2.5. Función de activación lineal y mixta.
Fuente: (Ponce, 2010).

Función tangente hiperbólica: La función de activación tangente hiperbólica se emplea en los casos que presentan variaciones suaves de valores positivos y negativos de la señal a clasificar. Es una de las funciones más empleadas en entrenamientos supervisados, como en el caso del entrenamiento de retropropagación del error (Ponce, 2010**).**

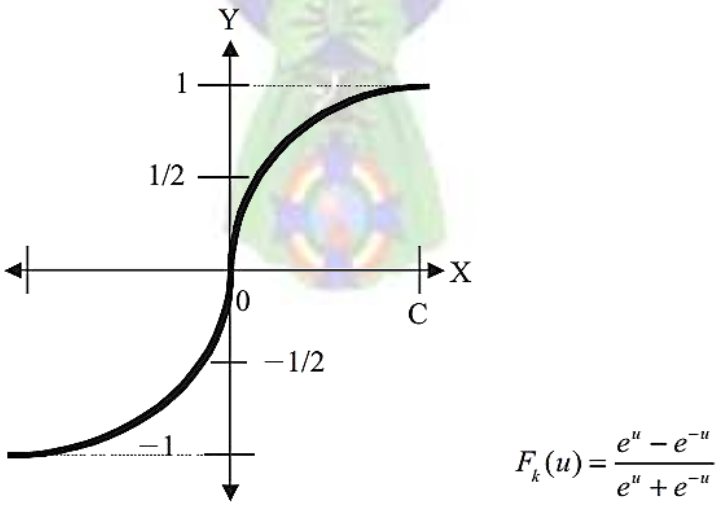


Figura 2.6: Función de tangente hiperbólica.
Fuente: (Ponce, 2010).

Función sigmoidal: Con la función sigmoidal el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que, en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal es que su derivada siempre es positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando x = 0 (Ponce, 2010).

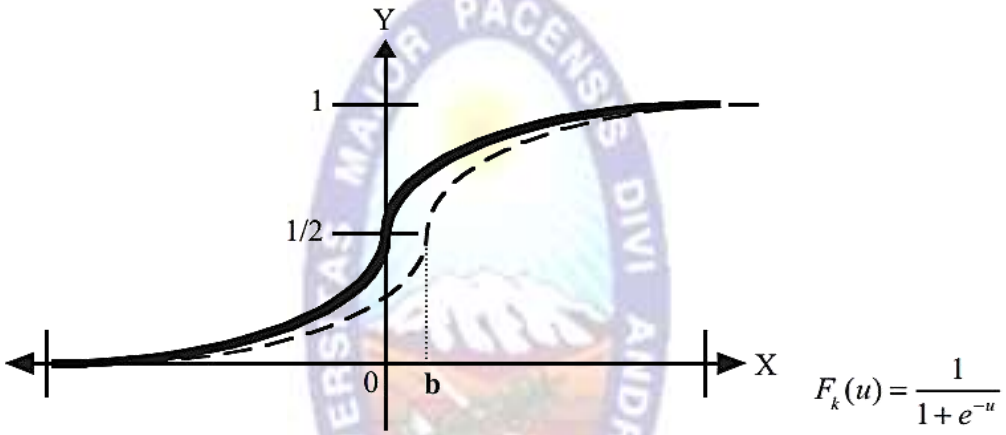


Figura 2.7: Función Sigmoidal.
Fuente: (Ponce, 2010).

2.3.4. TIPOS DE APRENDIZAJE

Existen dos tipos de aprendizaje: el supervisado y no supervisado.

- En el aprendizaje supervisado antes de realizar una clasificación para una imagen se realiza una preparación de un conjunto de datos perfectamente etiquetado. En visión por computador el conjunto de imágenes tiene la finalidad de que el clasificador aprenda las características del objeto a clasificar. El clasificador es capaz de encontrar los objetos clasificados en otras imágenes diferentes. Los algoritmos son Perceptron, Adaline, Madaline y Backpropagation.

- En el aprendizaje no supervisado, existe un conjunto de datos, pero el cual no está etiquetado. El clasificador es el encargado de realizar la clasificación a partir de las características que reciba en la entrada. Algunos de los algoritmos de clasificación utilizados en este tipo de aprendizaje son K-means o PCA (Principal Component Analysis) entre otros.

2.4. RED NEURONAL CONVOLUCIONAL

Las redes neuronales convolucionales están formadas por diferentes capas, donde las primeras capas se encargan de extraer características como los bordes o esquinas de una imagen. Una vez detectados estos bordes son utilizados para detectar formas en las capas posteriores. Cuando se detectan las formas se procede a detectar las características de alto nivel como puede ser una rueda en una imagen donde aparece un coche. Las últimas capas están totalmente conectadas y son las encargadas de dar una predicción a partir de estas últimas características extraídas (Oliva, 2018).

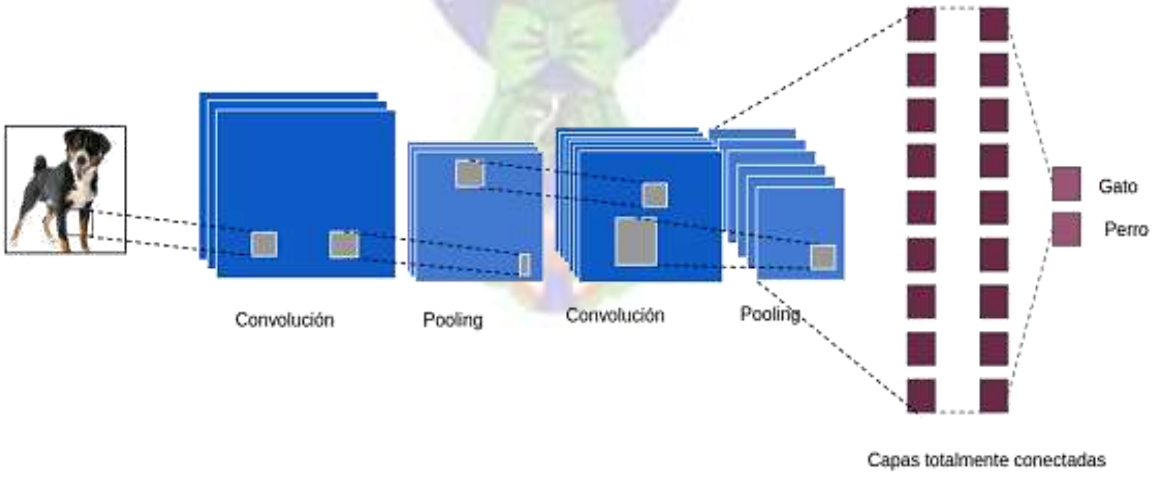


Figura 2.8: Red neuronal Convolucional.
Fuente: (Oliva, 2018).

Este tipo de redes surgen debido a la importancia de la visión por computador en el mundo del Aprendizaje Profundo. Las Redes Neuronales Convolucionales están diseñadas para trabajar sobre imágenes. Una de las claves de este algoritmo es la conservación de la espacialidad durante el proceso, ya que en todo momento mantienen la información sobre la localidad de las activaciones que van calculando gracias a trabajar con matrices en lugar de con datos por separado. Esto último es esencial cuando se trata de imágenes ya es muy importante no solo la relación existente entre píxeles, sino también la localidad de los mismos, que se pierde completamente si no lo hacemos de esta forma.

2.4.1. CONVOLUCIÓN

La operación de convolución consiste en superponer un filtro sobre una imagen e ir desplazando/deslizando el filtro sobre la misma hasta haber recorrido así todas las regiones de la imagen (todos los píxeles de la imagen han de haber sido visitados por el filtro en algún momento durante el proceso). Por cada desplazamiento del filtro sobre la imagen se calcula el valor resultante de la suma de todas las multiplicaciones de cada píxel por el valor correspondiente a la posición del filtro que superpone a ese píxel (Erroz, 2019).

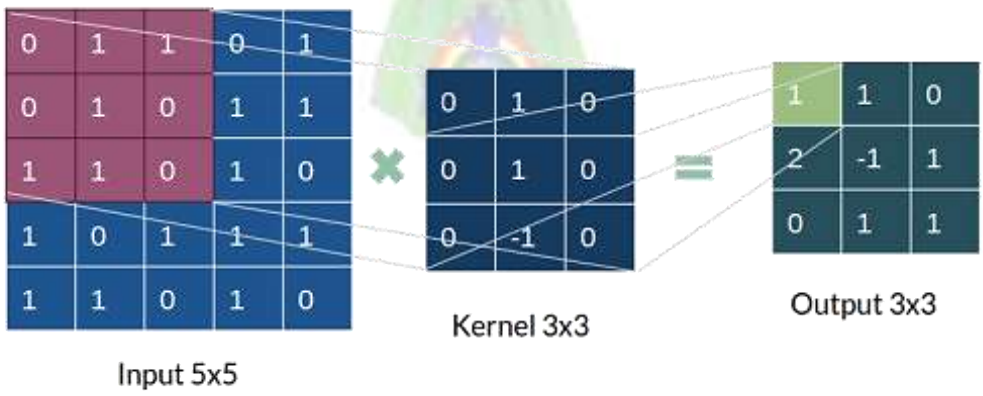


Figura 2.9: Aplicación de una operación de Convolución.
Fuente: (Oliva, 2018).

En la figura se tiene un producto escalar entre el kernel o filtrado y la matriz de la imagen de entrada. Cada imagen puede ser considerada como una matriz de números. Hay que tener en cuenta que cada imagen, puede ser representada como una matriz con de valores por cada pixel. Estos píxeles serán las entradas de la red (Casas, 2017).

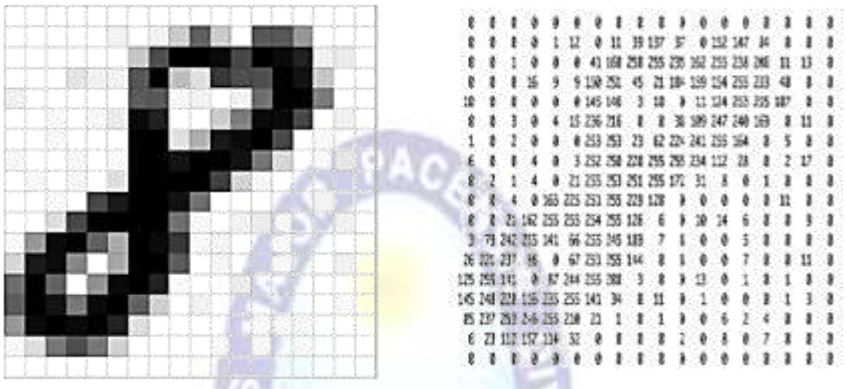


Figura 2.10: Representación de una imagen como una matriz.
Fuente: (Casas, 2017).

Se considera una imagen 5x5 cuyos elementos de la matriz solo pueden ser 0 o 1. También se considera otra matriz de 3x3 como la siguiente:

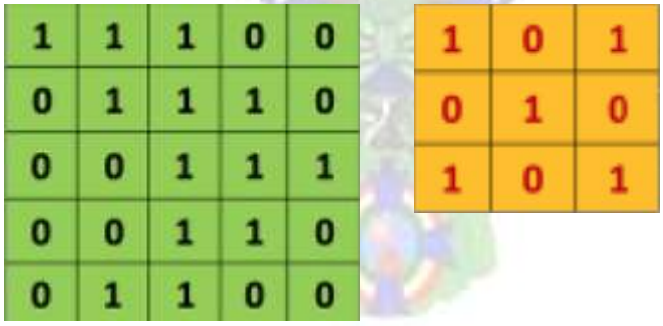


Figura 2.11: Representación de matrices.
Fuente: (Casas, 2017).

Entonces, la convolución entre la imagen de 5x5 y la matriz 3x3 puede ser calculada como:

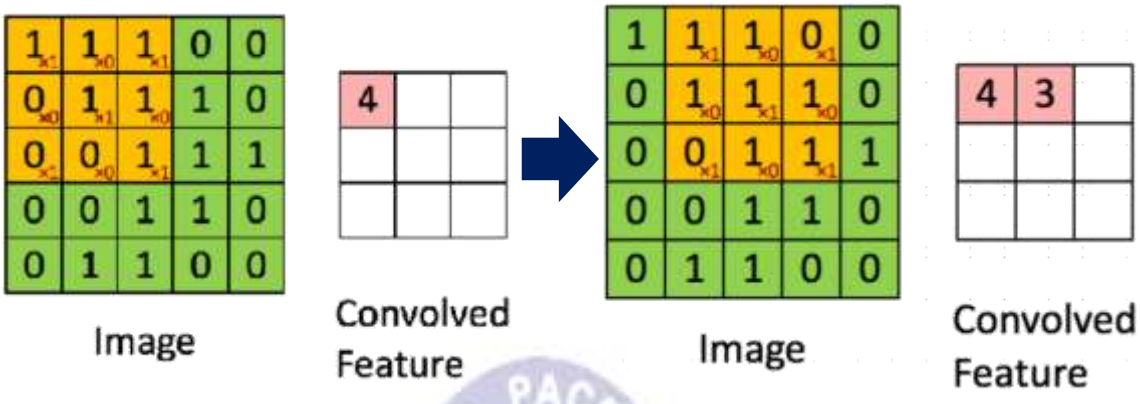


Figura 2.12: Convolución.
Fuente: (Casas, 2017).

La idea de la convolución es desplazar la matriz naranja sobre la imagen original (verde) con un salto de un pixel y para cada posición se calcula la multiplicación entre las dos matrices y se suman las salidas de la multiplicación para obtener un entero (de color rosa). Hay que tener en cuenta que la matriz 3x3 solo ve una parte de la imagen de entrada en cada paso. En terminología de las redes convolucionales, la matriz 3x3 se conoce como filtro o kernel y la matriz formada al realizar la convolución (rosa) se conoce como Convolved feature o Feature map. Es importante tener en cuenta que los filtros actúan como detectores de características de la imagen de entrada (Casas, 2017).

2.4.1.1. FILTRADO ESPECIAL

Podemos realizar operaciones como detección de bordes, nitidez y desenfoque simplemente cambiando los valores numéricos de nuestra matriz de filtro antes de la operación de convolución, esto significa que diferentes filtros pueden detectar diferentes características de una imagen, por ejemplo, bordes, curvas, entre otros.

Los filtros espaciales tienen como objetivo modificar la contribución de determinados rangos de frecuencias de una imagen. El término espacial se refiere al

hecho de que el filtro se aplica directamente a la imagen y no a una transformada de la misma, es decir, el nivel de gris de un pixel se obtiene directamente en función del valor de sus vecinos.

Responden a la siguiente ecuación:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t) \quad \text{Ecuación 2.2.}$$

Dónde:

$f(x+s, y+t)$: Es el valor de los píxeles del bloque seleccionado.

$w(s,t)$: Coeficientes que se aplicarán al bloque (máscara).

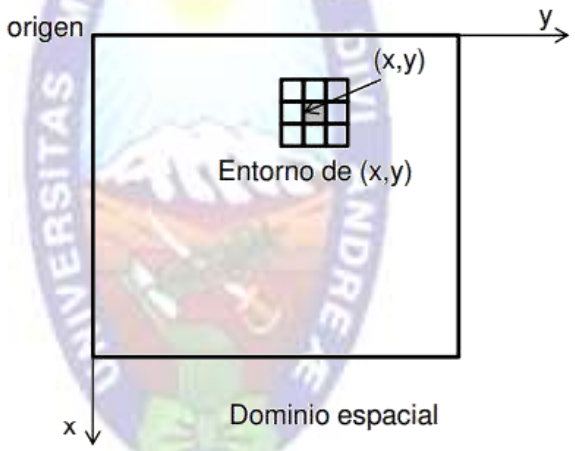


Figura 2.13: Filtros de dominio del espacio.
Fuente: (Casas, 2017).

Los filtros espaciales pueden clasificarse basándose en su linealidad: filtros lineales y filtros no lineales. A su vez los filtros lineales pueden clasificarse según las frecuencias que dejen pasar: los filtros paso bajo atenúan o eliminan las componentes de alta frecuencia a la vez que dejan inalteradas las bajas frecuencias; los filtros paso alto atenúan o eliminan las componentes de baja frecuencia con lo que agudizan las componentes de alta frecuencia; los filtros paso banda eliminan regiones elegidas de recurrencias intermedias (laboratorio I2.31, 2018).

2.4.2. CAPAS DE AGRUPAMIENTO

Extraídas las características desde una capa de convolución la siguiente capa es el agrupamiento o Pooling. La funcionalidad es reducir las dimensiones de la salida de la capa convolucional. Hay diferentes maneras de realizar esta reducción: Max Pooling, se desliza un filtro de tamaño $m\ x\ m$ a través de la matriz de entrada y se selecciona el valor más grande de dicha matriz. Average Pooling, igual que en la anterior, se desliza un filtro, pero el resultado es la media de todos los elementos de la matriz de entrada (Oliva, 2018).

Figura 2.14: Aplicación MaxPooling con stride.
Fuente: (Oliva, 2018).

Para esta capa también se aplica un stride cómo en la capa de convolución. El número de canales no se reduce, únicamente el número de dimensiones.

2.4.3. UNIDAD LINEAL RECTIFICADA

Unidad lineal rectificada (ReLU) es una función de activación que se encarga de truncar los valores negativos a 0. Normalmente es aplicada en la salida de una capa convolucional con la finalidad de introducir la no linealidad en un sistema que ha calculado operaciones lineales durante la capa convolucional (Oliva, 2018).

$$f(x) = \max(x, 0) \quad \text{Ecuación 2.3.}$$

Cuando procesamos una imagen, cada capa de convolución debe capturar algún patrón en la imagen y pasarla en la siguiente capa de convolución. Los valores negativos no son

importantes en el procesamiento de imágenes y, por lo tanto, se establecen en 0. Pero los valores positivos después de la convolución deben pasar a la siguiente capa. Es por eso que ReLU se está utilizando como una función de activación. Si utilizamos Sigmoidal o Tangente, la información se pierde ya que ambas funciones modificarán las entradas a un rango muy cerrado.

2.4.4. CONEXIÓN DE CAPAS

Las últimas capas de la red convolucional actúan como clasificador. La función de estas capas son las mismas que se explican en la sección de Redes Neuronales. En estas últimas capas es habitual aplicar la técnica de Dropout (expulsión) que desactiva un número de neuronas para evitar el sobreajuste (Oliva, 2018).

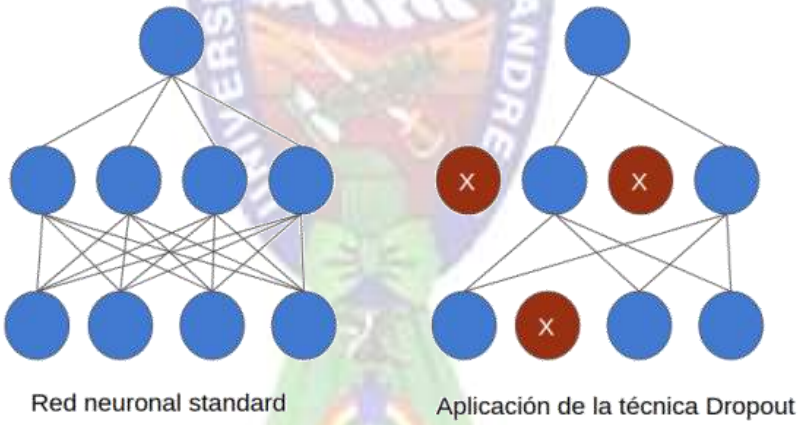


Figura 2.15: Aplicación de la técnica de expulsión.
Fuente: (Oliva, 2018).

En capas convolucionales tradicionales, A es un conjunto de neuronas en paralelo, que todas obtienen las mismas entradas y calculan diferentes características.

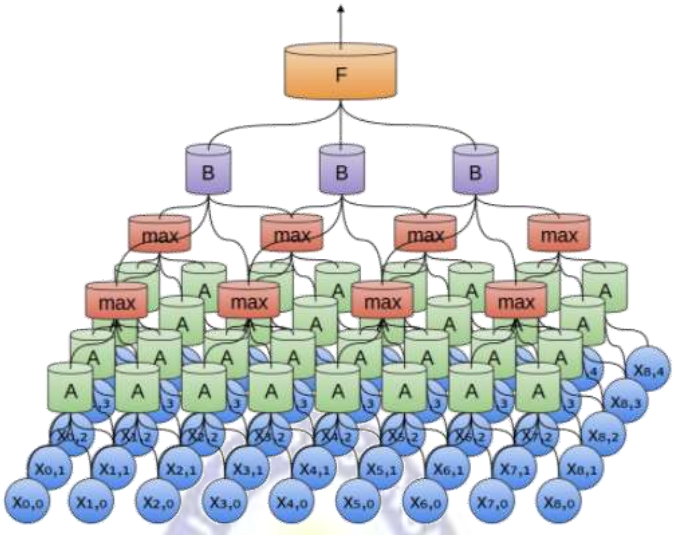


Figura 2.16: Ejemplo de una red neuronal convolucional 2D.
Fuente: (Colah, 2014).

2.4.5. ENTRENAMIENTO

Como se explicaba anteriormente, las capas de convolución y agrupamiento actúan como extractores de características de la imagen de entrada, mientras que la conexión de capas actúa como un clasificador. Todo el proceso de entrenamiento de la red convolucional puede resumirse en los cinco pasos (Casas, 2017).

- El primer paso, se inicializan los filtros y parámetros pesos con valores aleatorios.
- El segundo paso, la red toma una imagen de entrenamiento como entrada, pasa por el segundo paso, movimiento hacia adelante, (convolución y operaciones de agrupamiento en la conexión de capas) y encuentra las probabilidades de salida para cada clase.
- En el tercer paso, se calcula el error total en la salida de la red.

- El cuarto paso usa la predicción para calcular el gradiente del error con respecto a los pesos en la red y usa el gradiente descendente para actualizar todos los valores de los filtros y pesos y parámetros para minimizar el error de la salida.
- El quinto paso repite los pasos del 2 al 4 con todas las imágenes en el conjunto de entrenamiento. Los pasos anteriores entrenan a la neuronal convolucional, esto significa que todos los pesos y parámetros han sido optimizados para clasificar correctamente las imágenes del conjunto de entrenamiento.
- Cuando una nueva imagen entra a la red convolucional, la red pasaría por el paso de movimiento hacia adelante y emitirá una probabilidad para cada clase. Si el conjunto de entrenamiento es lo suficientemente grande para que la red pueda clasificar de forma correcta las imágenes de entrada.

2.5. PROCESAMIENTO DE IMÁGENES

El primer paso en el proceso es obtener la imagen digital. Por tanto, se necesitan sensores y la capacidad para digitalizar la señal producida por el sensor. Una vez que la imagen digitalizada ha sido obtenida, el siguiente paso consiste en el pre procesamiento de dicha imagen. El objetivo del pre procesamiento es mejorar la imagen de forma que cumpla con el objetivo final (Oliva, 2018).

Para el procesamiento de imágenes tenemos los siguientes pasos:

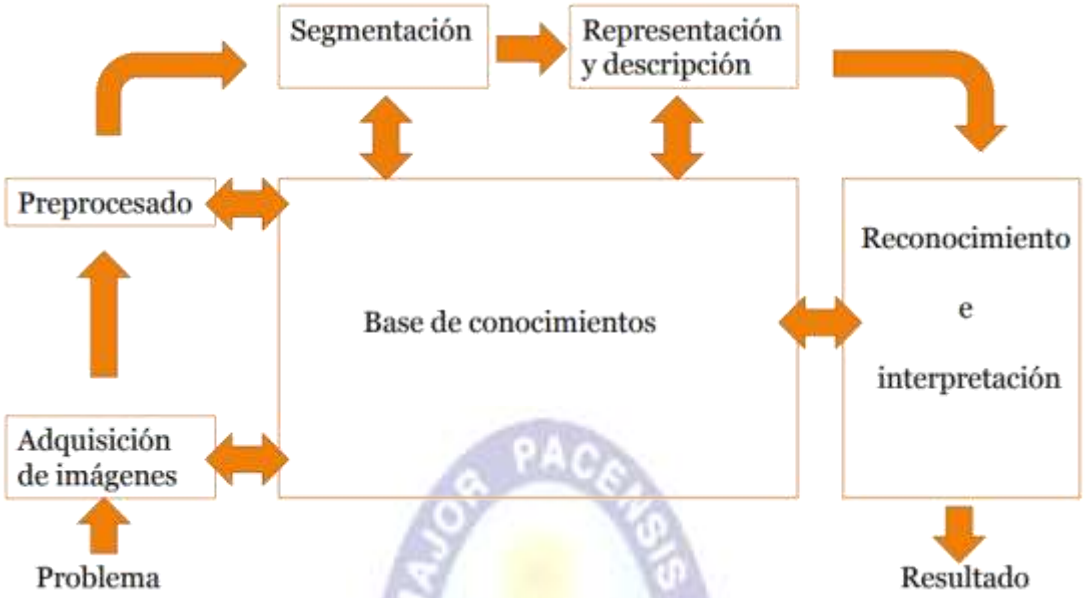


Figura 2.17: Etapas fundamentales del Procesamiento de imágenes digitales.
Fuente: (Laboratorio I2.31, 2018).

2.5.1. PREPROCESADO

En este apartado se tratan las operaciones y transformaciones que se aplican sobre las imágenes digitales en una etapa de procesamiento previa a las de segmentación y al reconocimiento. Su objeto es mejorar o destacar algún elemento de las imágenes, de manera que las etapas posteriores sean posibles o se simplifiquen (Vélez, et al., 2002).

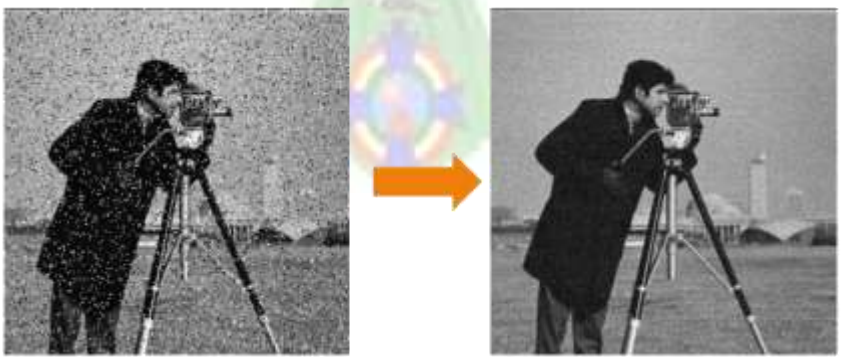


Figura 2.18: Etapa de Preprocesado.
Fuente: (Laboratorio I2.31, 2018).

2.5.2. SEGMENTACIÓN

Segmentar una imagen digital significa dividirla en zonas disjuntas e individualizadas. Es decir, consiste en diferenciar los diversos objetos y dónde se encuentran del fondo, que puede ser más o menos complejo, de la imagen (EDMANS, 2006).

En caso que los objetos y el fondo tengan niveles de gris diferentes se procede directamente a la detección de umbrales con el fin de extraer cada uno de los objetos de la imagen. Esta técnica clasifica cada pixel como perteneciente al objeto o al fondo según corresponda su nivel de gris. Sin embargo, en muchas ocasiones no resulta posible diferenciar de una forma tan sencilla los distintos elementos presentes en la imagen. En estas circunstancias se recurre al estudio de gradientes de luminosidad (Vélez, et al., 2002).

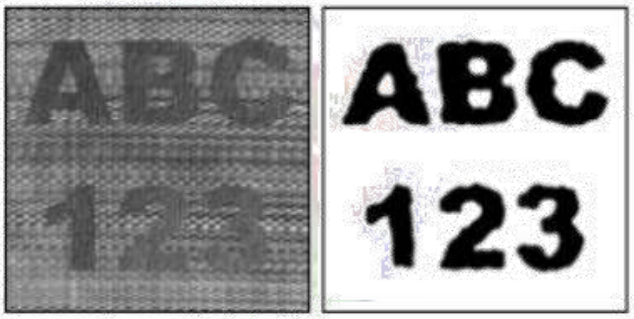


Figura 2.19: Etapa de Segmentación.
Fuente: (Laboratorio I2.31, 2018).

2.5.3. REPRESENTACIÓN Y DESCRIPCIÓN

La elección de una representación es sólo una parte de la transformación de los datos de entrada. Es necesario especificar un método que extraiga los datos de interés. La parametrización, que recibe también el nombre de selección de rasgos, se dedica a extraer rasgos que producen alguna información cuantitativa de interés o rasgos que son básicos para diferenciar una clase de objetos de otra (EDMANS, 2006).

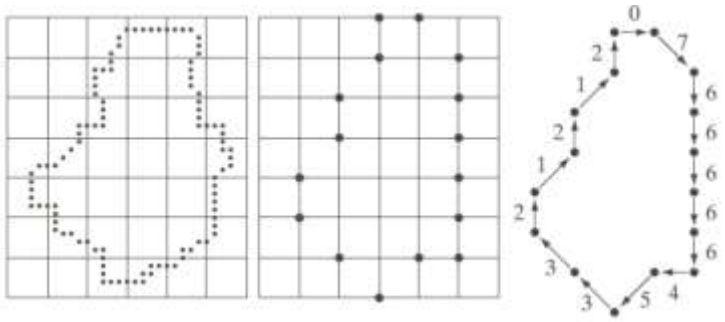


Figura 2.20: Etapa de representación y descripción.
Fuente: (Laboratorio I2.31, 2018).

2.5.4. RECONOCIMIENTO E INTERPRETACIÓN

El reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los descriptores (clasificación). La interpretación lleva a asignar significado al conjunto de objetos reconocidos (EDMANS, 2006).

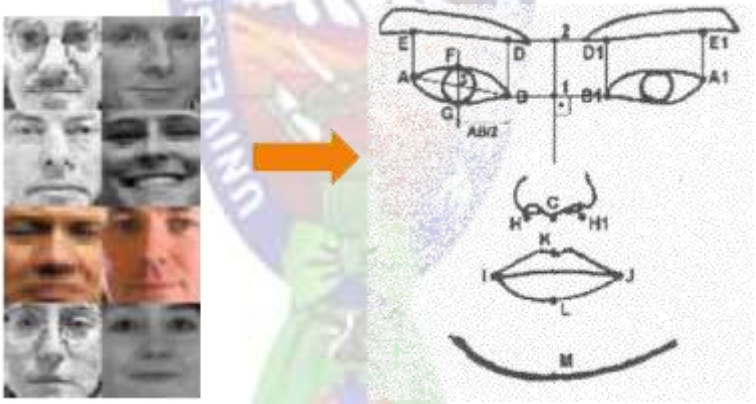


Figura 2.21: Etapa de reconocimiento e interpretación.
Fuente: (Laboratorio I2.31, 2018).

2.6. ARDUINO

Arduino fue creado durante 2005 por el entonces estudiante Massimo Banzi del instituto IVRAE en Italia, para integrar y facilitar el aprendizaje integrado de computación y electrónica para estudiantes de dicho instituto, con el principio de Hardware abierto (del inglés Open-Hardware). Desde entonces, dado su fácil acceso, ya que inicialmente era gratis y luego de bajo

costo, y por su gran éxito en el instituto IVRAE en Italia, se inició su distribución y comercialización tanto en Italia como en el resto del mundo. Gracias al éxito de Arduino, ya existen dispositivos hardware y aplicaciones software compatibles con Arduino (Margolis y Jones, 2017).

La figura 2.22 muestra y detalla los componentes de Arduino Uno R3. Dadas las entradas y salidas de una placa Arduino, para una interacción directa con ella, diferentes componentes y dispositivos de electrónica con el nivel de voltaje y corriente de Arduino (Margolis, 2011).

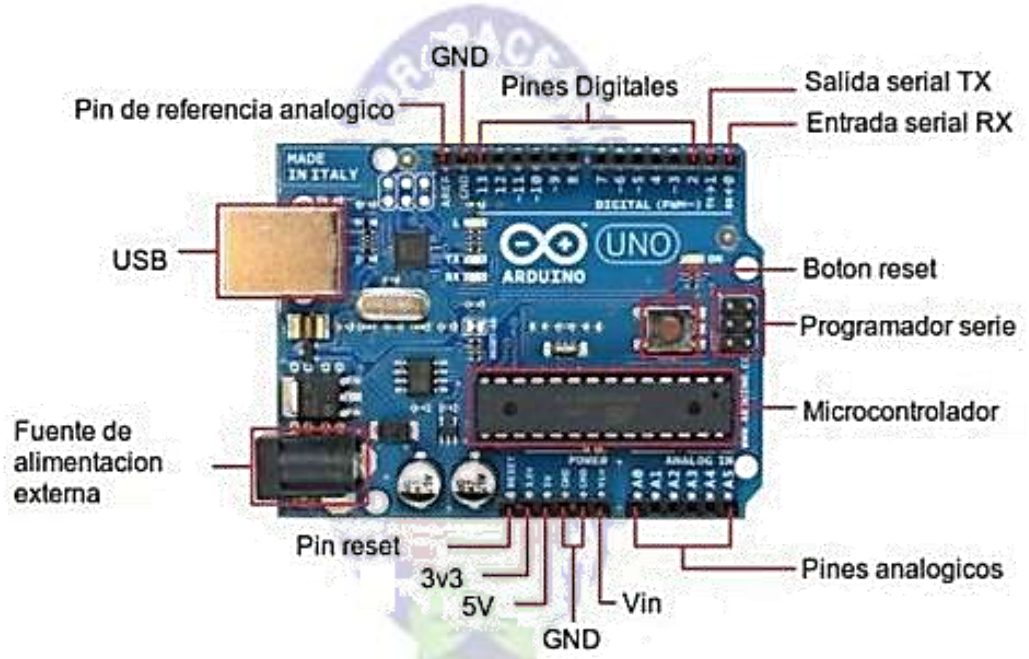


Figura 2.22: Placa Arduino Uno R3.
Fuente: (Adaptada de Lineros et al., 2018).

Por la accesibilidad y bajo costo de las placas Arduino y su accesible entorno de programación, es posible hoy en día la enseñanza de sistemas embebidos en cursos iniciales de programación y sistemas digitales (Brock et al., 2009).

2.6.1. DETECTORES EN ARDUINO

El uso más habitual del detector está asociado a ciertas máquinas que pueden captar algún fenómeno, por lo general peligroso. De este modo, cuando el detector descubre aquello que le

ha sido indicado que detecte en su programación, emite algún tipo de alerta o señal, dando aviso de la detección (Porto, 2012).

Existen muchos tipos de detectores, entre ellos, están:

- Detector de mentiras: Inventado en 1938 por Leonarde Keeler de la Policía de Berkeley, California, Estados Unidos, es conocido también bajo el nombre de polígrafo o máquina de la verdad. Se mide a través de respuestas fisiológicas como la presión arterial, ritmo cardíaco, frecuencia respiratoria, estímulos nerviosos y conductancia de la piel si la persona examinada está diciendo la verdad o está mintiendo frente a lo que se le pregunta.
- Detector de billetes falsos: Es el que se encarga de diferenciar los billetes legítimos de los que no lo son. Existen diversas tecnologías para esto, entre las que se hallan: infrarrojo, magnética y ultravioleta.
- Detector de temperatura, calor o térmico que es un instrumento de alarma de incendio. Está diseñado para activarse cuando el calor por convección, se produce un incendio aumenta la temperatura de un elemento que posee dicho instrumento y que es sensible al calor.
- Detector de detección de intrusos o de intrusiones que es un programa que detecta el acceso no autorizado a una computadora o una red.
- Detector de movimiento que es un dispositivo electrónico equipado por sensores que registran el movimiento físico. Normalmente se utilizan en sistemas de seguridad y en circuitos cerrados de televisión.

2.6.2. SENSORES EN ARDUINO

Los sensores que se usarán serán los siguientes:

2.6.2.1. SENSOR BUZZER

Un buzzer pasivo o un altavoz son dispositivos que permiten convertir una señal eléctrica en una onda de sonido. Estos dispositivos no disponen de electrónica interna, por lo que tenemos que proporcionar una señal eléctrica para conseguir el sonido deseado.

En oposición, los buzzer activos disponen de un oscilador interno, por lo que únicamente tenemos que alimentar el dispositivo para que se produzca el sonido. El buzzer activo es el que tiene la pegatina y al alimentarlo entre 5V y GND suena a una frecuencia fija.

Figura 2.23: Placa Arduino Uno R3.
Fuente: (Jecrespom, 2019).

El buzzer pasivo no tiene un oscilador interno y por lo tanto la frecuencia del sonido debemos hacerla desde Arduino.

2.7. PYTHON Y ARDUINO

Para la interfaz de Arduino con Python tenemos el método más fácil, utilizando el protocolo Firmata en la placa Arduino. El protocolo se implementa utilizando el firmware Standard Firmata, mientras que en Python usamos las bibliotecas Firmata, PyFirmata o PyMata. Otro método de interfaz Python-Arduino incluye el uso de una serie simple pero no estándar porque son comandos que utilizan el boceto personalizado de Arduino y la biblioteca pySerial en Python programa.

También es posible utilizar una red informática para establecer la comunicación entre Python y Arduino.

Del mismo modo conectamos varios componentes de hardware, es decir, sensores, resistencias con Arduino mediante pines digitales. Utilizando métodos simples de Python para establecer una conexión entre su placa Arduino y el programa Python.

2.7.1. COMUNICACIÓN SERIAL Y PYSERIAL

El uso del lenguaje de programación Python es muy popular por lo que es muy probable que ya esté disponible en cualquier sistema operativo; incluido las comunicaciones serie, trabajando en el campo de la electrónica y telecomunicaciones, ya que son bastante usuales, por supuesto, el mundo de Arduino no resulta una excepción y resulta muy sencillo conectar Arduino con Python, empleando el puerto serie y la librería PySerial.

Los puertos serie son la forma principal de comunicar una placa Arduino con un ordenador. Gracias al puerto serie podemos, por ejemplo, mover el ratón o simular la escritura de un usuario en el teclado, enviar correos con alertas, controlar un robot realizando los cálculos en el ordenador, encender o apagar un dispositivo desde una página Web a través de Internet, o desde una aplicación móvil a través de Bluetooth. Existen muchas posibilidades en las que se requiere el empleo del puerto serie. Por tanto, el puerto serie es un componente de alta relevancia para una gran cantidad de proyectos de Arduino, y es uno de los elementos básicos que debemos aprender para poder sacar todo lo dable de Arduino (Llamas, 2014).

Un puerto es el nombre genérico con que denominamos a los interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos. Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). No obstante,

pueden existir otros conductores para referencia de tensión, sincronismo de reloj, entre otros. (Llamas, 2014).

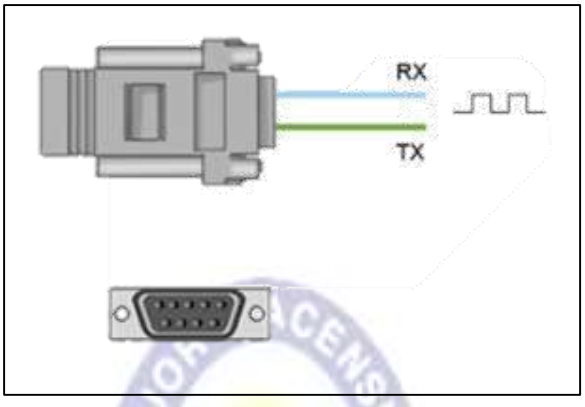


Figura 2.24: Comunicación serie.
Fuente: (Llamas, 2018).

Por el contrario, un puerto paralelo envía la información mediante múltiples canales de forma simultánea. Para ello necesita un número superior de conductores de comunicación, que varían en función del tipo de puerto. Igualmente existe la posibilidad de conductores adicionales además de los de comunicación (Llamas, 2018).

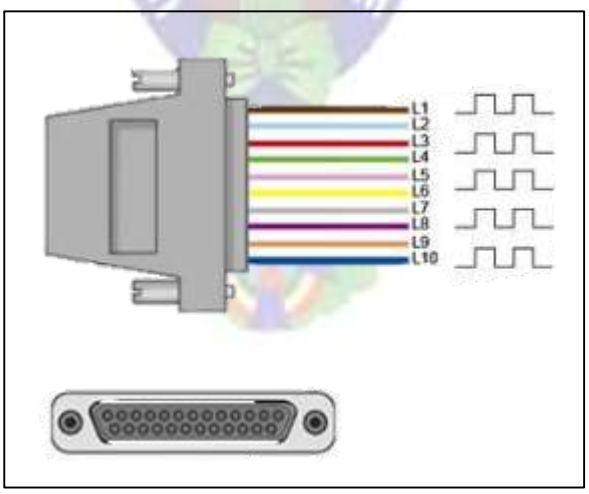


Figura 2.25: Comunicación paralelo.
Fuente: (Llamas, 2018).

Los puertos serie están físicamente unidos a diferentes pines de la placa Arduino. Evidentemente, mientras usamos los puertos de serie no podemos usar como entradas o salidas digitales los pines incorporados con el puerto serie en uso.

2.8. METODOLOGÍA

2.8.1. METODOLOGÍA DE INVESTIGACIÓN

El presente trabajo se basará en el tipo de investigación exploratorio-experimental utilizando el método de investigación inductiva. La investigación exploratorio- experimental se emplea para identificar un problema poco conocido, donde se ejerce un máximo control de las variables ya que su metodología es generalmente cualitativa. De ahí derivan los hallazgos y conclusiones, los cuales se fundamentan en la hipótesis sin obtener indispensablemente una muestra representativa.

2.8.2. METODOLOGÍA DE INGENIERÍA

2.8.2.1. MÉTODO

Para el modelo en V dice que las pruebas necesitan empezarse lo más pronto posible en el ciclo de vida. También muestra que las pruebas no son sólo una actividad basada en la ejecución. Hay una variedad de actividades que se han de realizar antes del fin de la fase de codificación. Estas actividades deberían ser llevadas a cabo en paralelo con las actividades de desarrollo, y los técnicos de pruebas necesitan trabajar con los desarrolladores y analistas de negocio de tal forma que puedan realizar estas actividades y tareas y producir una serie de entregables de pruebas. Los productos de trabajo generados por los desarrolladores y analistas de negocio durante el desarrollo son las bases de las pruebas en uno o más niveles (Soria, 2009).

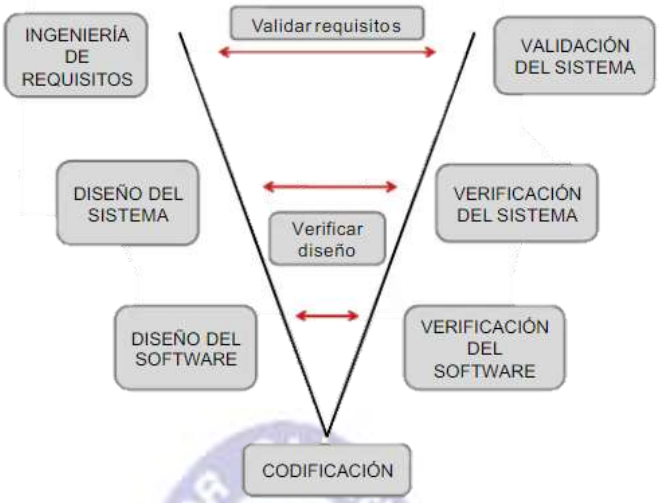


Figura 2.26: Modelo de ciclo de vida en V.
Fuente: (Inteco, 2009).

El modelo en V es un modelo que ilustra cómo las actividades de prueba (verificación y validación) se pueden integrar en cada fase del ciclo de vida. Dentro del modelo en V, las pruebas de validación tienen lugar especialmente durante las etapas tempranas, por ejemplo, revisando los requisitos de usuario y después, por ejemplo, durante las pruebas de aceptación de usuario (Inteco, 2009).

El modelo en V define las siguientes etapas de desarrollo (Perez et al.,2006).

- **Definición de especificaciones (Fase 1):** Se deben definir y documentar los diferentes tipos de detector a desarrollar, identificando los valores numéricos más concretos posibles. Entre ellos debe estar la especificación del nivel de integridad, en caso de ser requerido.
- **Diseño global (Fase 2):** También llamado diseño de alto nivel. Su objetivo es obtener un diseño y visión general del sistema.
- **Diseño en detalle (Fase 3):** Consiste en detallar cada bloque de la fase anterior.
- **Implementación (Fase 4):** Es la fase en la que se materializa el diseño en detalle.

- **Test unitario (Fase 5):** En esta fase se verifica cada módulo Hardware y Software de forma unitaria, comprobando su funcionamiento adecuado.
- **Integración (Fase 6):** En esta fase se integran los distintos módulos. Como en el caso anterior, ha de generarse un documento de pruebas. Por una parte, se debe comprobar en todo el funcionamiento correcto del detector, y por otra, en caso de tratarse con una tolerancia a fallos, debe verificarse que ante la presencia de un fallo persiste el funcionamiento correcto. Se comprueba el cumplimiento de los requisitos establecidos.
- **Test operacional del sistema (Fase 7):** Se realizan las últimas pruebas, pero sobre un escenario real, en su ubicación final, anotando una vez más las pruebas realizadas y los resultados obtenidos.

Las ventajas y desventajas de este modelo (Inteco, 2009).

Ventajas

- Es un modelo simple y fácil de utilizar.
- En cada una de las fases hay entregables específicos.
- Tiene una alta oportunidad de éxito sobre el modelo en cascada debido al desarrollo de planes de prueba en etapas tempranas del ciclo de vida.
- Es un modelo que suele funcionar bien para proyectos pequeños donde los requisitos son entendidos fácilmente.

Desventajas

- Es un modelo muy rígido, como el modelo en cascada.
- Tiene poca flexibilidad y ajustar el alcance es difícil y caro.
- El software se desarrolla durante la fase de implementación.

- El modelo no proporciona caminos claros para problemas encontrados durante las fases de pruebas.

2.8.2.2. TÉCNICAS

Las técnicas usadas serán de acuerdo al método mencionado, para organizar el ciclo de vida del proyecto, la cual está compuesta por: Definición de especificaciones, diseño global, diseño en detalle, codificación, test unitario, integración, test operacional del sistema.

2.8.2.3. HERRAMIENTAS

Las herramientas son las siguientes:

Fritzing es un programa libre de automatización de diseño electrónico para documentar el detector basado en Arduino y crear el esquema del circuito impreso para su posterior fabricación (Fritzing, 2020).

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Usaremos el Arduino para la codificación para el sensor buzzer.

Scikit-learn es una biblioteca de software de aprendizaje automático para el lenguaje de programación Python. Será de utilidad para poner a prueba las bibliotecas.

Python es un lenguaje de programación multiparadigma soportando la orientación a objetos, programación imperativa y programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es un lenguaje que está ganando usuarios que se dedican al campo de la inteligencia artificial (Oliva, 2018). El framework Keras utilizado

para el entrenamiento de redes neuronales es una API integrada en Python, siendo esta la razón de la elección para la integración en el proyecto.

TensorFlow es una biblioteca de software de código abierto para el aprendizaje automático. Fue creado y lanzado por Google en el año 2015 consiguiendo un gran éxito hasta el momento. Cuenta con una gran cantidad de colaboradores que lo sitúan como líder en el sector del aprendizaje profundo (Oliva, 2018). Para el presente proyecto se hace uso indirecto de Tensor Flow debido a la integración de la biblioteca Keras.

Keras es una API de redes neuronales de alto nivel, escrita en Python y con posibilidad de integrarse con Tensor Flow, CNTK4 o Theano5. Su desarrollo se basó en crear una API con la que realizar un experimento sobre una red fuese una tarea sencilla y rápida. Los modelos se escriben en código Python. La implementación en Python ofrece la ventaja a la hora de la depuración y extensión de los modelos. Gracias al desarrollo de librerías como Keras, Python está aumentando en el sentido de desarrolladores que trabajan con él (Oliva, 2018).

OpenCV es una biblioteca libre de visión por computador desarrollada por Intel. Desde su aparición en 1999 ha sido utilizado en infinidad de aplicaciones. Esto se debe a que su publicación se da bajo licencia Berkeley Software Distribution (BSD), que permite que sea usado libremente para fines comerciales y de investigación con las condiciones expresadas. Otras funciones como redimensionamiento o recorte de imágenes también están incluidas en la biblioteca. Estas características junto la detección de objetos han decantado la elección para la integración en el proyecto (Oliva, 2018).

YOLOv3 es el sistema de detección de objetos mediante Darknet, como una herramienta que posee un modelo ya pre entrenado que permite detectar variedad de objetos

en imágenes y videos en tiempo real. Este modelo se puede modificar para crear un modelo personalizado, es ahí donde inician los aportes de este documento, procurando ser una guía práctica de cómo empezar a generar productos que se integren en el campo del Machine Learning, como el reconocimiento de objetos en imágenes (Miranda, 2019).

Adobe Photoshop es un editor de fotografías desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos. Utilizaremos esta herramienta para cambiar el tamaño de las imágenes en el entrenamiento.

Cascade Trainer GUI es un programa que ofrece una interfaz gráfica, la cual facilita el uso de herramientas de OpenCV para el entrenamiento y prueba de clasificadores. Será utilizado tanto para el entrenamiento y la validación de las imágenes mediante un modelo generado por el mismo programa.

CAPÍTULO III

MARCO APLICATIVO

CAPÍTULO III: MARCO APLICATIVO

3.1. INTRODUCCIÓN

En este punto se da a conocer el uso de la metodología de acuerdo al detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales. La metodología que se utiliza es el Modelo en V, la cual se basa en la validación y verificación, donde se realiza un sistema mediante componentes que pueden agruparse en diferentes módulos, hasta llegar a representar por completo el resultado de la detección y prevención del uso de barbijo y distanciamiento físico mediante las redes neuronales convolucionales y el Arduino.

Para el desarrollo del dispositivo lo separamos por etapas, que sirven para acercarnos hacia el propósito final del sistema para prevenir los casos de emergencia sanitaria la cual está descrita en la metodología del modelo en V, como etapas que son parte de un producto final. Las etapas corresponden a siete pasos a desarrollar: la implementación de los sensores, la implementación y entrenamiento de la red neuronal, la alarma en base a Arduino y la unión de los todos los anteriores mencionados; y formar el producto final con su respectiva validación y verificación.

3.2. DESARROLLO DE LA METODOLOGÍA EN V

3.2.1. FASE 1: DEFINICIÓN DE ESPECIFICACIONES

Antes de definir las especificaciones detallaremos el pre estudio que se realiza para el desarrollo de los proyectos relacionados a Arduino y redes neuronales convolucionales, donde se describe un análisis previo al desarrollo del proyecto con los principales puntos en esta fase inicial.

- La necesidad que se presenta es prevenir el distanciamiento físico y recomendar el uso de barbijo.
- Actualmente se cuenta con dispositivos de grabación de la alcaldía, los cuales se pueden utilizar para obtener datos de multitudes y falta de uso de barbijo, una alternativa más barata y casera es la utilización de placas Arduino para emitir una alarma de prevención.
- El detector de restricciones para la prevención para casos de contagio en la emergencia basado en redes neuronales y Arduino sirve de alerta al usuario para notificar las aglomeraciones y recomendar el uso de barbijo.
- Se utiliza el entorno de desarrollo integrado multiplataforma de código abierto para la codificación de las redes neuronales en base a la prevención de uso de barbijo y el distanciamiento físico.
- En este proyecto se utiliza la plataforma Arduino en la que nos apoyaremos con otros dispositivos como sensores para poder construir el detector.

Aclarados estos puntos continuamos con la descripción de la definición de especificaciones. La prevención de contagio en la emergencia sanitaria, se basa en un control de detección de personas que no cumplen el distanciamiento físico y el uso de barbijo, además de que se elabora una alarma para prevenir dichos casos. Para ello se utilizan las redes neuronales convolucionales, las cuales reciben imágenes de una cámara, la prevención es mediante un Arduino Uno y finalmente se envían alertas.

Figura 3.1: Diagrama general del sistema.

El detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, es manejado a través de las redes neuronales que está encargado de recibir imágenes y mandar una solicitud de alarma a la tarjeta Arduino, la cual debe activar o desactivar la alarma para el distanciamiento físico y el uso de barbijo.

3.2.2. FASE 2: DISEÑO GLOBAL

También llamado diseño de alto nivel, su objetivo es obtener un diseño y visión general del sistema. Para el detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, tenemos los siguientes puntos.

Redes Neuronales Convolucionales.

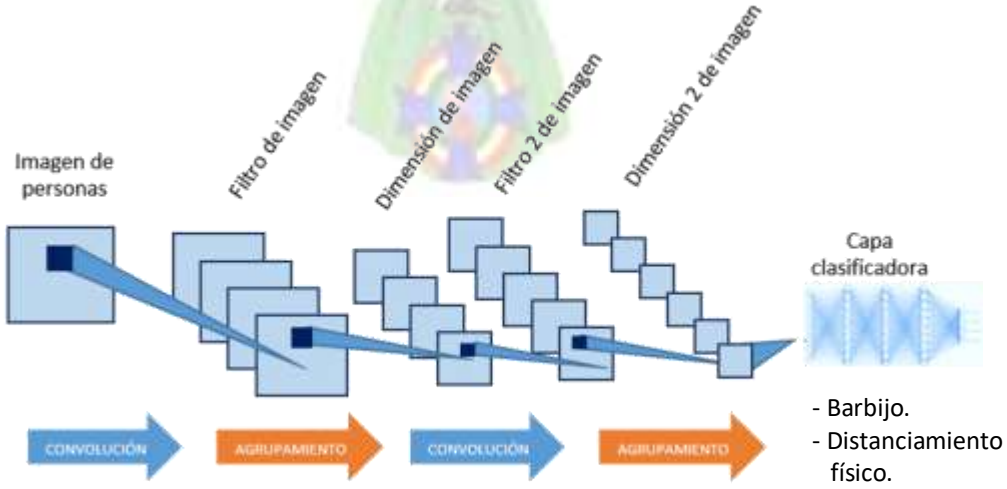


Figura 3.2: R.N.C. Detección de barbijo y distanciamiento físico.

Circuito del esquema para la alarma en Arduino.

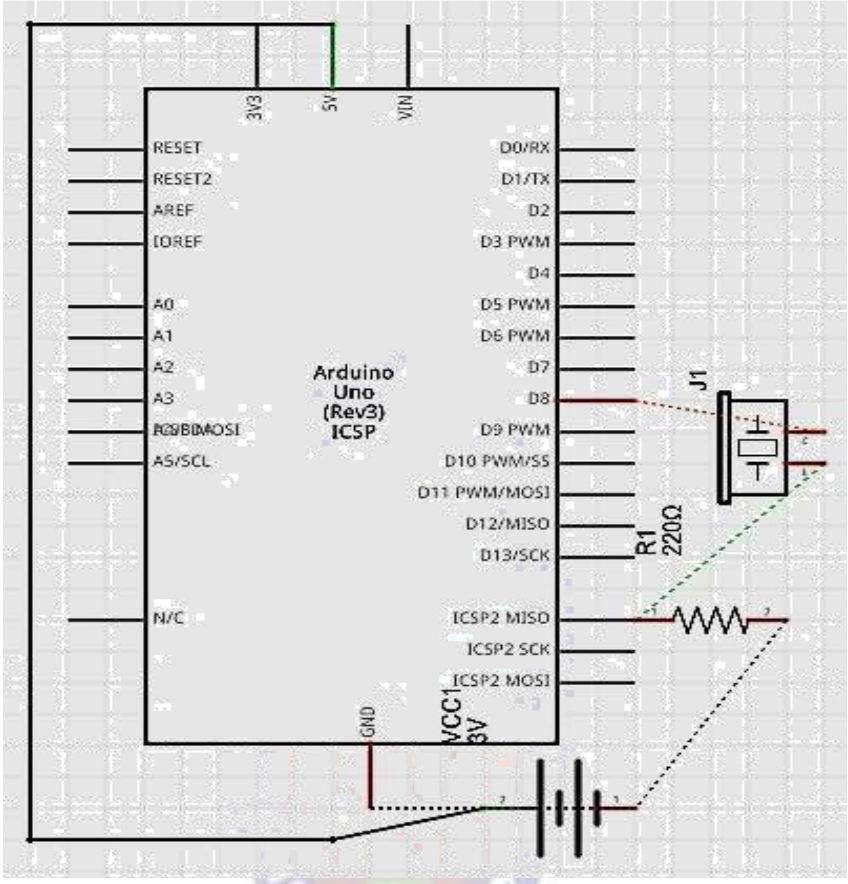


Figura 3.3: Alarma en Arduino.

3.2.3. FASE 3: DISEÑO EN DETALLE

A continuación, se muestra el esquema hardware y la relación con las redes neuronales donde se aprecia la forma de conexión de los diferentes componentes del sistema y posteriormente el desarrollo del programa en Anaconda-Spyder.

Redes neuronales convolucionales.

En este caso tenemos la codificación de los algoritmos para el uso de barbijo, mediante el siguiente procedimiento:

- Leer la imagen o video.
- Aplicar la red neuronal Convolucional para detectar todos los rostros en la imagen.

- Aplicar la red neuronal Convolucional para determinar si el rostro tiene o no mascarilla.
- Insertar recuadro clasificador en rostro de imagen original.
- Repetir el proceso para todos los rostros en la imagen.

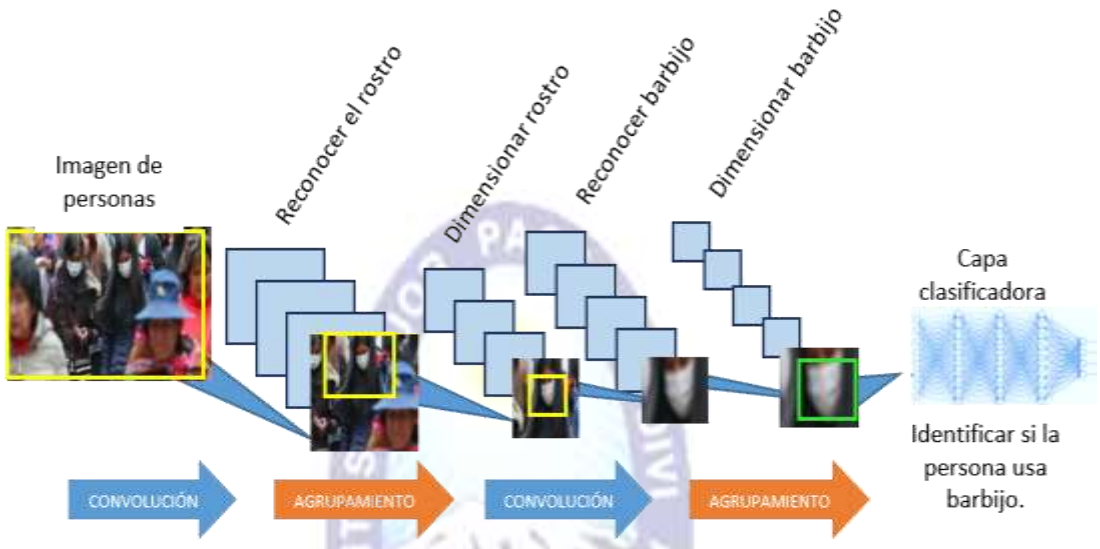


Figura 3.4: Detección de rostros.

Para el distanciamiento físico se realiza el siguiente proceso mediante la detección de una imagen o video de personas que cumplen o no con el distanciamiento físico:

- Leer la imagen del video o la imagen.
- Aplicar el sistema de detección de objetos YOLOv3 para detectar los cuadros delimitadores de las personas.
- Obtener los puntos de interés a partir de los cuadros delimitadores generados.
- Mapear los puntos de interés utilizando la matriz de transformación de perspectiva pre-calculada. Se considera estos puntos puestos que son los que nos brindan la referencia de la ubicación de las personas desde una perspectiva de arriba hacia abajo.
- Calcular todas las distancias entre los puntos transformados para determinar en base a un límite definido las personas que cumplen una distancia mínima requerida o no.

- Mostrar los resultados, se colorean de rojo los que no cumplen y en verde los que sí cumplen, además de mostrar la cantidad de violaciones al distanciamiento físico.
- Repetir el proceso para todas las imágenes recibidas.

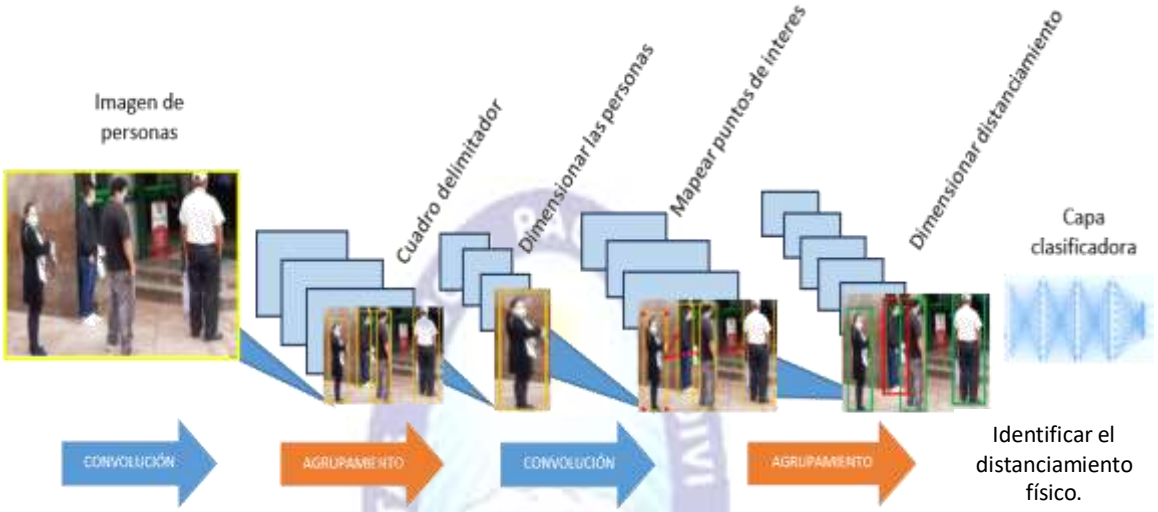


Figura 3.5: Detección de distanciamiento físico.

Conexión alarma

Para la instalación de la alarma se sabe que el Buzzer activo es un dispositivo que emite un sonido cuando lo conectamos a GND y 5V. Esto quiere decir, que, si en lugar de conectarlo a 5V lo conectamos a cualquier pin digital de nuestro Arduino, podemos controlarlo cuando emite el sonido. La diferencia entre un Buzzer pasivo y activo es que al primero tenemos que fijar la onda para que emita sonidos, mientras que el buzzer activo ya tiene en su circuito interno esa onda, por lo que cuando el pin esté HIGH emite el sonido.

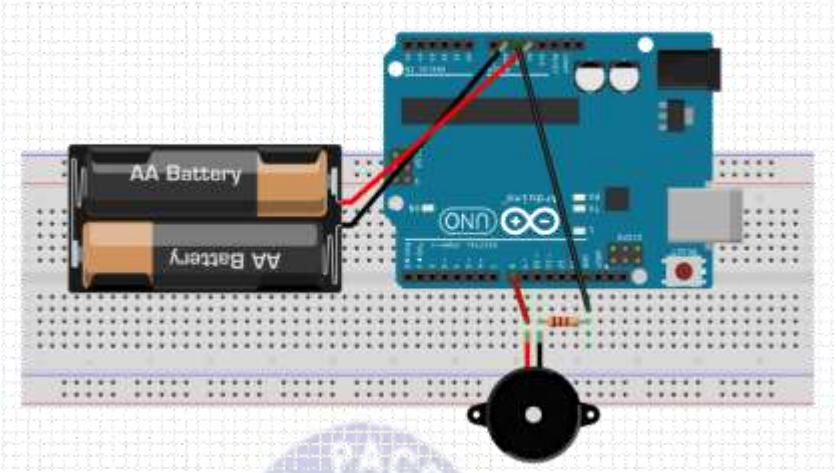


Figura 3.6: Diseño en detalle de la alarma en Arduino.

En la figura anterior notamos la conexión de los sensores magnéticos que van conectados para emitir una alarma en caso de que exista multitudes o falta de uso de barbijo.

3.2.4. FASE 4: CODIFICACIÓN

En esta fase describiremos el desarrollo de los módulos del programa. El código de programación del detector para la prevención de casos de contagio mediante el uso de Arduino y redes neuronales, está constituido de la siguiente manera:

- Inclusión de librerías tanto para la comunicación serial y el uso del Arduino.
- Creación del puerto serial.
- Declaración de las variables.
- Definición de las entradas para realizar el monitoreo de los dispositivos electrónicos y el control de acceso.
- Definición de las salidas para activar y desactivar el detector en base a Arduino.
- Inicialización del puerto serie para establecer la comunicación entre las redes neuronales convolucionales y el Arduino UNO.
- Creación de las funciones para realizar la alarma y la detección de imágenes o video.

- Sentencias para realizar el detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado Arduino y redes neuronales, los parámetros requeridos para activar y desactivar la alarma invocando a la función respectiva, al igual que a la función que realiza el envío de alertas mediante Python.
- Instrucciones que detecten las restricciones en la emergencia sanitaria.
- Instrucciones para realizar el detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, con la utilización de redes neuronales convolucionales e invocación de las funciones para prevenir y alertar a la persona.
- Instrucciones para que el Arduino y Spyder puedan realizar una acción.
- Finalización del programa, la repetición del ciclo mediante el void loop y el entrenamiento del modelo de las redes neuronales convolucionales.

CÓDIGO DE PROGRAMACIÓN UTILIZADO PARA ARDUINO

A continuación, se muestra el código que se ejecuta para que el Arduino con los cuales se realiza un detector para prevenir la falta de uso de barbijo y el distanciamiento físico. La figura describe el uso de librerías, la iniciación de variables y declaración de salidas digitales. Se procede a la inicialización de las variables y parámetros necesarios para la implementación de la alarma.

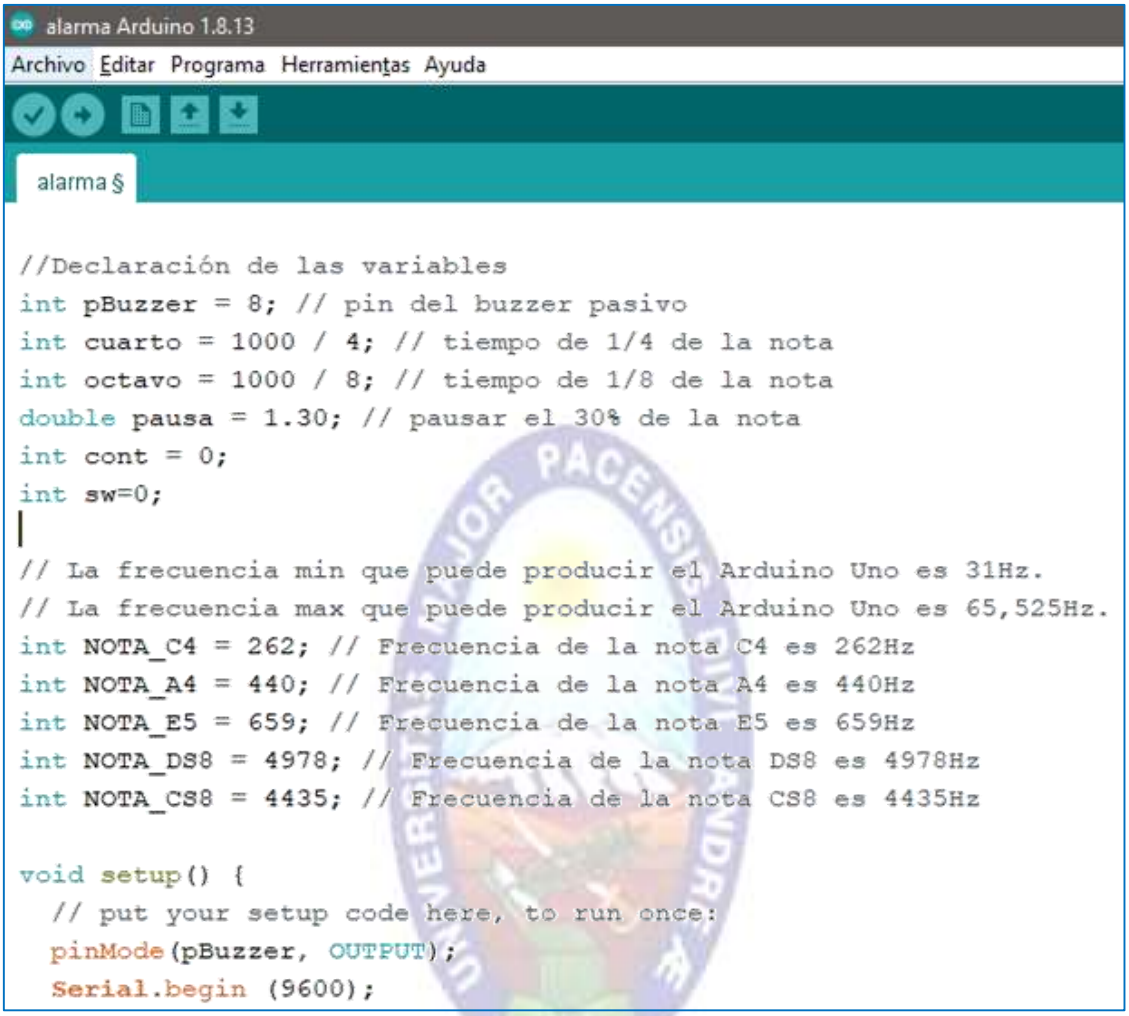


Figura 3.7: Inicialización y declaración de variables.

El detector en Arduino se implementa con dos sonidos diferentes, tanto para la detección del uso de barbijos y el distanciamiento físico, por lo cual se realiza la combinación de notas musicales, tomando en cuenta el tiempo y frecuencia de reproducción para obtener diferentes sonidos.

Del mismo modo el puerto serial fue implementado con un contador para la obtención de datos mediante la terminal.

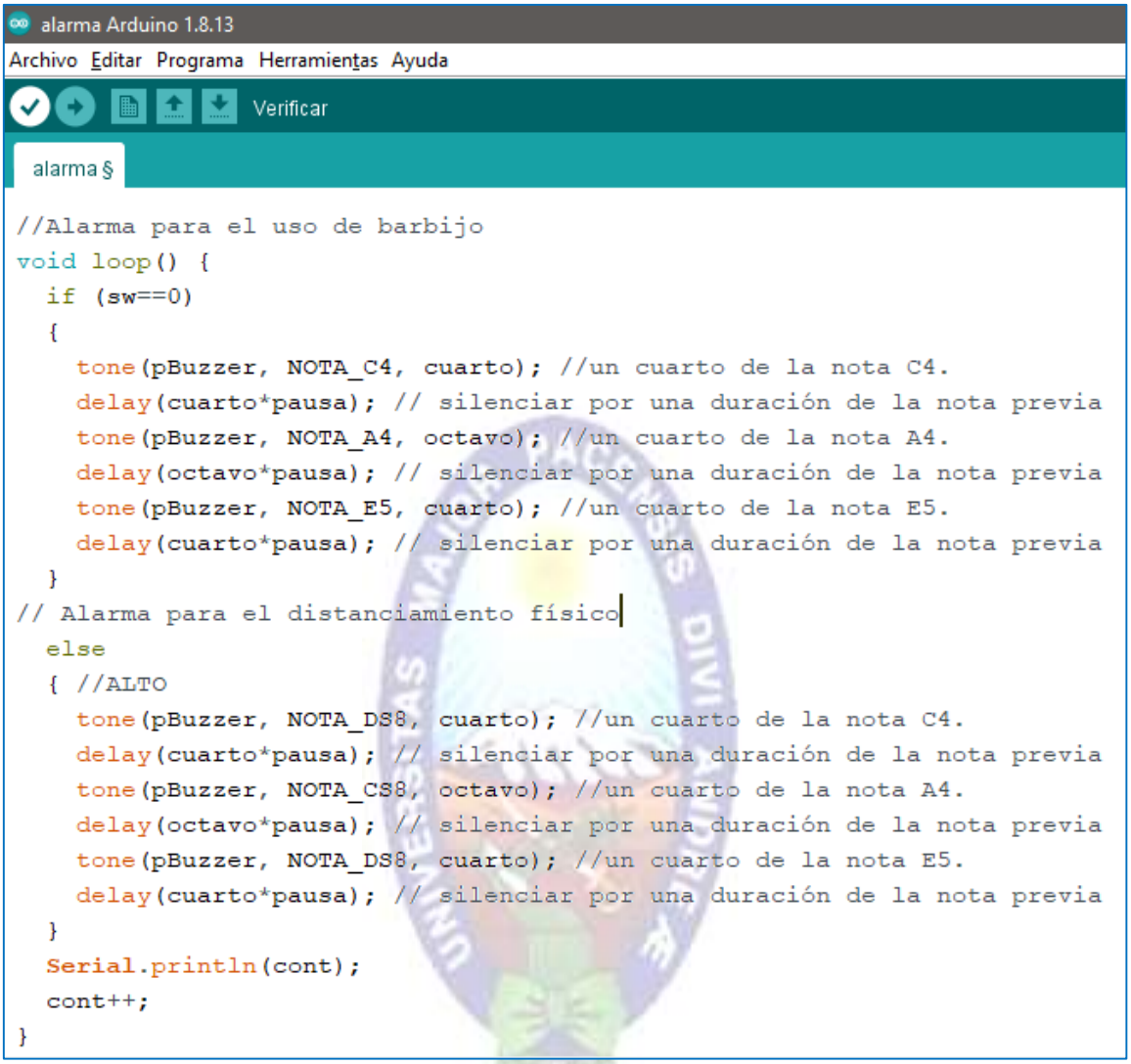


Figura 3.8: Sensor Buzzer para el detector.

CÓDIGO DE PROGRAMACIÓN UTILIZADO PARA REDES NEURONALES CONVOLUCIONALES

Para la codificación y ejecución se realiza mediante Spyder-Anaconda, el cual tiene cargados los archivos necesarios y la validación del entorno de ejecución.

Red neuronal Convolucional para la detección de uso de barbijo, se realiza mediante los pasos: Convolución, capas de agrupamiento, unidad rectificada, conexión de capas, el entrenamiento, predicción y el procesamiento de imágenes.

Detección de barbijo.

- Se efectúa el entrenamiento con el siguiente código.

Primero se realiza la implementación de las librerías y paquetes necesarios para el código.

```
import sys
import os
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras import optimizers
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dropout, Flatten, Dense, Activation
from tensorflow.python.keras.layers import  Convolution2D, MaxPooling2D
from tensorflow.python.keras import backend as K
#Iniciamos sesión en Keras
K.clear_session()
```

Figura 3.9: Librerías.

En la figura 3.9. tenemos los siguientes componentes:

- Librerías sys, os: Para tener acceso a los documentos del computador.
- Se utiliza TensorFlow, la cual es una librería de código abierto, en nuestro caso se maneja como clasificador de imágenes y determina la probabilidad. En cuanto a esta librería también encontramos, las siguientes características:
  - ImageDataGenerador: Se utiliza para el preprocesamiento de imágenes.
  - Optimizers: Es un comando para entrenar el algoritmo de convolución.
  - Sequential: Permite hacer redes neuronales secuenciales, donde cada capa está en orden para cada convolución.
  - Dropout: Es utilizado para apagar las redes neuronales y generar nuevos caminos de procesamiento.
  - Flatten: Generamos la imagen plana.
  - Dense: Agregamos en la función la cantidad de neuronas y la función de activación.

- Activation: Utilizado para la función de activación ReLU.
- Convolution2D y MaxPooling2D: Son las capas de convolución y agrupamiento
- Backend: Para reiniciar los datos en el núcleo de Anaconda-Spyder.

```
#Imágenes de entrenamiento de persona c/s barbijo
data_entrenamiento = './data/entrenamiento'
data_validacion = './data/validacion'
```

Figura 3.10: Conjunto de imágenes.

En la figura 3.10, se presentan como datos de entrada la inicialización y se considera el directorio de las dos carpetas de imágenes de las personas que usan o no usan barbijo, en otro caso utilizan mal el barbijo, tanto para el entrenamiento y validación. Por el momento, las imágenes utilizadas provienen de la cámara del computador o de capturas ya tomadas.

```
#Parámetros
epocas=20
longitud, altura = 100, 100
batch_size = 32
pasos = 1000
pasos_validacion = 200
filtrosConv1 = 32
filtrosConv2 = 64
tamano_filtro1 = (3, 3)
tamano_filtro2 = (2, 2)
tamano_pool = (2, 2) #tamaño de filtro maxpooling
clases = 2
lr = 0.0005 #ajustes de la red neuronal
```

Figura 3.11: Parámetros de detección.

En la figura 3.11. Como primer paso se tiene la cantidad de iteraciones, que son veinte. En la convolución mediante los filtros se observa la detección del barbijo en distintas posiciones y escalas. En este paso tenemos el procesamiento de imágenes, el cual pasa por el preprocesado de la imagen con una altura y longitud de 100 pixeles, batch_size es el número de imágenes para el procesamiento, pasos es el número de veces que se procesa en las iteraciones, pasos de validación para el aprendizaje del algoritmo, filtros de convolución

ponemos la profundidad de imagen que el igual a 32 y 64, el tamaño de filtro el cual es (3,3) para la primera iteración. Se cuenta con dos clases, las personas que utilizan barbijo y las que no lo usan, finalmente el ajuste de la red neuronal, usualmente es un número pequeño.

```
#Preparamos nuestras imágenes para el entrenamiento
entrenamiento_datagen = ImageDataGenerator(
 # toma las images de distintas posiciones
    rescale=1. / 255,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True)

validacion_datagen = ImageDataGenerator(
    rescale=1. / 255)
```

Figura 3.12: Convolución de la imagen.

Sobre la figura 3.12. Las capas de agrupamiento se realizan mediante el entrenamiento y validación con los parámetros ya establecidos. Se reescala la imagen con los pixeles, luego generamos las imágenes inclinadas, toma las imágenes más cercanas o lejanas que distinguen la direccionalidad de la red neuronal.

```
imagen_entrenamiento = entrenamiento_datagen.flow_from_directory(
    data_entrenamiento,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')

imagen_validacion = validacion_datagen.flow_from_directory(
    data_validacion,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')
```

Figura 3.13: Capa de agrupamiento.

En el siguiente paso se observa la figura 3.13, el paso de Capas de Agrupamiento y la Unidad lineal Rectificada, donde se realiza la reducción de la imagen mediante el Max Pooling y utilizamos la función de activación ReLU. Se define la altura y longitud de imágenes,

ingresa al directorio de data y procesa las imágenes, se procesa en un batch_size de 32, las etiquetas son del tipo categorical.

```
#Red neuronal convolucional
cnn = Sequential()
cnn.add(Convolution2D(filtrosConv1, tamano_filtro1, padding ="same",
                      input_shape=(altura, longitud, 3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Convolution2D(filtrosConv2, tamano_filtro2, padding ="same",
                      activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Flatten())  # Una dimension que tiene toda la informacion de RNC
cnn.add(Dense(256, activation='relu'))
cnn.add(Dropout(0.5))  # Aprende caminos alternos para aprender los datos
cnn.add(Dense(clases, activation='softmax'))
```

Figura 3.14: Capas de Agrupamiento y la Unidad lineal Rectificada.

La figura 3.14. Se tiene la conexión de capas y la generación del modelo entrenado. Donde formamos varias capas apiladas, después de una capa de convolución obtendremos la capa de agrupamiento, añadimos la siguiente capa Convolucional con el tamaño de filtro. Luego generamos la imagen plana, con 256 neuronas y activamos la función, además desactivamos las neuronas para aprender caminos alternos y crear modelos para información nueva.

```
#Conexión de capas
cnn.fit_generator(
    imagen_entrenamiento,
    steps_per_epoch=pasos,
    epochs=epocas,
    validation_data=imagen_validacion,
    validation_steps=pasos_validacion)

#Generación del modelo entrenado
target_dir = './modelo/'
if not os.path.exists(target_dir):
  os.mkdir(target_dir)
cnn.save('./modelo/modelo.h5')
cnn.save_weights('./modelo/pesos.h5')
```

Figura 3.15: Conexión de capas y generación del modelo.

Finalizamos con las imágenes de entrenamiento en la figura 3.15, el número de imágenes e iteraciones y las imágenes de validación. Guardamos el modelo entrenado en un archivo.

- Se realiza el código para la predicción, con el siguiente código.

```
import numpy as np
from tensorflow.python.keras.preprocessing.image import load_img, img_to_array
from tensorflow.python.keras.models import load_model
```

Figura 3.16: Librerías para la predicción.

Figura 3.16. Se efectúa la implementación de las librerías y paquetes necesarios para el código. Tenemos la carga de imágenes con la altura y ancho.

```
#Parámetros de la red neuronal convolucional
longitud, altura = 100, 100
modelo = './modelo/modelo.h5'
pesos_modelo = './modelo/pesos.h5'
cnn = load_model(modelo)
cnn.load_weights(pesos_modelo)
```

Figura 3.17: Parámetros para la predicción.

Figura 3.17. Se procede a la inicialización de los parámetros y el modelo ya procesado en el entrenamiento. La longitud y la altura que se definió a 100, se carga el modelo.h5 y cargamos la red neuronal ya procesada.

```
def predict(file): # Recibe la imagen de personas c/s barbijo
  x = load_img(file, target_size=(longitud, altura))
  x = img_to_array(x) # arreglo de valores
  x = np.expand_dims(x, axis=0)
  arreglo = cnn.predict(x)
  resultado = arreglo[0]
  respuesta = np.argmax(resultado) # indice del valor mas alto del resultado
  if  respuesta== 0:
    print("Predicción: Usa barbijo")
  elif respuesta == 1:
    print("Predicción: No usa barbijo")

#Predecimos la imagen c/s barbijo
predict('sb1.jpg')
```

Figura 3.18: Implementación de la predicción.

Figura 3.18 La predicción se realiza mediante el modelo adquirido en el entrenamiento. Creamos una función llamada predict, cargamos la imagen para la predicción, convertimos la imagen con un arreglo con el img_to_array, luego añadimos una dimensión extra. Después realizamos la predicción con la imagen como arreglo. La respuesta es el índice del valor más alto y obtenemos el resultado.

Figura 3.19: Imagen de prueba.

Figura 3.19. Se utiliza la imagen de prueba y de esta manera se determina si la persona usa o no usa el barbijo.

Detección del distanciamiento físico

Para la detección del distanciamiento físico, se cumple mediante la herramienta YOLOv3, el cual es un sistema que nos permite la detección de objetos con una variedad de imágenes o videos a tiempo real. También se utiliza las herramientas open-cv y numpy.

```
 8 # Directorio de Yolo
 9 MODEL_PATH = "yolo-coco"
10
11 #Iniciamos la minima probabilidad para la detección
12 #para el filtrado de imágenes
13 MIN_CONF = 0.3
14 NMS_THRESH = 0.3
15
16 #Se define la distancia mínima entre dos personas en pixeles
17 MIN_DISTANCE = 50
```

Figura 3.20: Parámetros para la detección.

Figura 3.20. Como primer paso se inicializa el directorio de YOLOv3, donde se tiene las variables para el filtrado de imágenes y la probable distancia mínima que deberían cumplir las personas para la detección.

```
# Librerias y paquetes
from Yolo3 import NMS_THRESH
from Yolo3 import MIN_CONF
import numpy as np
import cv2
```

Figura 3.21: Librerías.

Figura 3.21. En cuanto al archivo de detección de personas, se importan las librerías necesarias para determinar el distanciamiento físico mediante el filtrado de imágenes las cuales se encuentran en el archivo anteriormente creado. Además, se tiene las librerías numpy y cv2 para el uso de la visión computacional.

```
#Dimensión de la imagen en una lista
def deteccion_persona(frame, net, ln, personIdx=0):

    (H, W) = frame.shape[:2]
    results = []
    #Creamos una salida(blob) y el detector Yolo avanza en las imágenes
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
    net.setInput(blob)
    layerOutputs = net.forward(ln)

    #Se inicializa los cuadros delimitadores y los centros de cada imagen
    boxes = []
    centroids = []
    confidences = []
```

Figura 3.22: Dimensión de imágenes y cuadros delimitadores.

Figura 3.22. Como siguiente paso para la detección de personas, a las salidas obtenidas mediante el procesamiento mediante YOLOv3, el cual dimensiona las imágenes y se generan los cuadros delimitadores de las personas. Para YOLOv3 se tiene una salida mediante el

comando blob, el cual permite que el detector avance imagen por imagen, llegando así delimitar las imágenes en base a los puntos centros.

```
# Importamos las librerias necesarias
import Yolo3 as config
from Deteccion_Persona import deteccion_persona
from scipy.spatial import distance as dist
import numpy as np
import argparse
import imutils
import cv2
import os
```

Figura 3.23: Librerías.

Figura 3.23. Para la detección del distanciamiento físico entre personas se implementan las librerías o paquetes necesarios. Donde tenemos las librerías:

- Yolo3: Archivo del filtrado de imágenes para determinar la distancia física mínima.
- Deteccion_Persona: Archivo creado para la dimensión de imágenes y cuadros delimitadores.
- distance: Se importa de spicy.spatial que se refiere al dato de un espacio geométrico.
- numpy: Librería para crear vectores y matrices grandes.
- argparse: Es utilizado para el análisis de argumentos de los archivos.
- imutils: Librería de conjuntos de funciones para el procesamiento de imágenes.
- cv2: Es la librería para la visión por computador.
- Librería os: Para tener acceso a los documentos del computador.

```python
#Detección de personas y calcular los mapas de distanciamiento
if len(results) >= 2:
    #Se obtiene la distancia Euclidiana de las personas
    centroids = np.array([r[2] for r in results])
    D = dist.cdist(centroids, centroids, metric="euclidean")

    #Matriz de distancias
    for i in range(0, D.shape[0]):
        for j in range(i + 1, D.shape[1]):
            #Distancias de dos personas es menor que el #pixeles
            if D[i, j] < config.MIN_DISTANCE:
                #Personas que violan el distanciamiento
                violate.add(i)
                violate.add(j)
```

Figura 3.24: Cálculo y detección del distanciamiento físico.

Figura 3.24. Se procede al cálculo de la distancia entre personas mediante los mapas de distanciamiento, donde se obtiene la distancia Euclidiana y se guardan los valores obtenidos en una matriz, entonces se observa si se cumple el distanciamiento físico o se comete una violación. Para este proceso se tiene la captura de imágenes con la cámara del computador o ya sea imágenes guardadas.

3.2.5. FASE 5: TEST UNITARIO

DETECCIÓN USO DE BARBIJO

En este punto tenemos el Arduino conectado al Buzzer y emite una alerta, donde se puede realizar la diferencia de sonido para la detección de barbijo y el distanciamiento físico.

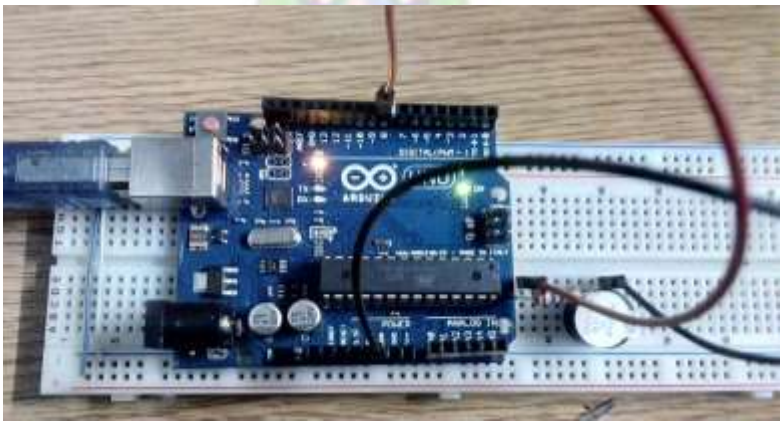


Figura 3.25: Implementación del sensor Buzzer.

Figura 3.25. Se obtiene la implementación del Arduino y el Buzzer para la detección correspondiente.

DETECCIÓN USO DE BARBIJO

Las redes neuronales convolucionales que muestran el entrenamiento que se realiza con personas que usan y no usan barbijo.

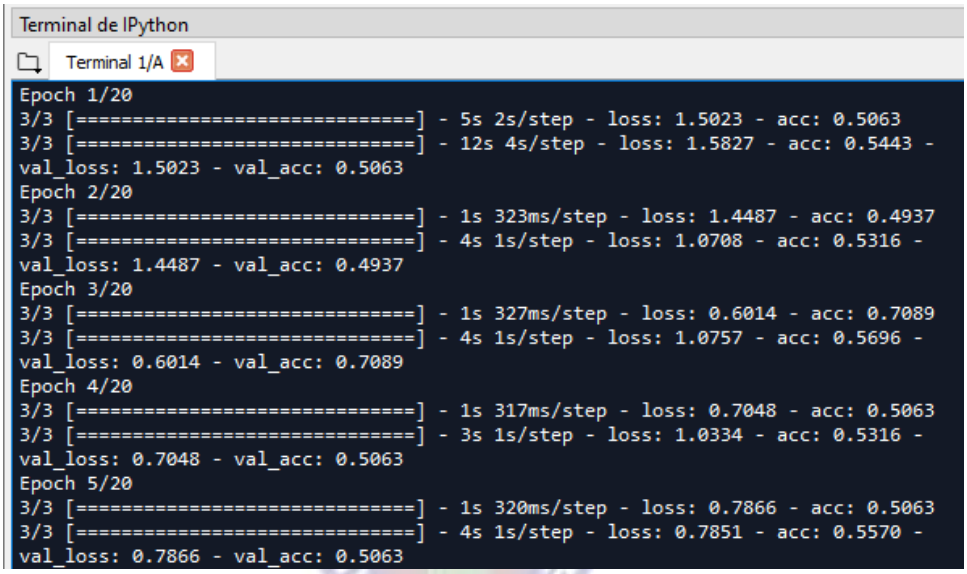


Figura 3.26: Entrenamiento de la red neuronal convolucional.

En la figura 3.26, en la estructura implementada el número de cada capa arroja un valor, en la elección de cada imagen para la distinción de los rostros y el barbijo.

Como primera etapa recibimos las imágenes y se guardan en una carpeta de capturas, donde definimos si se hace el uso de barbijo o no, en el caso que se use el barbijo de manera inadecuada, la imagen corresponde a que la persona está sin barbijo.

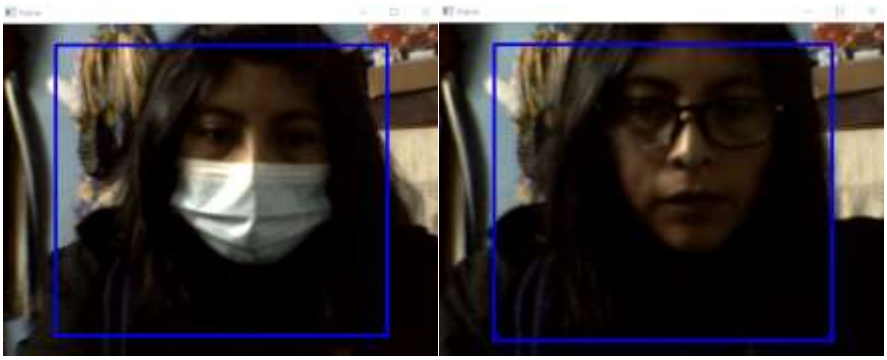


Figura 3.27: Recepción de imágenes.

Figura 3.27. El formato de las imágenes recibidas es en .jpg y .png, en cuanto a las dimensiones puede variar, pero es recomendable que sean de un solo tamaño. Para el entrenamiento y validación utilizamos imágenes de altura:300 y ancho:262. En relación al video, se recomienda la extensión .mp4, pero con la calidad media o baja, así de esta manera se realice un mejor procesamiento de imágenes. Las capturas tomadas por el computador pueden ser de distintos ángulos tanto del rostro como del barbijo.

Para el entrenamiento de forma gráfica se utiliza el programa Cascade Trainer GUI, donde se realizará el entrenamiento de las personas con o sin barbijo, y así obtener un modelo entrenado.

Figura 3.28: Imágenes de entrenamiento.

En la figura 3.28, realizamos el entrenamiento con Cascade Trainer GUI.

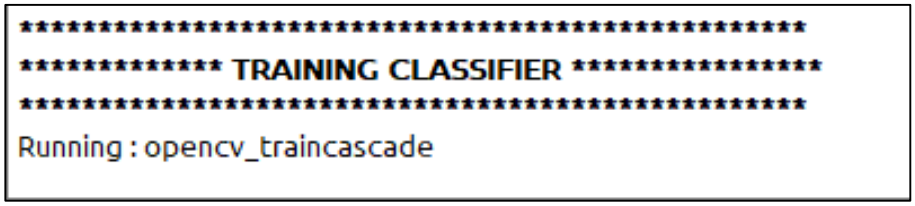


Figura 3.29: Entrenamiento en Cascade Trainer GUI.

La predicción de la red neuronal convolucional se realiza con la imagen de una persona que no usa barbijo, determinando lo siguiente:

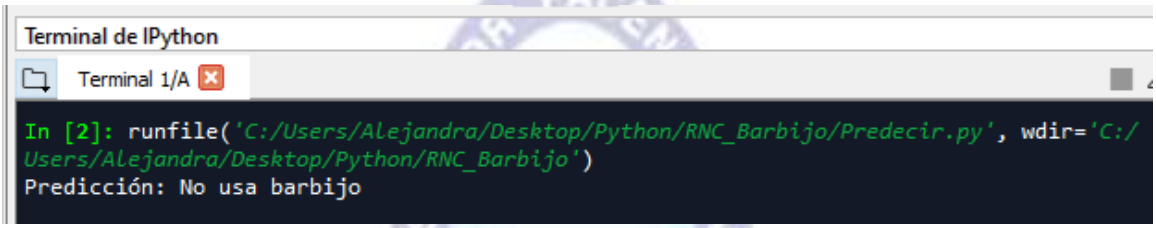


Figura 3.30: Predicción de la red neuronal convolucional para el barbijo.

Por tanto, para la detección de barbijo se tiene el siguiente modelo de genérico de funcionamiento, tanto el entrenamiento y la predicción de imágenes que se obtienen en las capturas.

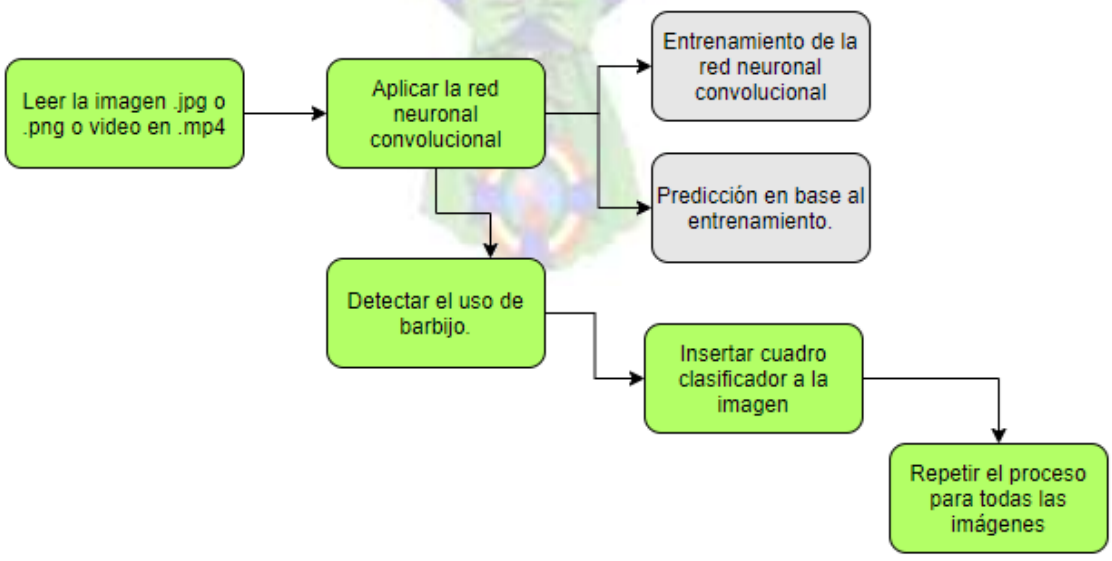


Figura 3.31: Modelo genérico de detección de barbijo.

DETECCIÓN DE DISTANCIAMIENTO FÍSICO

Las redes neuronales convolucionales que muestran de forma gráfica como se detecta la imagen o video para el distanciamiento físico.

- El sistema desarrollado en el presente proyecto integrador evalúa cada persona y su distanciamiento, a partir de las funciones y el modelo de Yolov3.
- Se maneja imágenes para su entrenamiento y en base a ello se obtuvieron los resultados en imágenes.

Figura 3.32: Procesamiento de la detección de distanciamiento físico.

Por tanto, para la detección de distanciamiento físico se tiene el siguiente modelo de genérico de funcionamiento, mediante el uso de YOLOv3.

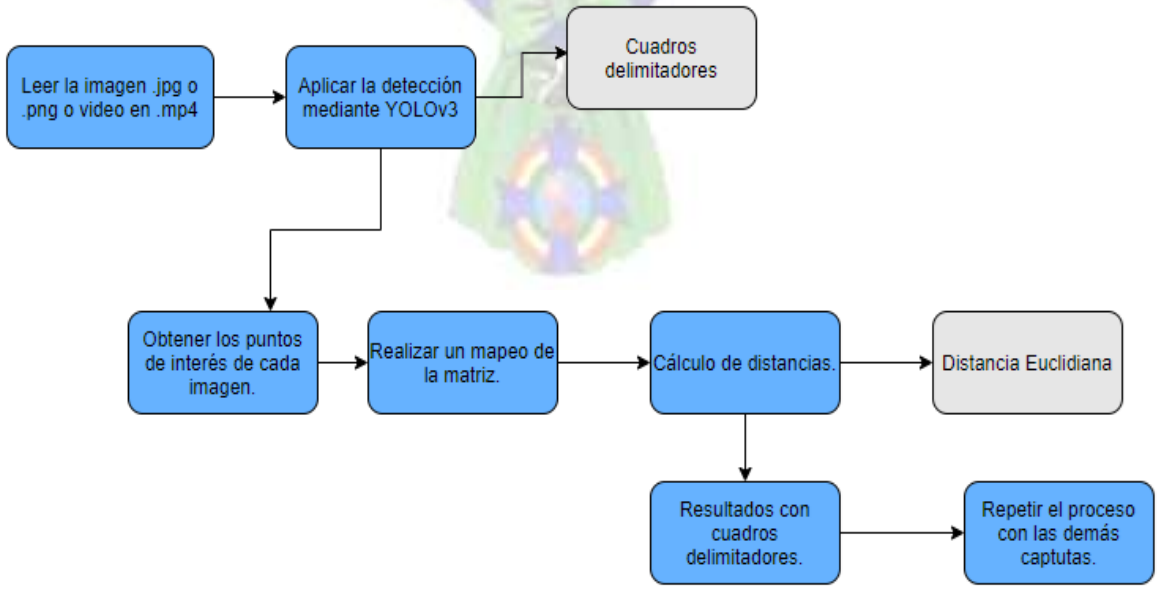


Figura 3.33: Modelo genérico la detección de distanciamiento físico.

3.2.6. FASE 6: INTEGRACIÓN

La integración se realiza mediante el Arduino y el Spyder, los cuales se unen mediante el puerto PySerial, esto es de gran ayuda para emitir la alarma de acuerdo a los resultados de las redes neuronales convolucionales y YOLOv3. Ha de generarse un documento de pruebas. Por una parte, se debe comprobar el funcionamiento correcto del detector; en caso de tratarse con un sistema tolerante a fallos, debe verificarse que ante la presencia de un fallo persiste el funcionamiento correcto.

Documento de Prueba 1		
Modulo	**Actividad**	**Observaciones**
Sensor Buzzer	Alarma en Arduino.	El sensor solo tiene frecuencia de sonido hasta un metro de distancia.
		Solución: Obtener un Buzzer más grande.
Conexión de capas.	Detección de Barbijo.	Se genera el modelo, pero no cuenta con una parte gráfica para observar la detección.
		Solución: Implementar un modelo en Cascade Trainer GUI para la detección de barbijo y así obtener los resultados de forma gráfica.
Resultados por consola.	Detección del distanciamiento físico.	Los resultados sobre detección de distanciamiento se procesan, pero no se adquieren mediante consola.
		Solución: Mostrar mediante la consola la cantidad de veces que se viola el distanciamiento físico.
Resultados por consola.	PySerial	Los resultados varían dependiendo del proceso que se ejecuta, ya sea la detección de barbijo o el distanciamiento físico.

Tabla 3.1: Documento de prueba 1.

3.2.7. FASE 7: TEST OPERACIONAL DEL SISTEMA

En esta última fase se realizan las últimas pruebas del detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, como producto final, pero sobre un escenario real se realizan las pruebas necesarias y se dan soluciones a las fallas en general para su uso y funcionamiento.

En la siguiente tabla 3.2 muestra la documentación del funcionamiento del detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales. Además de dar soluciones y correcciones a las fallas del detector en general para su uso y funcionamiento adecuado en base a la calidad de software.

Documento de Prueba 2		
Modulo	**Actividad.**	**Observaciones**
Sensor Buzzer	Alarma en Arduino.	El sensor no se escucha a más de 2 metros de distancia en la calle.
		Solución: Obtener un parlante o realizar una conexión con otro dispositivo.
Detección de imágenes con la cámara.	Detección de Barbijo.	Se genera el modelo, pero solo con imágenes de la cámara y no del video, ya que el proceso es más lento.
		Solución: Implementar con YOLOv3 la detección de barbijo.
Cascade Trainer	Altura y tamaño de imágenes.	Solo se debe generar imágenes pequeñas mediante la cámara ya que genera un gran retardo en el procesamiento y clasificación.
		Solución: Utilizar otros programas para el entrenamiento de imágenes.
Resultados por consola.	Detección del distanciamiento físico.	Se obtienen los resultados mediante la plataforma de Arduino, aunque la alarmar es continua.

Tabla 3.2: Documento de prueba 2.

CAPÍTULO IV

PRUEBA DE HIPÓTESIS

CAPÍTULO IV: PRUEBA DE HIPÓTESIS

4.1. INTRODUCCIÓN

La hipótesis es una proposición que nos permite establecer relaciones entre los hechos Tamayo (1989). En el presente capítulo se realiza la evaluación de la hipótesis planteada en la investigación, la cual indica:

Hi: Mediante el detector de aglomeraciones y uso de barbijos, elaborado en base al procesamiento de imágenes, Arduino y redes neuronales, se previene la concurrencia de personas y la falta de uso de barbijos por medio de alertas en la emergencia sanitaria con una precisión del 90%.

El nivel de confianza o significancia que se elige para 10% es:

$$1 - \propto = 0.1 \quad \text{Ecuación 4.1.}$$

$$\propto = 1 - 0.1$$

$$\alpha = 0.9$$

4.2. PRUEBA DE CONTRASTE

El contraste de hipótesis es una estrategia metodológica fundada en la inferencia estadística que permite evaluar si una propiedad que el investigador supone existe en una población, es compatible con lo observado en una muestra experimental. El contraste de hipótesis se caracteriza por constituir uno de los instrumentos más apropiados para la toma de decisiones en condiciones de incertidumbre (López Casuso, 1996).

Un contraste de hipótesis o Test de hipótesis estadístico es una prueba de significación o una prueba estadística, que indican el proceso mediante el cual decidimos si una propuesta de tesis respecto de la población, debe ser aceptada o no. Esta propuesta es lo que se denomina hipótesis estadística.

El contraste de la hipótesis estadística se basa en la información proporcionada por la muestra. De esta manera, si rechazamos la hipótesis, queremos indicar que los datos de la muestra ofrecen cierta evidencia sobre su falsedad. Si la aceptamos simplemente queremos significar que no se rechaza. Un contraste de hipótesis consiste, por tanto, en estudiar dos hipótesis:

Ho: Hipótesis nula.

Hi: Hipótesis alternativa.

De manera que el investigador divide los resultados muestrales en dos zonas; una zona de rechazo y otra de aceptación, según como obtengamos el resultado, aceptaremos o rechazaremos la hipótesis.

- Zona de rechazo: Es el área bajo la curva para el estadístico que corresponda, en la distribución Normal, t-Student entre otros.
- Zona de aceptación: Es el área bajo la curva que es complementaria a la zona de rechazo, ambas se generan a partir del nivel de significancia de la prueba.
- Al aplicar un contraste de hipótesis, obtenemos los siguientes datos asociados a la tesis.

Alcance de la hipótesis

- **Población**

  En la presente tesis se realizará el trabajo con los ciudadanos de La Paz y El Alto.

- **Muestra**

  La tesis tiene una muestra aleatoria la cual cuenta con 15 personas, las cuales pertenecen al área de salud, policía, alcaldía y vecinos de distintas zonas de La Paz y El Alto.

- **Técnica**

  Para ello, se utiliza el cuestionario de la encuesta, la cual debe contener una serie de preguntas o ítems respecto a una o más variables a medir.

- **Instrumento**

  Como instrumento se utiliza el cuestionario para obtener información y para el contraste de hipótesis se utiliza la distribución muestral de una proporción con t-Student.

4.3. ANÁLISIS DE LA PRUEBA DE HIPÓTESIS

En la prueba de hipótesis se utiliza la distribución muestral de una proporción con la t-Student. Consideramos una población en la que la proporción de elementos poblacionales portadores de una cierta característica, donde la población es infinita.

Debido a que nuestra muestra es de 15 personas. Cumple con el siguiente estadígrafo estadístico con la distribución t-Student.

Donde:

$\bar{p}: Media\ muestral$

$p: Proporción\ muestral$

$n: Observaciones$

$$Z = \frac{\bar{p} - p}{\sqrt{\frac{p(1-p)}{n}}}$$ Ecuación 4.2.

Los datos obtenidos en base a la experimentación del detector de restricciones para la prevención para casos de contagio en la emergencia sanitaria mediante el uso de Arduino y redes neuronales, son los siguientes:

Muestra: 15 personas

	Casos					
Respuestas	Policía	Salud	Alcaldía	Otros	Total	%
Totalmente en desacuerdo					0	0
Desacuerdo					0	0
Indeciso		1		1	2	13.3 %
De acuerdo	1	1		4	6	40.0 %
Totalmente de acuerdo	1	2	2	2	7	46.7 %
Total	2	4	2	7	15	100 %
%	13.3 %	26.7 %	13.3 %	46. 7%	100 %	

Tabla 4.1: Tabla de frecuencias.

Como resultado podemos observar los siguientes datos:

- Dos personas encuestadas están indecisas respecto al sistema.
- Seis personas están de acuerdo respecto al sistema.
- Siete personas si están totalmente de acuerdo.

De esta manera como resultado tenemos que 13 personas encuestadas aceptan el detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales. Así reemplazando los datos obtenidos de acuerdo a lo propuesto en la tesis y así mismo calculando y reemplazando en la ecuación de prueba.

Donde:

$$\bar{p} = \frac{13}{15} = 0.87$$

$$p = 90\% = 0.90$$

$$n = 15$$

$$Z = \frac{\bar{p} - p}{\sqrt{\frac{p(1-p)}{n}}} = \frac{0.87 - 0.90}{\sqrt{\frac{0.90(1-0.90)}{15}}} = \frac{-0.03}{0.08} = -0.38$$

Según la hipótesis y el dato obtenido, tenemos:

Hipótesis nula: $H_0 : p = p_0$

Hipótesis alternativa: $H_1 : p < p_0$

Obtenemos con la tabla de t-Student el resultado de 3.733, el valor obtenido nos indica que rechazamos la hipótesis nula lo cual significa que aceptamos nuestra hipótesis, se observa que el valor obtenido está en la región de aceptación.

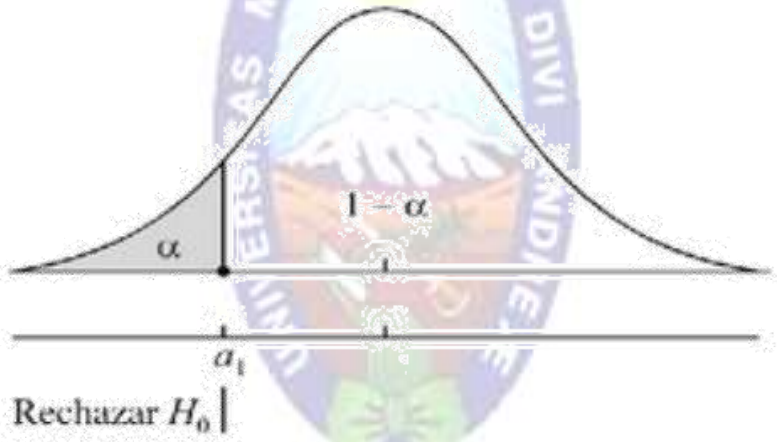


Figura 4.1: Prueba de hipótesis unilateral izquierda.

Mediante la implementación del detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, podemos comprobar que la propuesta de la tesis es aceptada.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

Después de haber realizado el análisis, diseño e implementación del presente trabajo, se llegan a las siguientes conclusiones:

- ✓ El entrenamiento de una red neuronal se llevó a cabo a la perfección. entrenandola en base a imágenes y videos los cuales fueron una parte fundamental para el entrenamiento de la red neuronal e implementación al software siendo la red neuronal primordial dentro de este trabajo.
- ✓ Se implementó un detector el cual es capaz de emitir son sonidos diferentes, tanto para la prevención del distanciamiento físico y el uso del barbijo, a través de un circuito que se fue detallando en esta tesis y unido al detector final, el cual proporciona una base sólida para la prevención de casos de contagio en la emergencia sanitaria mediante el uso de Arduino y redes neuronales.
- ✓ El sensor Buzzer fue probado y aplicado en base a los requerimientos del detector de la tesis, usando una red neuronal para detectar imágenes o video, capaz de arrojar datos, trabajando la alarma y la red neuronal como uno solo.
- ✓ Ya visto los puntos anteriores la alarma no tiene ningún error respecto a los datos recibidos por las redes neuronales, ya que cada uno va trabajando de una forma donde se enviará una señal y una emisión de sonido.
- ✓ Gracias al entrenamiento de la red neuronal en base al procesamiento de las imágenes, cumple detección automática de aglomeraciones y el uso inadecuado del barbijo para la prevención de mayores contagios en esta emergencia sanitaria.

- ✓ Llegamos a tener una hipótesis aceptable, como se muestra en el capítulo 4 aplicando fórmulas estadísticas, y es más contamos con el detector el cual es funcional, el cual cumplió con las expectativas deseadas.

5.2. RECOMENDACIONES

La realización de mejoras continuas en base a la tesis planteada, se recomienda a otros futuros estudiantes para ampliar el conocimiento de las redes neuronales y Arduino. Las cuales pueden acoplarse al detector a futuro y que son factores que sí sean un beneficio para la sociedad.

Proponer una mayor usabilidad del detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales y aplicar la praxis al diario vivir, la cual puede ser usada en la mayor parte del mundo y no siendo aplicado por el alto costo que tiene.

Posteriormente, las mejoras del detector de restricciones para la prevención de casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales, se recomienda seguir con las metodologías propuestas o mantenerlas ya que el detector puede aún ser adaptable a muchas otras áreas utilizando las redes neuronales y Arduino de distinta forma.

BIBLIOGRAFÍA

Alvarado, Raúl (2018). Ciudad inteligente y sostenible: hacia un modelo de innovación inclusiva. *Scielo.* Obtenido de http://www.scielo.org.mx/scielo.php?script=sci_ arttext&pid =S2007-36072018000100002

Alvear, Rosero, Peluffo, Pijal (2017). Internet de las Cosas y Visión Artificial, Funcionamiento y Aplicaciones: Revisión de Literatura. *Scielo.* Obtenido de http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-65422017000100244

Arduino. (2020). *Arduino*. Obtenido de https://www.arduino.cc/

Arduino. (2016). Sección: Reference. Consultado el 20 de agosto de 2020 de https://www.arduino.cc/en/Reference/HomePage

Arnal, D. (2020, marzo, 20). Proteger a tu paciente del coronavirus 2: uso adecuado mascarillas quirúrgicas. Obtenido de https://sensar.org/2020/proteger-a-tu-paciente-del-coronavirus-2-uso-adecuado-mascarillas-quirurgicas/

Atria Innovation (2019, octubre, 22). Las redes neuronales y sus funciones. Obtenido de https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

Betancur, Sonia (2020). Operacionalización de Variables. *Hacia promoc. Salud*, 29-36. Obtenido de http://fcaenlinea.unam.mx/anexos/1349/1349_u2_Act2.pdf

Braun. (Revisado 2020, septiembre, 14). Contribuye a frenar la propagación de la covid-19. Obtenido de https://www.bbraun.es/es/pacientes/prevencion-de-la-infeccion-para-pacientes/distanciamiento-fisico.html

Brun, Jesús (2014). Sistema de seguridad perimetral programable inteligente. Obtenido de https://riunet.upv.es/bitstream/handle/10251/39849/Memoria.pdf?sequence=1

Calvo, Diego (2017, 07, 13). Clasificación de las redes neuronales artificiales. Obtenido de https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/

Colah. (2014, 8 de julio). Conv Nets: A Modular Perspective. Obtenido de https://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Casas, Álvaro. (2017). *Reconocimiento de imágenes con Redes Convolucionales en C.* (Trabajo de fin de grado). Universidad de Sevilla.

Condori, Melissa (2016). *Sistema domótico de seguridad perimetral basado en arduino*. (Módulo GSM). Universidad Mayor de San Andrés. Bolivia.

Correa, Chichizola. (Consultado 22 de septiembre, 2020). *Diseño de Sistemas de reconocimiento de rostros.*

Decretos y Normas (Revisado 14 de Julio, 2020). Gaceta, Bolivia Segura. Obtenido de https://www.boliviasegura.gob.bo/normativa.php

Desai, Pratik. (2015). *Python Programming for Arduino* (1st Ed.). Packt Publishing Ltd: Livery Place

Doque, Santiago (2020, 03, 26). En Bolivia inició el estado de emergencia sanitaria por el COVID-19. Obtenido de https://www.aa.com.tr/es/mundo/en-bolivia-inici%C3%B3-el-estado-de-emergencia-sanitaria-por-el-covid-19-/1780656

EDMANS. (2006). *Técnicas y algoritmos básicos de Visión Artificial*. Universidad de La Rioja, España: Universidad de la Rioja

El cajón de Ardu. (2013, 17 de diciembre). Tutorial: conectando una pantalla LCD 1602A a Arduino UNO [Mensaje en blog]. Consultado el 20 de abril de 2016 de http://elcajondeardu.blogspot.mx/2013/12/tutorial-conectando-una-pantalla-lcd.html

Erroz, David (2019). *Visualizando neuronas en Redes Neuronales Convolucionales.* Universidad Pública de Navarra.

Fernández Güell, José. (2015). Ciudades Inteligentes: La mitificación de las nuevas tecnologías como respuesta a los retos de las ciudades contemporáneas. *Economía Industrial*, pp. 17-28. Obtenido de http://oa.upm.es/40941/

Flores, Medina, Mayorquín, Garcia (2017). Detección de Objetos a Color en Tiempo Real con Técnicas de Visión Artificial y Arduino. Revista de Prototipos Tecnológicos. *Vol. 3 No.7 1-6.*

Fritzing. (2020). *Fritzing.* Obtenido de https://fritzing.org/

Garcia, Francisco. (2018). *Ingeniería del Software: Modelos de Proceso*. Obtenido de https://repositorio.grial.eu/bitstream/grial/1142/1/IS_I%20Tema%203%20%20Modelos%20de%20Proceso.pdf

Garcia (2013). *Reconocimiento de imágenes utilizando redes neuronales artificiales.* Facultad de Informática, Universidad Complutense de Madrid.

Gausin, S. (2012, 19 de abril). Usando un buzzer con Arduino [Mensaje en blog]. Consultado el 20 de abril de 2016 de http://computointegrado.blogspot.mx/2012/04/usando-un-buzzer-con-arduino.html

Gonzales, Parra (2015). *Diseño e implementación de un sistema de reconocimiento de naranjas para el robot GIO 1 usando visión asistida por computador.* Universidad católica de Colombia, facultad de ingeniería, Bogota.

Gonzales y Woods. (2002). *Digital Image Processing* (2da ed.). Richmond, TX, Estados Unidos de America.

Hernández, Roberto (2014). *Metodología de investigación* (6ª ed.). México: Santa Fe.

Instituto nacional de tecnologías de comunicación, (2009). *Ingeniería del software: metodología y ciclos de vida*. Laboratorio nacional de calidad del software de inteco.

Jecrespom. (2019, marzo, 03). Aprendiendo arduino. Obtenido de https://aprendiendoarduino.wordpress.com/2019/03/03/buzzer-con-arduino/

Laboratorio I2.31 (26 de septiembre de 2009). *Filtros espaciales en imágenes-Visión artificial.* (Laboratorio). Universidad de los Llanos.

Lara Felipe. (2015). *Fundamentos de Redes Neuronales Artificiales.* Obtenido de: http://conceptos.sociales.unam.mx/conceptos_final/598trabajo.pdf

Leyva, G. (5 de julio de 2009). El cubre bocas o mascarilla, un recurso para garantizar la seguridad del personal de salud y del paciente. *Enfermería Universitaria*, 6. Obtenido de https://www.redalyc.org/pdf/3587/358741832007.pdf

Llamas, Luis (enero 25, 2016). Controlar Arduino con Python y la librería PySerial. Obtenido de https://www.luisllamas.es/controlar-arduino-con-python-y-la-libreria-pyserial/

Lugo, Arellano, Hernández (2017). Automatización de un sistema de inmersión temporal con base en plataformas abiertas de hardware y software. *Scielo.* Obtenido de http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0187-57792017000300269

Miranda, Solano, Méndez (2019). Introducción al Aprendizaje Automático con YOLO. *Revista de la Facultad de Ingenierías y Tecnologías de la Información y Comunicación. Vol. 2, Núm. 6.*

Montes, Roberto (2019). *Visión a nivel intermedio: Filtros.* Instituto tecnológico superior de San Andrés de Tuxtla, Veracruz.

Navarro, Kiara (2015). *Python + Arduino – Comunicación Serial*. Obtenido de http://panamahitek.com/python-arduino-comunicacion-serial/

Oliva, Alejandro. (2018). *Desarrollo de una aplicación de reconocimiento en imágenes utilizando Deep Learning con OpenCV.* (Trabajo de grado). Universidad Politécnica de Valencia.

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

Orozco, Andrés (2014). Diseño e implementación de un prototipo de plataforma robótica basado en Arduino para el desarrollo motriz de personas con discapacidad cognitiva. *Semillero de investigación Hardware Libre.* Obtenido de http://andresorozco.co/web/images/docs/Plataforma%20Robotica_Samir%20Granada_Final.pdf

Ortega, Francisco (2015). *Algoritmos de aprendizaje neuro computacionales para su implementación hardware.* Escuela Técnica Superior de Ingeniería Informática, Universidad de Málaga.

Perez, A., Berreteaga, O., Ruiz, A., Urkidi, A., & Perez, J. (2006). Una metodología para el desarrollo de Hardware y Software embebidos y Sistemas críticos de Seguridad. *Sistemas Cibernética e Informática,* 3(2),6-2

Ponce Pedro C. (2010). *Inteligencia Artificial con aplicaciones a la ingeniería*. Mexico D.F., Mexico: Alfaomega Grupo Editor, S.A. de C.V.

Porto. (2012). Definición de detector. Obtenido de https://definicion.de/detector/

Ramírez, Jiménez, Prieto (2015). Enseñanza del Procesamiento de Imágenes en Ingeniería usando Python. *IEEE-ES. Vol. 3, Núm. 4.*

Romero, Aaron (2016). *Sistema para la detección de engaños basado en redes neuronales y arduino*. Universidad Mayor de San Andrés. Bolivia.

Schweers, Robert (2002). *Descripción en VHDL de arquitecturas para implementar el algoritmo CORDIC.* (Tesis de Grado). Universidad Nacional de La Plata.

Silva, Cristian (2019). Electrónica para Todos con el Uso de Arduino: Experiencias Positivas en la Implementación de Soluciones Hardware Software. *Scielo.* Obtenido de https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642019000600377

Soria, D. (2009). Metodologías de Desarrollo (consultado: 11 de Noviembre de 2015).

Taquía (2017). El procesamiento de imágenes y su potencial aplicación en empresas con estrategia digital. *Interfaces.* Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=6230450

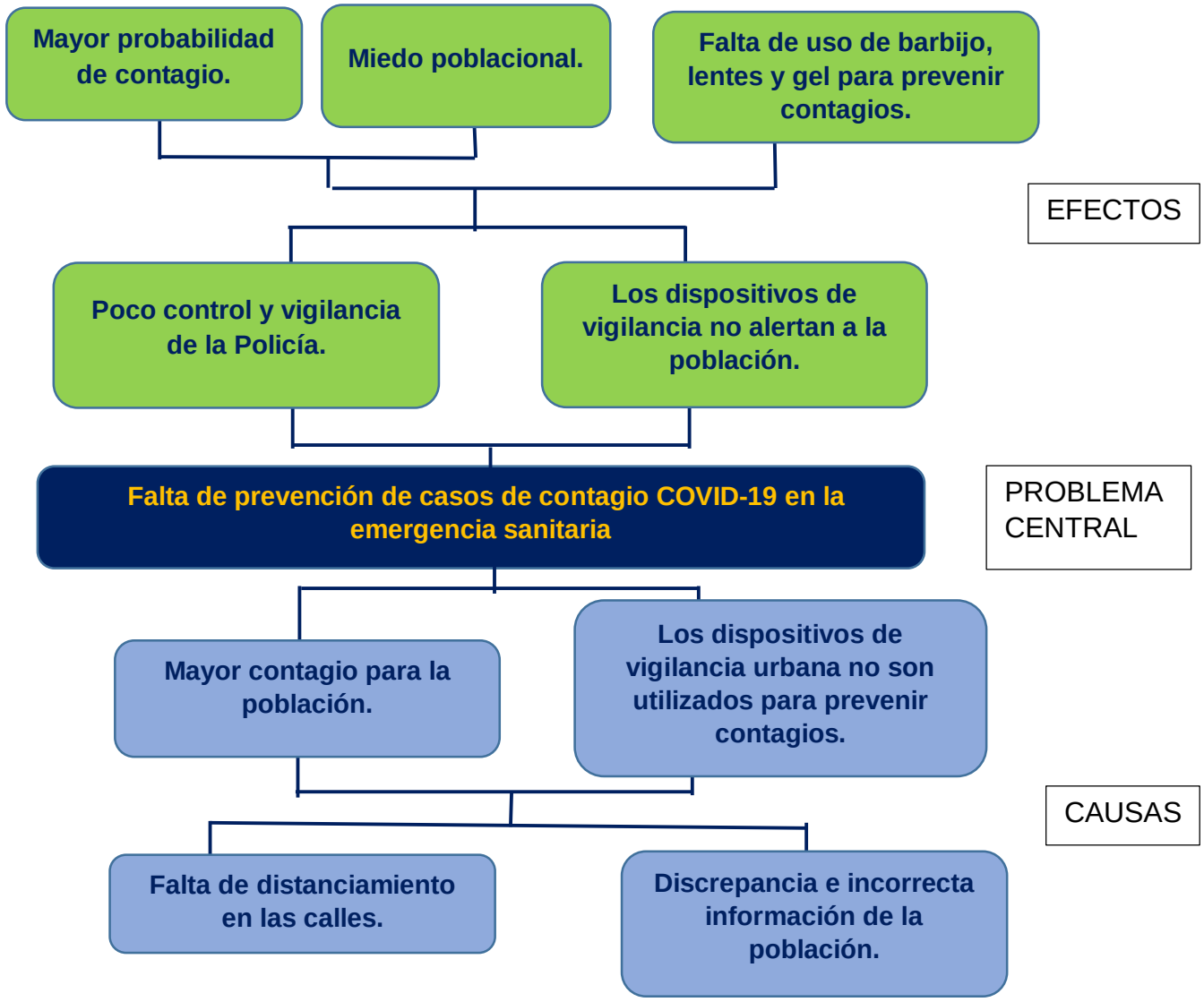
Vélez J., Moreno A., Sánchez A., & Sanchez J. (2002). *Visión por computador*. Obtenido de: http://www.visionporcomputador.es/libroVision/VisionPorComputador.pdf

Ventura, Víctor (2018). PySerial. https://polaridad.es/tag/pyserial/

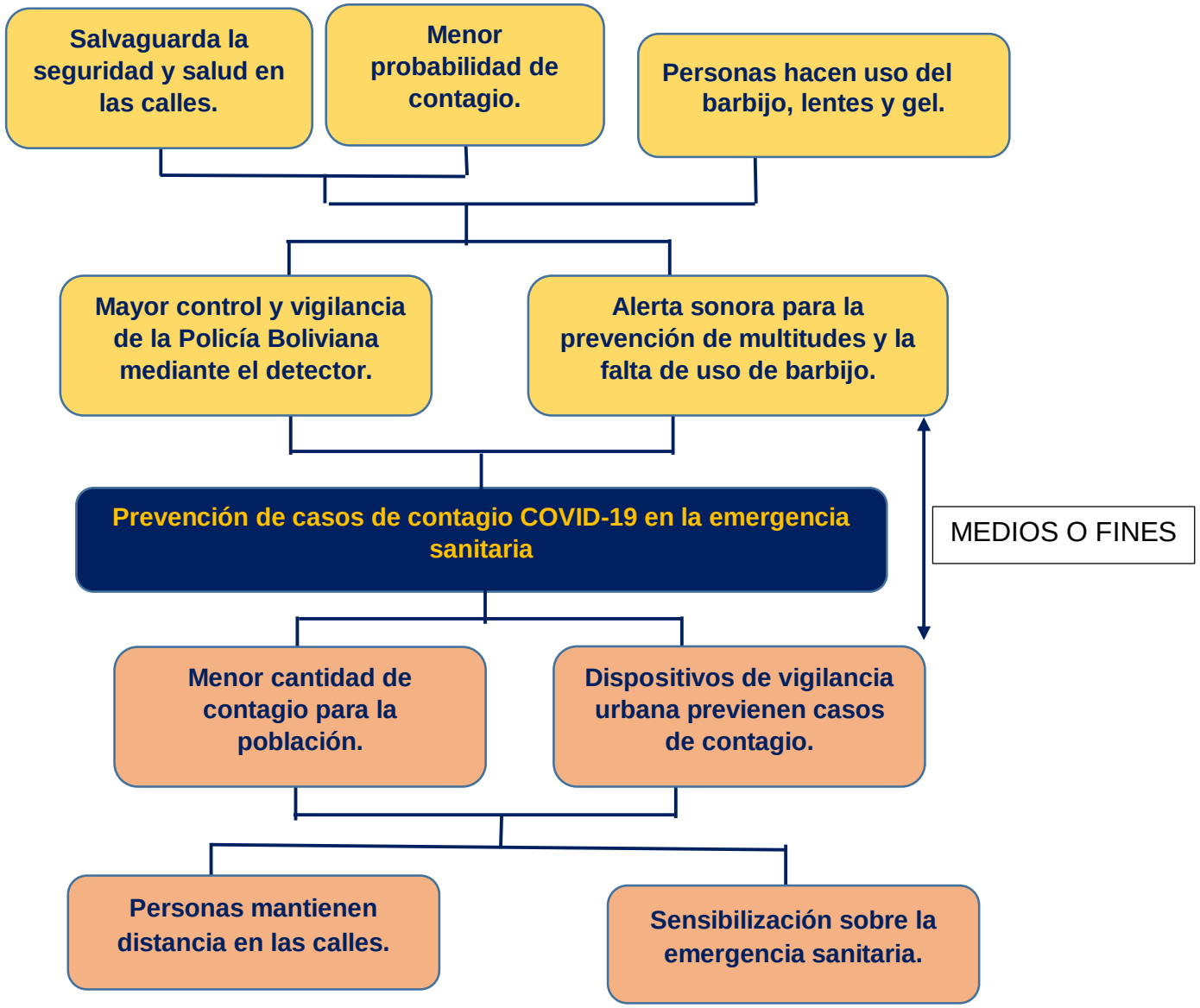
Warden, Situnayake (2019). *Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers.* Obtenido de https://tinymlbook.files.wordpress.com/2020/01/tflite_micro_preview.pdf

ANEXOS

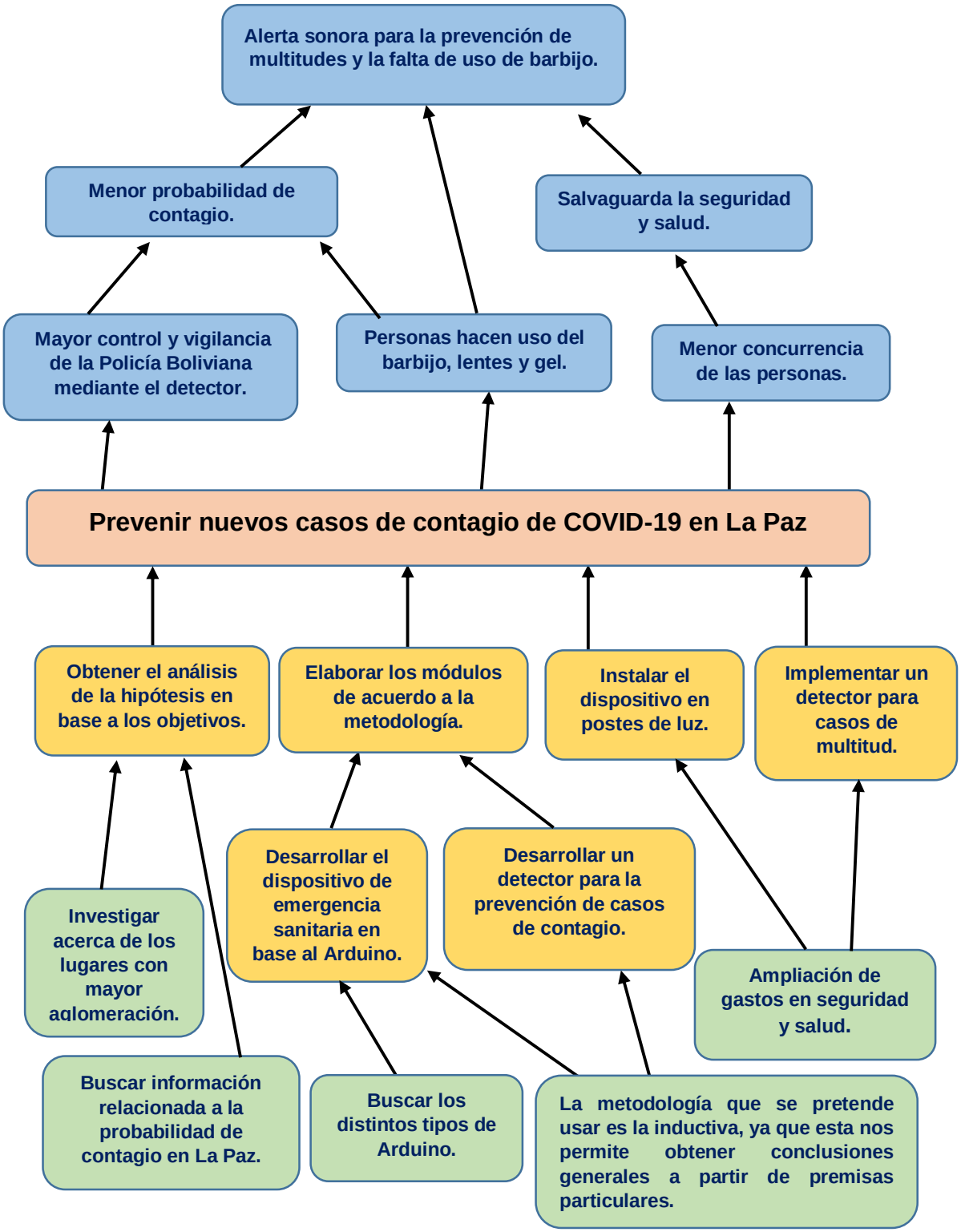
ANEXO 1: ÁRBOL DE PROBLEMAS



ANEXO 2: ÁRBOL DE OBJETIVOS



ANEXO 3: ÁRBOL DE EFECTO



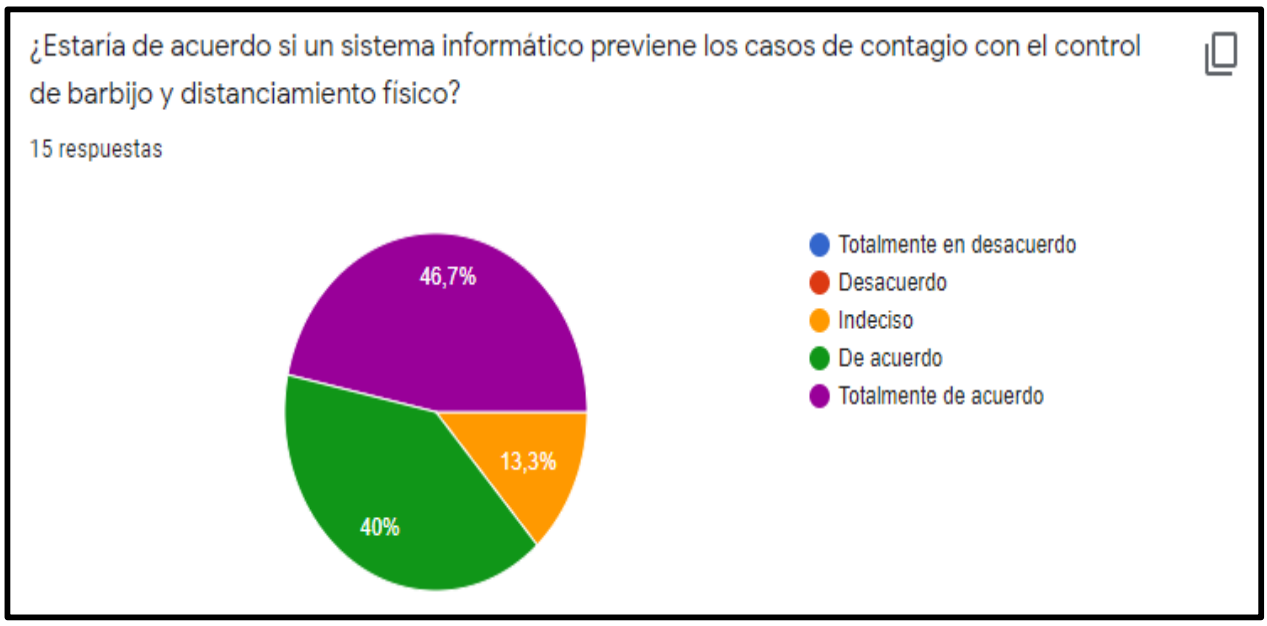
ANEXO 4: MARCO LÓGICO

DESCRIPCIÓN	INDICADORES (PRESUPUESTO)	VERIFICADORES (MEDIOS DE VERIFICACIÓN)	SUPUESTOS
Fin Prevenir los casos de contagio mediante una alerta sonora, para prevenir las multitudes y la falta de uso de barbijo.	Obtenemos menores casos de aglomeraciones e incremento de educación sanitaria en cuanto al uso de barbijo, mediante la tecnología la cual satisfaga las expectativas prácticas en el uso y manipulación de herramientas informáticas.	Recolección de datos según las necesidades de los involucrados.	Tenemos información relacionada al COVID-19 en La Paz-Bolivia y la cantidad de casos por día. Información de alarmas comunitarias.
Propósito Prevenir los casos de contagio de COVID-19 en la emergencia sanitaria, mediante un detector de restricciones con el uso de procesamiento de imágenes, para la detección de aglomeraciones y uso de barbijos elaborado en base a Arduino y redes neuronales.	Para la finalización de tesis propuesta en noviembre 2020 se implementará el detector de restricciones para prevención para casos de contagio en la emergencia sanitaria basado en Arduino y redes neuronales.	• Evaluación del detector de restricciones por medio de encuestas y pruebas de satisfacción. • Evaluar el proyecto mediante la prevención y protección sanitaria de cada persona en las calles.	• Existen tesis similares, las cuales mencionan los dispositivos Arduino. • Información de Algoritmos para redes neuronales, Procesamiento de imágenes y el Microcontrolador Arduino. • Se tiene datos actualizados sobre nuevos casos de COVID-19 en La Paz-Bolivia.
Componentes C1: Hipótesis en base a los objetivos. C2: Módulos de requerimiento, análisis-diseño y desarrollo. C3: Detector de restricciones para la prevención de casos de contagio realizado. C4: Dispositivo de emergencia sanitaria en base al Arduino y redes neuronales. C5: Concientización para la población sobre el COVID-19.	C1: Desarrollar la hipótesis. C2: Solución del Modelo teórico. C3: Detector de restricciones para prevención de casos de contagio en la emergencia sanitaria mediante Arduino y redes neuronales implementado hasta octubre 2020. C4. Dispositivo de emergencia sanitaria elaborado hasta octubre 2020. C5: Las pruebas del detector de restricciones. C6: Mayor cantidad de personas cumplen con el cuidado para la emergencia sanitaria.	• Planteamiento de la hipótesis mediante pruebas. • Elaboración de la documentación de la tesis. • Entrega de documentación elaborada. • Los informes que realizarán revisor y tutor.	• Se cuenta con información de las zonas que infringen la cuarentena. • Se posee la información sobre seguridad y salud de parte de las autoridades. • Se tiene libre software para la realización del dispositivo de emergencia sanitaria. • Disposición de una variedad de herramientas para el desarrollo del detector de restricciones. • Concientización de parte de las autoridades o personas afines.
Actividades • Investigar sobre los lugares donde no hay educación sanitaria y barrios con mayor aglomeración. • Buscar información relacionada a la probabilidad de contagio en La Paz. • Buscar los distintos tipos de Arduino para la alarma. • La metodología que se pretende usar es la inductiva, ya que esta nos permite obtener conclusiones generales a partir de premisas particulares. • Ampliación de gastos en seguridad y salud.	• Material de consultas: internet, libros de consultas de forma gratuita. • Material de escritorio, registros de encuestas online. • Material de internet, visitas para las entrevistas. • Componentes para la instalación del dispositivo. • Dispositivo de Arduino y componentes. Módulo electrónico ARDUINO UNO 60Bs Componentes -Sensor Buzzer Bs. 10 -Componentes para conexión Bs 50	• Documento final de la tesis. • Resultados obtenidos de los distintos datos de investigación. • Documentación obtenida de la documentación del estudio de la calidad de seguridad.	• Recopilar información necesaria y adecuada para la elaboración de la tesis de forma correcta. • Contar con el apoyo de las autoridades y la población para la aplicación del método propuesto. • Obtener la información necesaria de los casos con aglomeraciones y falta de uso de barbijo para alcanzar el propósito. • Contar con las herramientas y los materiales necesarios.

ANEXO 5: ENCUESTA.

No.	PREGUNTAS
1	¿Usted sale con frecuencia de su hogar a pesar de la pandemia?
2	¿Usted hace uso del barbijo?
3	¿Usted cumple con la distanciamiento físico mínimo de 1 metro?
4	¿Tiene conocimiento sobre las alarmas comunitarias instaladas en la ciudad?
5	¿Estaría de acuerdo si un sistema informático previene los casos de contagio con el control de barbijo y distanciamiento físico?
6	¿Usted estaría de acuerdo en la instalación de alarmas para la prevención de casos de contagio?
7	¿Qué sugerencia daría para prevenir los casos de contagio?

ANEXO 6: RESPUESTAS DE LA ENCUESTA.



DOCUMENTACIÓN