UNIVERSIDAD SAN FRANCISCO DE QUITO
USFQ

Colegio de Ciencias e Ingeniería

Modelo de detección de mascarillas faciales en un sistema embebido en un vehículo autónomo en tiempos de COVID-19.

Andrés Sebastián Benavides Guerrero
Diego Francisco Pazmiño Lopez

Ingeniería Electrónica

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero electrónico

Quito, 23 de diciembre de 2020

UNIVERSIDAD SAN FRANCISCO DE QUITO
USFQ

Colegio de Ciencias e Ingeniería

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Modelo de detección de mascarillas faciales en un sistema embebido en un vehículo autónomo en época del COVID-19.

Andrés Sebastián Benavides Guerrero
Diego Francisco Pazmiño Lopez

René Játiva Espinoza, Ph.D.

Quito, 23 de diciembre de 2020

© DERECHOS DE AUTOR

Nombres y apellidos: Andres Sebastián Benavides Guerrero

Código: 00139634

Cédula de identidad: 1727317305

Lugar y fecha: Quito, 23 de diciembre de 2020

Nombres y apellidos: Diego Francisco Pazmiño Lopez

Código: 00136489

Cédula de identidad: 1721625232

Lugar y fecha: Quito, 23 de diciembre de 2020

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En un vehículo autónomo terrestre se implementó inteligencia artificial sobre un sistema embebido Raspberry pi 3 B usando Open CV de Intel y TensorFlow de Google, proporcionándole la capacidad de reconocer rostros de personas y analizar si estas portan o no una mascarilla.

A finales del 2019 surgió una emergencia sanitaria que forzó al distanciamiento social y el uso de la mascarilla se tornó obligatorio alrededor del mundo. El sistema diseñado e implementado en este robot autónomo permite analizar en tiempo real si una persona porta o no mascarilla dentro de su sitio de trabajo, tornándose en un dispositivo de vigilancia que ayude a reducir la tasa de contagios y por ende el número de decesos debido a la enfermedad.

Para realizar este control, el vehículo robot observa su entorno por medio de una cámara web y posibilitando la reproducción de un video en tiempo real con esta información en cualquier dispositivo portátil. En caso de que el vehículo detecte que una persona no porta mascarilla, automáticamente tomará una foto del rostro de la persona y se la enviará a una base de datos o centro de supervisión para que la administración pueda tomar las medidas correspondientes.

Palabras claves: *Inteligencia artificial, machine learning, vehículo autónomo, detección tiempo real, COVID-19.*

ABSTRACT

In an autonomous land vehicle, artificial intelligence was implemented on a Raspberry pi 3 B embedded system using Intel's Open CV and Google's TensorFlow, giving it the ability to recognize people's faces and analyze whether they are wearing a mask.

At the end of 2019, a health emergency arose that forced social distancing and the use of the mask became mandatory around the world. The system designed and implemented in this autonomous robot allows to analyze in real time whether a person wears a mask within their workplace, becoming a surveillance device that helps reduce the rate of infections and therefore the number of deaths due to disease.

To carry out this control, the robot vehicle observes its environment through a web camera and enables the reproduction of a video in real time with this information on any portable device. If the vehicle detects that a person does not wear a mask, it will automatically take a photo of the person's face and send it to a database for monitoring center so that the administration can take the corresponding measures.

Keywords: *Artificial intelligence, machine learning, autonomous vehicle, real-time detection, COVID-19.*

TABLA DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

INTRODUCCIÓN.

Frente a lo sucedido a finales del año 2019 con la aparición de un virus desconocido para las personas, un nuevo tipo de coronavirus, produciendo la enfermedad Covid-19 y llegando a declararse pandemia por la OMS. Las medidas de confinamiento se han ido expandiendo a lo largo del mundo y la comunidad científica tratando de entender esta nueva enfermedad (Darlington, 2020). Actualmente en nuestro siglo, la tecnología es el eje principal para el desarrollo de cualquier sector y no es de sorprender que en la búsqueda de una solución a la pandemia la tecnología esté involucrada, pero existe un sector que su contribución puede llegar a ser muy importante para la lucha contra el COVID-19. El sector de la inteligencia artificial ha estado presente en cada una de las etapas de esta pandemia, con la aparición de nuevos software que nos permiten desde un control de la enfermedad, un diagnostico o para prevención (SANOFI, 2020).

La inteligencia artificial (IA) tiene el potencial para revolucionar la forma de inventar y pensar del ser humano. La IA aumenta de manera significante la eficiencia de la economía existente (Quintero, 2020), este campo se centra en la creación de programas que muestran comportamientos o capacidades similares en la toma de decisiones, a las de un ser humano, considerando entonces al sistema como inteligente. Un sistema IA es capaz de analizar gran cantidad de datos, después se encarga de identificar patrones en los datos y lograr formular predicciones de forma automática rápidamente y el objetivo de este sistema es que pueda ser incorporado en alguna máquina o dispositivo para que este tenga “inteligencia propia” (salesforce, 2017).

Existen técnicas de inteligencia artificial de nuestro interés tales como el machine learning y el deep learning.

Machine learning (ML) se encarga de hacer que las computadoras realicen acciones sin necesidad de programación explícita; es decir, usa algoritmos para organizar datos, reconocer patrones y hacer que una computadora pueda ser "inteligente". Estos algoritmos aprenden con los datos a los que son sometidos y así las maquinas van siendo entrenadas para que ejecuten una acción de manera autónoma. Las maquinas toman decisiones a partir de modelos previamente creados en los que se suministraron los datos y se buscaron patrones (salesforce, 2018). Mientras que el deep learning es una forma de ML que permite a las computadoras aprender de la experiencia y comprender el mundo en términos de una jerarquía de conceptos (Goodfellow., Bengio., Courville. & Bengio. 2016).

Entonces, ¿podría un robot prevenir la propagación del COVID-19 usando inteligencia artificial? El propósito de este proyecto es principalmente implementar un modelo de detección de mascarillas faciales en un sistema embebido de un vehículo autónomo usando la cámara web de este vehículo. Con el uso de esta aplicación se podrá controlar al personal de una empresa, por ejemplo, en el uso obligatorio de la mascarilla, obteniendo información de los empleados que incumplieron las normas y así facilitar la mitigación de la propagación del virus. Al ser un robot el que realiza de manera autónoma la vigilancia in situ, se prescinde de que una persona controle este proceso.

DESARROLLO

Vehículo autónomo y sistema embebido propuesto.

En este proyecto se propone hacer el uso de un vehículo autónomo el cual posee un sistema embebido encargado del funcionamiento de todos de los módulos del vehículo para que este funcione adecuadamente y cumpla con todas las funcionalidades programadas e implementadas.

El sistema embebido del vehículo autónomo es un Raspberry Pi 3B encargado de procesamiento de datos y comunicación entre los módulos. Este sistema se encarga de recolectar toda la información que han tomado los sensores del vehículo. Ésta data se procesa dentro del Raspberry Pi 3B y posibilita que el vehículo funcione y ejecute cada una de sus funciones. (Barbosa & Hernández, 2020). A continuación, en la figura (1) se muestra el diagrama de los módulos del vehículo.

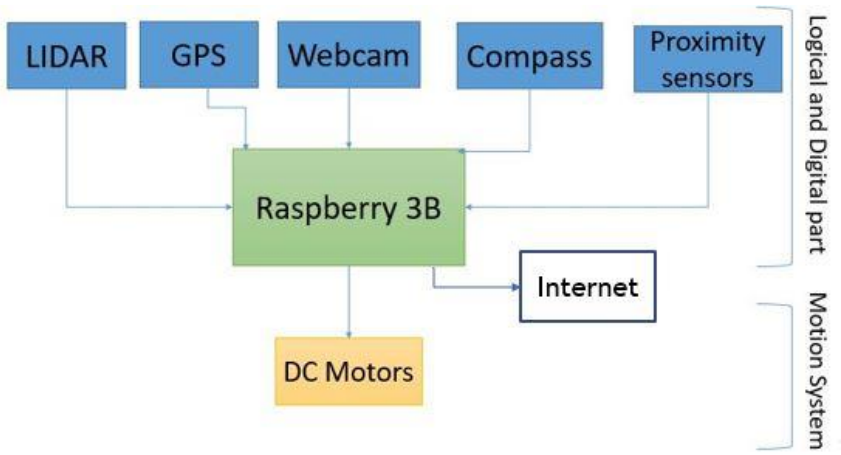


Figura1. Sistema embebido 1 con sus módulos. (Barbosa & Hernández, 2020).

Para cumplir los objetivos de este proyecto, se incorporó otro sistema embebido al vehículo, basado en un Raspberry pi 3B+ adicional, que se enfoca exclusivamente en la inteligencia artificial del sistema de vigilancia del vehículo, usando los módulos de internet y de la cámara web. Los dos Raspberry se comunican por hardware, haciendo

que uno envié señales al otro dependiendo de si el software detecta una persona sin mascarilla. Ambos sistemas usan la misma batería como su fuente de alimentación disminuyendo el tiempo de operación del vehículo. A continuación, en la figura (2) se muestra el diagrama de bloques del segundo sistema embebido.

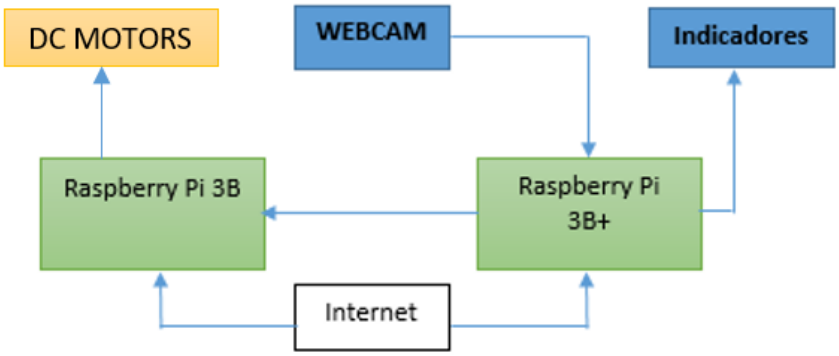


Figura 2. Conexión entre sistemas embebidos y módulos del segundo.

- **Vehículo autónomo y sistema embebido 1**

El sistema embebido 1 basado en el Raspberry Pi 3B se encuentra implementado en el vehículo autónomo y usa un sistema operativo Linux con distribución Ubuntu Mate 16.04 LTS (Xenial Versus). Este soporta ROS (robot Operating System) con todos los paquetes de software actualizados. (Barbosa & Hernández, 2020).

ROS es un framework para escritura de software de robot, ROS es un meta-sistema operativo que provee los servicios de: abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, mantenimiento entre paquetes, etc.

"ROS está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros." (Ortego, 2017).

El vehículo usa ROS ya que permite dividir el funcionamiento general de un robot en varios comportamientos específicos y luego poder enlazarlos y que puedan trabajar juntos.

El robot tiene varias funciones ya previamente instaladas y enlazadas por medio de ROS, estas son las siguientes:

- El vehículo se mueve manual y automáticamente, evitando las colisiones de la manera automática por medio de los sensores de proximidad.
- Ubicación y seguimiento del vehículo en tiempo real con el Api de Google Maps.
- SLAM o mapeo y localización simultáneos; es decir, construye un mapa en 2D del entorno en el que se encuentra.
- Cámara web para observar la trayectoria.

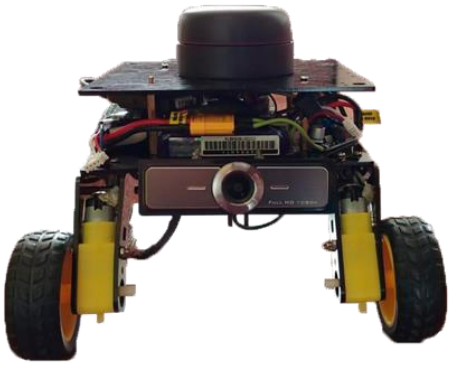


Figura 3. Vehículo con sistema embebido 1. (Barbosa & Hernández, 2020).

- **Sistema embebido 2 (Raspberry Pi 3B+)**

Para el segundo sistema embebido se usó la distribución del sistema operativo GNU/LINUX Raspberry Pi OS (Raspbian) basada en Debian. Este sistema se enfoca principalmente en el machine learning que se implementó en este proyecto. El objetivo principal de usar otro Raspberry ajeno al ya existente en el vehículo se debe a que se

necesitan recursos adicionales para realizar el procesamiento del machine learning, ya que las funciones del sistema embebido existente consumen toda la memoria y los recursos de procesamiento del sistema original 1.

Este nuevo Raspberry Pi 3B+ ejecuta el programa de reconocimiento de mascarillas faciales y dependiendo de la respuesta se encienden los indicadores y se envía simultáneamente una orden al sistema embebido 1 para que pare los motores y detenga el vehículo hasta que la persona cumpla con el requerimiento de ponerse la mascarilla.

La comunicación entre ambos sistemas se realiza de manera digital enviando una señal del sistema 2 al 1 dependiendo de lo que detecte la cámara web. Esta señal va a pasar por un cable de comunicación para que la respuesta sea inmediata y reducir el delay al mínimo posible.

Figura 4. Vehículo con ambos sistemas.

- **Webcam.**

El sistema 1 usa una cámara web que permite un video en Full HD (1080p), sin embargo, opera a resolución estándar (SD) para optimizar el procesamiento del sistema. Para este proyecto se optó por cambiar la cámara debido a que la original posee un ángulo de visión demasiado grande, dificultando que el software de la aplicación de vigilancia detecte las mascarillas faciales debido a que el modelo utilizado fue entrenado con una cámara con otro ángulo de visión. La nueva cámara se instaló en el sistema 2 usando su puerto USB para vincularla con el software y arrojar datos correctamente. Esta cámara tiene una resolución de 640x480 VGA.

- **Indicadores**

Los indicadores son componentes electrónicos básicos encargados de alertar, por medio del sentido de la vista y del oído, si se detectó a una persona sin mascarilla facial. Estos indicadores son 1 LED con un voltaje de trabajo de 3V y un buzzer. Puesto que el pin de salida del Raspberry tiene un voltaje de 3.3V se protegió el diodo usando una resistencia de valor R, calculada como sigue:

$$R = \frac{3.3[V] - 3[V]}{15[mA]} = 20[\Omega]$$

El otro indicador, de tipo auditivo, es un buzzer que emite una señal auditiva para dar a conocer que la persona no usa mascarilla. Este buzzer se conecta a un pin del sistema 2 directamente.

- **Internet**

El sistema 1 se conecta a la red wifi para que funcione el framework de ROS. El sistema 2 se conecta a la misma red de wifi haciendo posible la trasmisión de la cámara web hacia cualquier dispositivo que esté conectado a la misma red. Cuando el sistema 2

detecta una persona sin mascarilla toma capturas por frame; es decir guarda el frame en el que no se detectó mascarilla facial y la envía a una carpeta compartida que se puede ver de un dispositivo conectado a la misma red.

Diagrama de conexión.

En la figura (5) se presenta el diagrama de conexiones del segundo sistema, en el que están conectados los respectivos indicadores. De igual manera la fuente de alimentación del sistema 2 es el pin 1 que es enviado al sistema 1 debido a que se alimentan con la misma batería.

La batería que alimenta al sistema 1 tiene un voltaje nominal de 11.1V y usa un conversor DC-DC de 5V y 3A para poder regular el voltaje a 5.15V. Debido a que usamos la misma batería para el segundo sistema, se necesita de un nuevo conversor DC-DC de 5v y 5A, ya que ahora trabajamos con dos Raspberry's que necesitan 2.5A cada uno y así la batería original abastece a ambos sistemas.

El sistema 1 que usa el prototipo permite una autonomía promedio de alrededor de 24 minutos. Ahora que usamos un segundo sistema que comparte la batería, su tiempo de funcionamiento es menor, decayendo aproximadamente a 16 minutos al usarse todos los módulos de ambos sistemas.

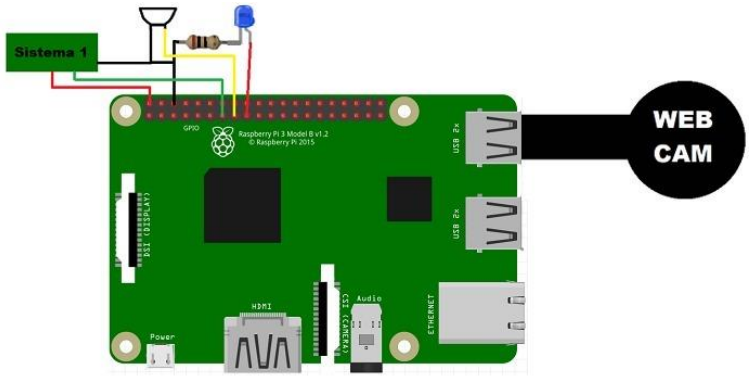


Figura 5. Diagrama de conexiones.

Implementación del reconocimiento de mascarillas faciales

En esta sección se describe detalladamente como se implementó el software en el sistema embebido 2, y se describen los modelos pre-entrenados que hacen posible el reconocimiento de caras y mascarillas faciales.

Código third-party utilizado

Para poder implementar y poner en marcha el sistema de reconocimiento de mascarilla se incorporó varias librerías de código abierto en Python. Durante todo el desarrollo del proyecto se utilizó la versión de Python 3.7.3, la cual está disponible desde el 25 de julio del 2020. Se instaló OpenCV v4.2.0 de Intel, que es una librería de código abierto en la que se encuentra varios modelos de machine learning y está enfocado a la visión artificial con modelos optimizados para una rápida respuesta (OpenCV, 2015). Adicionalmente se instaló TensorFlow v2.1.0 de Google, ya que el modelo de reconocimiento de mascarillas fue preentrenado con MobileNetV2 que es una herramienta de diseño orientada para trabajar con redes neuronales en dispositivos móviles o de escasos recursos de procesamiento. (Parreño, 2019)

El enfoque del proyecto no es entrenar una red neuronal, sino más bien proporcionar una solución para el reconocimiento de mascarillas usando una red

previamente entrenada. El reconocimiento de mascarillas está conformado por dos modelos, uno que detecta la presencia de una cara y otro que en base a la respuesta del primer modelo hace la clasificación de si existe o no existe una mascarilla en el rostro de la cara detectada para poder tomar una decisión. Esto nos ayuda a reducir el tiempo de implementación del sistema, ya que para entrenar el modelo se necesita crear un dataset, el cual está compuesto de imágenes positivas y negativas. Para el primer modelo las imágenes positivas se refieren a aquellas que contienen varios rostros de personas y como negativas las imágenes que no contienen un rostro. Para el segundo modelo se deben ingresar rostros con mascarilla y rostros sin mascarilla. El tamaño del dataset usado para el modelo de detección facial es de 5000 rostros, mientras que el detector de mascarillas es de alrededor de 4000 rostros entre caras con mascarilla y otras sin mascarilla. Cabe recalcar que para entrenar un modelo de machine learning mientras mas grande sea el dataset mas preciso será el detector, pero a su vez el entrenamiento requiere de más tiempo.

Software del reconocimiento de mascarillas faciales.

En esta sección hablaremos de como está compuesto el código y como se usaron cada uno de los modelos pre-entrenados. A continuación, en la figura (6) se muestra el diagrama de bloque indicando cómo funciona el código y como viaja la información de la imagen a través de cada uno de los bloques y finalmente tomando una decisión positiva o negativa dependiendo si la persona porta o no mascarilla.

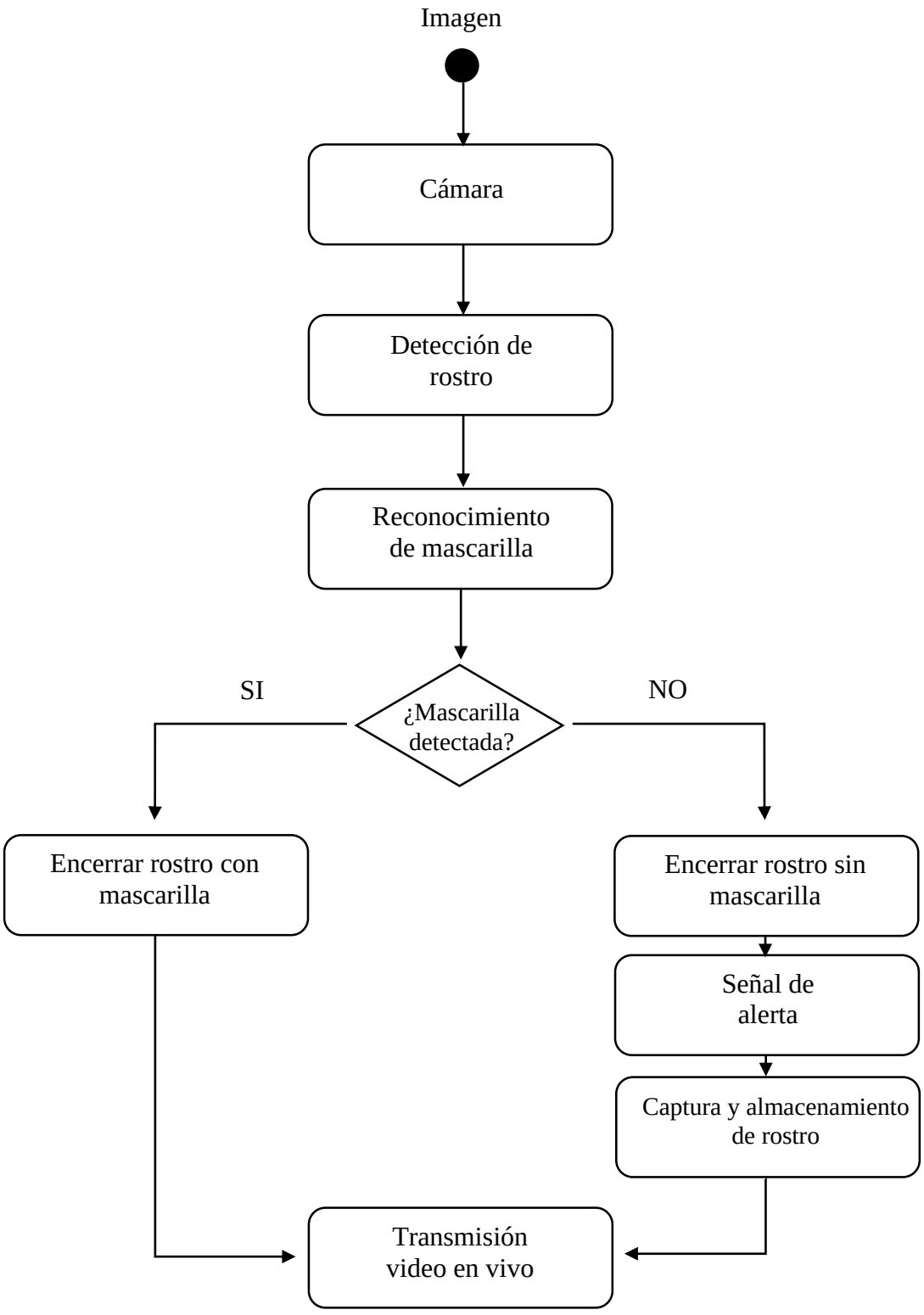


Figura 6. Diagrama de bloques del código implementado

1. **Detección de imagen**

La detección de imagen se hace mediante una cámara web instalada en el Raspberry por el puerto USB. La resolución de la cámara es de 640X480 VGA.

Detección de rostro

El propósito de este paso es detectar en una imagen si existe o no un rostro, por lo que la detección facial se hace mediante el modelo clasificador. Actualmente existen varios métodos de reconocimiento facial entre ellos está el método Viola-Jones y el método HOG que divide la imagen en cuadriculas y en cada cuadricula computa una gradiente para poder comparar con valores entrenados. Estos métodos son comúnmente usados en dispositivos móviles por su velocidad en detección y el poco uso de recursos para el procesamiento (Cardona & Pineda, 2019). Se empleará el algoritmo de Viola-Jones ya que presenta una rápida detección usando acelerador de cascada basado en características simples de Haar. Este algoritmo tiene 4 partes principales:

- **Extracción de características de Haar**

  Este algoritmo hace su predicción en base a características y no pixeles, lo cual es una forma de acelerar su procesamiento. Existen 3 tipos de características: dos rectángulos, tres rectángulos y cuatro rectángulos.

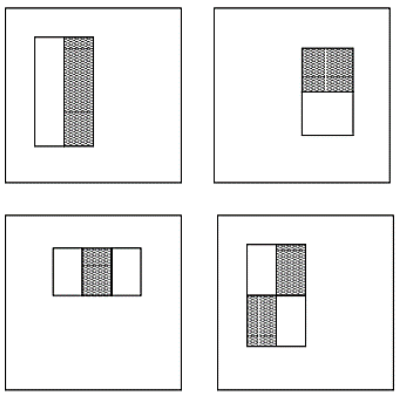


Figura 7. Características de Haar (Viola & Jones, 2001).

Cada característica se compara con la imagen de entrada en escala de grises por lo que las zonas mas oscuras pueden estar representadas por ojos, cejas, líneas faciales o simplemente profundidad. Para el siguiente paso se resta la suma de pixeles del recuadro negro de la suma de pixeles del recuadro blanco obteniendo un valor representativo de la intensidad en escala de grises. (Viola & Jones, 2001)

- **Imagen integral**

  Del paso anterior se obtiene una cuadrícula en la que se muestra el resultado de la suma de pixeles en base a la intensidad de la imagen representada en escala de grises. Para el desarrollo de cada recuadro en una imagen integral, se suma los recuadros izquierdos y arriba obtenidos en la imagen original de escala de grises. Cada recuadro se calcula secuencialmente en cada fila. Con la imagen integral se puede calcular regiones rectangulares de la imagen de forma más rápida que haciendo un cálculo pixel por pixel (Viola & Jones, 2001). El método de ilustra en las matrices de la Figura 8.

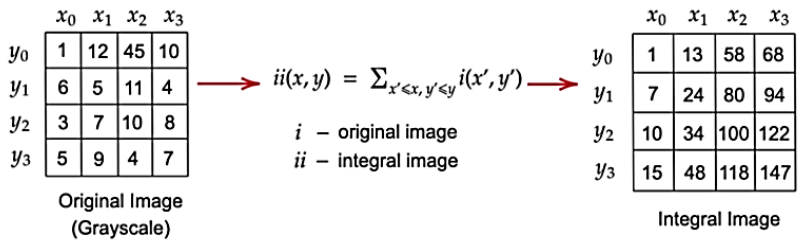


Figura 8. Algoritmo desarrollo imagen integral

- **Acelerador AdaBoost**

  En esta etapa es donde se obtiene el clasificador robusto, ya que de los procesos anteriores se obtiene las características y se las combina linealmente para poder hacer un clasificador más robusto. En cada

iteración del algoritmo se busca las que exhiben el menor error posible para clasificarlas como candidatas a un reconocimiento positivo. Las que mayor error presentan corresponderían a los casos negativas. (Viola & Jones, 2001)

- **Clasificador de cascada**

  Es un clasificador de varias etapas conectadas en cascada para una detección rápida del rostro. Estas etapas se encuentran ordenadas desde los clasificadores más fuertes hasta los más débiles obtenidos en el algoritmo anterior. Cuando una imagen entra al clasificador esta es comparada rápidamente y si cumple con la etapa 1 pasa la siguiente, caso contrario se elimina la imagen (Viola & Jones, 2001). El proceso se ilustra en la Figura 9.

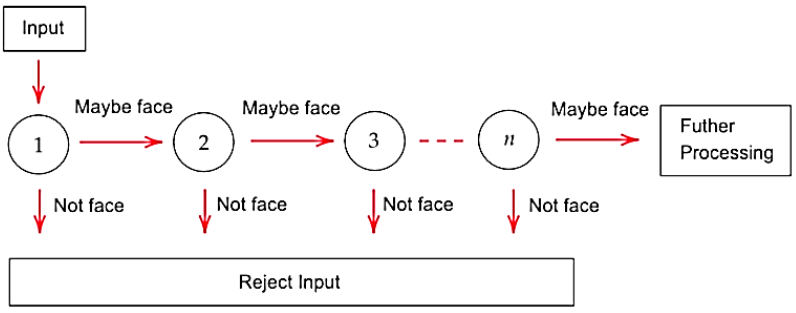


Figura 9. Clasificador Haar-Cascade

El algoritmo implementado usa el clasificador Haar-Cascade, por lo que al momento de capturar la imagen de la cámara ésta se convierte a escala de grises para que el clasificador pueda comparar las características de Haar entrenadas con las de la imagen de entrada (Viola & Jones, 2001).

2. Detección de mascarilla

En ese paso se obtiene el frame que contiene el rostro y se analiza en base a una clasificación si tiene o no tiene mascarilla. Se utiliza un modelo ya entrenado en MobileNetv2 ya que por el bajo uso de recursos de CPU puede ser fácilmente implementada en el Raspberry (Mujtaba, 2020). El clasificador básicamente de la imagen de entrada realiza una conversión a RGB y se redimensiona la imagen a 224x224 pixeles y mediante la convolución con filtros pre-entrenados y una reducción de la imagen se puede obtener el valor de predicción de la entrada. Este valor puede ser configurado para indicar las salidas, si supera cierto valor se toma una decisión, caso contrario se niega. En el clasificador implementado se configura de tal forma que el predictor de portar mascarilla es mayor que el de sin mascarilla el algoritmo indique que el rostro detectado si tiene mascarilla (Sandler, Howard, Zhu, Zhmoginov & Chen, 2018).

3. Output

Una vez clasificada la imagen se tiene 2 posibles opciones: se usa mascarilla o se prescinde de ella en el rostro detectado en cada frame. Con relación a la información de salida del clasificador se toma una decisión.

- **Si se encuentra mascarilla**

Si el clasificador reconoce que el rostro está usando mascarilla, se muestra un rectángulo verde en el rostro de la persona que la usa, como se muestra en la Figura 10.

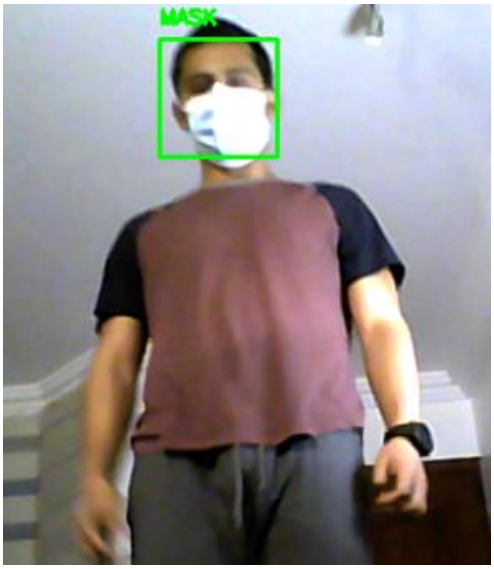


Figura 10. Resultado de frame positivo con mascarilla

Este resultado se muestra en vivo mediante el uso de un visualizador de servidor VNC, que nos permite observar lo que muestra la cámara.

- **Si se encuentra sin mascarilla**

Si el clasificador reconoce que el rostro no está usando mascarilla, se muestra un rectángulo rojo en el rostro de la persona, como se muestra en la Figura 11. Adicionalmente el vehículo autónomo se detiene al informarse de esta situación mediante la señal que recibe del sistema #2. Esta señal básicamente es un 1 lógico que funciona como interrupción para el sistema #1. En el vehículo se detecta la señal de alerta y se deshabilitan los motores hasta que la persona se ponga la mascarilla. Paralelamente en el sistema #2 existen señales visuales para que el usuario sea amonestado y se percate de la infracción que significa el no uso de la mascarilla. Estas señal es una luz LED que parpadea repetidamente como advertencia visual y un buzzer en sonido constante como advertencia sonora.

Figura 11. Resultado de frame sin mascarilla.

El sistema está programado para guardar el frame que se detectó sin mascarilla en una carpeta compartida con el ordenador que controla ambos sistemas, como se ilustra en la Figura 12. La transferencia de datos entre los sistemas operativos es de forma instantánea y se logra mediante Samba, el cual es un software que permite la transferencia de archivos mediante el protocolo SMB (Server Message Block) para la comunicación entre distintos sistemas operativos (Huili, Z,2008).

Para habilitar la carpeta compartida fue necesario instalar Samba en el Raspberry y modificar las configuraciones con el editor de texto en terminal *nano* a la carpeta que se quiere compartir; de esa forma se habilita para que ésta pueda ser visualizada desde otro ordenador ingresando la IP del sistema que está compartiendo y las credenciales pertinentes.

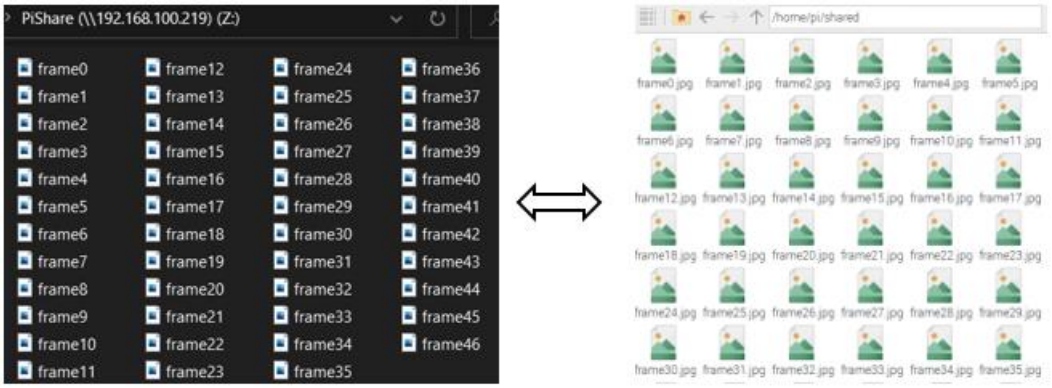


Figura 12. Visualización de sincronización de carpeta en diferentes sistemas operativos

Una vez finalizado el recorrido del vehículo autónomo se obtiene la carpeta en donde están los rostros de personas sin mascarillas encontrados en todo el trayecto. Como se guarda cada frame detectado, es necesario hacer un filtrado para eliminar las fotos similares, minimizar la cantidad de imágenes repetidas y facilitar la identificación de la persona. De igual forma, el video se muestra en tiempo real con el servidor VNC en el ordenador de control.

En términos de seguridad, no habría inconvenientes al transmitir la información mediante el internet ya que al habilitar la carpeta compartida esta puede visualizarse desde cualquier ordenador conectado a la misma red Wi-FI, pero hay que tener en cuenta que al momento de crear la carpeta se le asigna una contraseña para evitar cualquier violación. En forma análoga, el servidor VNC dispone de credenciales que posibilitan el acceso seguro a la visualización en tiempo real de la cámara y del sistema 1, a la interfaz de *rviz* para ver el SLAM obtenido por el sensor LIDAR junto con la trayectoria del vehículo.

Evaluación de resultados

- **Test de precisión algoritmo**

Con fines de evaluar la precisión del reconocedor de mascarillas se plantea realizar un script que pueda indicarnos el número de detecciones acertadas sobre una base de datos. Debido a la actual pandemia no se pudo recolectar con el mismo vehículo autónomo un número de imágenes sustentable para minimizar el error de precisión. Se optó por usar un dataset de imágenes de rostros con mascarilla y sin mascarilla, mismo dataset que cuenta con 1915 imágenes de personas con mascarilla y 1918 imágenes de personas sin mascarilla. El script desarrollado básicamente cuenta de un ciclo *for* que permite contabilizar si una mascarilla fue detectada o no, almacenando esta información en dos contadores independientes. Los indicadores que evalúan el proceso se muestran en las tablas 1 y 2 respectivamente.

Como resultado de este script se obtiene lo siguiente:

		Condición verdadera	
		Con mascarilla	**Sin mascarilla**
Condición prevista	**Con mascarilla**	1763	96
	Sin mascarilla	152	1822

Tabla 1. Matriz de confusión

	Con mascarilla	**Sin mascarilla**
Precisión	94,80%	92,30%
Exhaustividad	92,10%	95%
Exactitud del modelo	93,53%	

Tabla 2. Resultados matriz de confusión

Cabe recalcar que el dataset cuenta con rostros de distintos géneros, de igual forma se encuentran a diferentes distancias desde el punto de captura por lo que nos da mayor seguridad en la prueba del algoritmo. En base a los resultados de esta prueba se puede concluir que el error no es muy representativo y se es factible implementar el algoritmo de detección de mascarillas para una primera instancia. Posteriormente se plantea hacer pruebas futuras en tiempo real con el vehículo en acción.

- **Test de reconocimiento diferentes distancias**

En este apartado se plantea evaluar los alcances mínimos y máximos de los dos algoritmos que se usan en el reconocedor de mascarillas. Se hizo la prueba en 5 distancias, empezando de la mínima distancia que puede reconocer el vehículo debido a la posición y ángulo de la cámara. La distancia mínima fue de 1.5m ya que a una distancia inferior, el rostro sale del ángulo de captura de la cámara. La máxima distancia fue determinada mediante prueba y error. La distancia máxima en la que se pudo reconocer un rostro fue de 3.5m. A partir de este rango, se estableció un intervalo de 50cm para realizar cada medición. Se realizó la prueba y se tomó datos para cada distancia del detector de rostros, mientras que para el detector de mascarillas se realizó pruebas para ambos casos: usando mascarilla y prescindiendo de ella. Resultados de este experimento se muestran en la tabla 3.

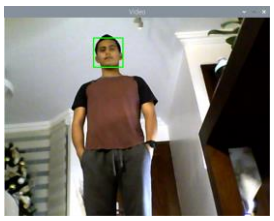
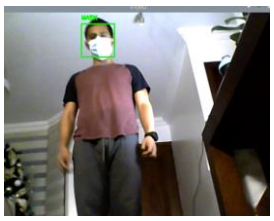
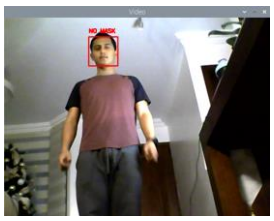
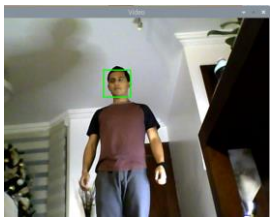
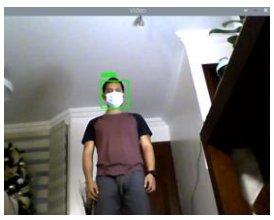
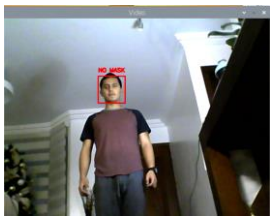
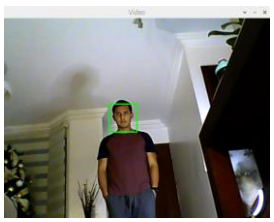
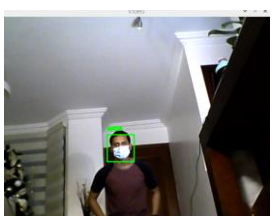
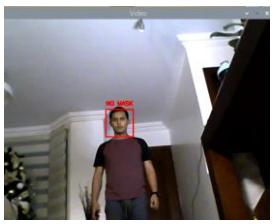
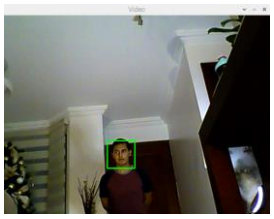
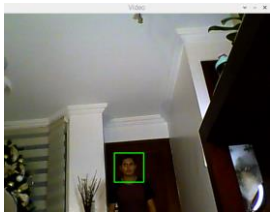
Modelo	Detector de rostro	Con mascarilla	Sin mascarilla
1.5 [m]			
2.0 [m]			
2.5 [m]			
3.0 [m]		-	-
3.5 [m]		-	-

Tabla 3. Resultados test de algoritmos de reconocimiento para diferentes distancias

- **Evaluación de confiabilidad de resultados en base a varias distancias**

Lo que se busca en este apartado es comprobar cómo afecta la distancia entre cámara y rostro a la confiabilidad del resultado que se muestra. Se realizó la prueba en intervalos de 50cm entre las distancias máximas y mínimas. Se tomó datos para cada distancia del detector de rostros, mientras que para el detector de mascarillas se la realizo con y sin la

misma. La confiabilidad marca qué tan confiable es el valor de la predicción que se hizo en el clasificador.

Modelo	**Con mascarilla** Confiabilidad (%)	**Sin mascarilla** Confiabilidad (%)
1.5 [m]	99.40	99.21
2.0 [m]	88.7	94.8
2.5 [m]	76.1	87.4
3.0 [m]	-	-
3.5 [m]	-	-

Tabla 4. Evaluación de confiabilidad para diferentes distancias

- **Evaluación de recursos de procesamiento por cada sistema**

  Se mide el porcentaje de CPU utilizado en cada núcleo de cada sistema

Raspberry 3 (sistema 1)	
Núcleo	**% utilizado**
1	91.0%
2	89.0%
3	90.0%
4	90.0%
Mem	534M/862M

Tabla 5. Porcentaje utilizado de CPU en cada núcleo sistema 1

Como se observa el Raspberry encargado del sensor LIDAR y la navegación ya tiene los recursos disponibles muy limitados, por lo que se optó por instalar un Raspberry adicional para poder trabajar la visión artificial con el mismo.

Raspberry 3b+ (sistema 2)	
Núcleo	**% utilizado**
1	50.3%
2	52.0%
3	50.6%
4	81.8%
Mem	350M/873M

Tabla 6. Porcentaje utilizado de CPU en cada núcleo sistema 2

Como observación de esta medición se puede concluir que se tiene alrededor de 50% disponible de CPU por lo que nos deja el campo abierto para poder trabajar en más proyectos de visión artificial con el vehículo autónomo.

Escalabilidad del proyecto y Planes de desarrollo.

Previamente el vehículo no tenía la posibilidad de tener inteligencia artificial logrando ser altamente escalable. Actualmente con la implementación de la inteligencia artificial mediante el segundo sistema embebido es posible escalarlo aún más. Teniendo este sistema encargado netamente del procesamiento de la inteligencia artificial del vehículo, se puede buscar y trabajar en el desarrollo de programas con modelos nuevos modelos de reconocimiento facial y de mascarillas más actualizados. De igual manera se puede trabajar en el desarrollo de sistemas con diferentes modelos enfocados a otro tipo de reconocimiento, como por ejemplo: reconocimiento de otros vehículos, señales de tráfico, vías, etc. Dentro del sector industrial sería posible reconocer señaléticas específicas como parte de un proceso, brindando al prototipo un mayor nivel de autonomía y haciendo uso de este nuevo sistema embebido.

Continuando con nuestro algoritmo actual también se puede escalar. Para obtener una mayor resolución se puede buscar nuevas cámaras web que se acoplen a los modelos usados. También se puede hacer el uso de una cámara de temperatura para que el vehículo detecte si la persona tiene una temperatura elevada y si esta porta o no mascarilla, posibilitando un mayor control de la enfermedad.

Como plan de desarrollo se pretende usar este vehículo autónomo con inteligencia artificial en el sector laboral como un sistema de vigilancia de bioseguridad automático, permitiendo que el empleador garantice que su personal tome las medidas de cuidado de

salud apropiadamente. De igual forma en el sector de la industria alimenticia se puede aplicar para garantizar las medidas de buenas practicas de manufactura que se emplea, estas son el correcto uso de mascarilla durante todo el proceso de la elaboración del producto.

Conclusiones

Haciendo uso de un prototipo de vehículo terrestre que puede moverse de manera manual y autónoma evitando las colisiones y proporcionando a un interfaz la posibilidad de realizar una reconstrucción del lugar (SLAM) y proporcionando visión del sitio con la cámara web se logró implementar un segundo sistema embebido dotando al vehículo de inteligencia artificial y creando una comunicación digital entre ambos sistemas para que dependiendo de la información de si una persona usa o no mascarilla facial, el robot pueda responder con ciertas acciones que ayuden a una implementación de esta importante medida sanitaria. El programa implementado permite obtener capturas de los rostros identificados sin mascarillas y llevarlos a una carpeta compartida para que se pueda hacer uso de esta información tanto ejecutando medidas correctivas, así como para dar soporte al desarrollo de nuevas políticas más eficaces.

El prototipo tiene autonomía promedio de 16 minutos debido a que se usó la misma batería para energizar ambos sistemas embebidos.

Bibliografía

Barbosa, J., & Hernández, C. (2020). Diseño e implementación de un vehículo autónomo con navegación basada en un sensor tipo LIDAR (Proyecto integrador). Universidad San Francisco de Quito, Quito, Ecuador.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb' s Journal of Software Tools*.

Cardona-López, A., & Pineda-Torres, F. (2019). Reconocimiento de rostros en tiempo real sobre dispositivos móviles de bajo costo. *Lámpsakos*, *1*(20), 30-39.

Darlington, K. (22 de Mayo de 2020). *OpenMind BBVA*. Obtenido de https://www.bbvaopenmind.com/tecnologia/inteligencia-artificial/esta-ayudando-la-inteligencia-artificial-contener-la-pandemia-covid-19/

Franco Quintero, J. A. (2020). Aplicación de la inteligencia artificial (IA) en imagen médica durante la crisis del Covid19: Un estudio de caso de Deep Learning como invención del Método de Invención.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.

Huili, Z. (2008). Realization of Files Sharing between Linux and Windows Based on Samba. In *2008 International Seminar on Future BioMedical Information Engineering* (pp. 418-420). IEEE.

Mujtaba, H. (26 de Julio de 2020). *greatlearning*. Obtenido de https://www.mygreatlearning.com/blog/real-time-face-detection/

OpenCV. (2015). *Open-Source Computer Vision Library*.

Ortego, D. (21 de Septiembre de 2017). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/que-es-ros/

Parreño Lara, M. (2019). Exploración de topologías para el reconocimiento de género.

salesforce. (22 de Junio de 2017). Obtenido de https://www.salesforce.com/mx/blog/2017/6/Que-es-la-inteligencia-artificial.html

salesforce. (2018 de Julio de 2018). Obtenido de https://www.salesforce.com/mx/blog/2018/7/Machine-Learning-y-Deep-Learning-aprende-las-diferencias.html

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

SANOFI, C. (09 de noviembre de 2020). Obtenido de https://campussanofi.es/e-professionals/noticias/inteligencia-artificial-covid-19/

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1*, I-511 — I-518. doi: 10.1109/CVPR.2001.990517.