Universidad Tecnológica Nacional

Proyecto Final

Sistema de seguridad y control del automóvil mediante Reconocimiento Facial - SecAR

Autores:

- Aquino Gastón Andrés
- Fálico Julio Ezequiel

Proyecto final presentado para cumplimentar los requisitos académicos para acceder al título de Ingeniero en Electrónica

en la

Facultad Regional Paraná

Marzo de 2020

Declaración de autoría:

Nosotros declaramos que el Proyecto Final “Sistema de seguridad y control del automóvil mediante Reconocimiento Facial - SecAR” y el trabajo realizado son propios. Declaramos:

- Este trabajo fue realizado en su totalidad, o principalmente, para acceder al título de grado de Ingeniero en Electrónica, en la Universidad Tecnológica Nacional, Regional Paraná.
- Se establece claramente que el desarrollo realizado y el informe que lo acompaña no han sido previamente utilizados para acceder a otro título de grado o pre-grado.
- Siempre que se ha utilizado trabajo de otros autores, el mismo ha sido correctamente citado. El resto del trabajo es de autoría propia.
- Se ha indicado y agradecido correctamente a todos aquellos que han colaborado con el presente trabajo.
- Cuando el trabajo forma parte de un trabajo de mayores dimensiones donde han participado otras personas, se ha indicado claramente el alcance del trabajo realizado.

Firmas:

-

-

Fecha:

Agradecimientos:

Queremos agradecer a la institución por la formación, a los docentes de la carrera por su constante disposición y ayuda ante nuestras consultas y a nuestras familias y seres queridos por el apoyo incondicional.

Aquino Gastón Andrés

Fálico Julio Ezequiel

Universidad Tecnológica Nacional

Abstract

Facultad Regional Paraná

Ingeniero en Electrónica

Sistema de seguridad y control del automóvil mediante Reconocimiento Facial - SecAR

Aquino Gastón Andrés

Fálico Julio Ezequiel

Abstract:

A safety system applied to vehicles was implemented, which drivers are analyzed by facial recognition and then checked in a database that is a user enabled to start the car. Also, once enabled, the designed device tracks the frequency of blinking of the eyes to detect signs of drowsiness and prevent accidents by issuing sound alerts. All this, with the ability to communicate via the Internet and make requests for various information, through the use of the Telegram app.

The project uses a Raspberry Pi for real-time image processing, and an infrared camera for taking these. In addition, it has a GPS module to obtain information regarding the location of the vehicle, and through the connection of a 3G USB modem to be able to send this data to the user in charge of taking a control or to who requests this information.

Finally, a simple and effective system was obtained to recognize people and detect drowsiness, and obtain a record of use. All this without the need for major modifications to the electrical / mechanical installation of the vehicle.

Keywords: Facial recognition, image processing, drowsiness, flickering, communication, control, security.

Resumen:

Se implementó un sistema de seguridad aplicado a vehículos, mediante el cual los conductores son analizados por un reconocimiento facial y luego se comprueba en una base de datos que es un usuario habilitado a realizar el encendido del automóvil. Así mismo, una vez habilitado, el dispositivo diseñado realiza un seguimiento de la frecuencia de parpadeo de los ojos para detectar signos de somnolencia y prevenir accidentes emitiendo alertas sonoras. Todo esto, con la capacidad de poder comunicarse vía internet y realizar peticiones de diversa información, mediante el uso de la aplicación Telegram.

El proyecto utiliza una Raspberry Pi para el procesamiento de imágenes en tiempo real, y una cámara infrarroja para la toma de estas. Además, cuenta con un módulo GPS para obtener información referente a la ubicación del vehículo, y mediante la conexión de un modem 3G USB poder enviar estos datos al usuario encargado de llevar un control o a quien solicite esta información.

Finalmente, se obtuvo un sistema sencillo y eficaz para reconocer personas, detectar somnolencia y obtener un registro de uso. Todo esto sin necesidad de realizar grandes modificaciones sobre la instalación eléctrica/mecánica del vehículo.

Palabras Clave: Reconocimiento facial, procesamiento de imagen, somnolencia, parpadeo, comunicación, control, seguridad.

Reconocimientos:

Queremos reconocer en primer lugar al Ingeniero Joaquín Neuman que nos brindó mucha información en cuanto a mecánica en general y electricidad del automóvil, al Ingeniero Gustavo Maggiolo por su ayuda, y al Ingeniero Alejandro Dachary por habernos presentado una idea general del presente proyecto.

Índice:

Lista de Figuras:

Lista de Tablas

Lista de Abreviaciones

OMS	Organización mundial de la salud
SSN	Superintendencia de seguros de la nación
CESVI	Centro de Experimentación y Seguridad Vial
GPS	Global positioning system
USB	Universal serial bus
GNU	Sistema operativo de tipo Unix
HLBP	Histogramas de patrones binarios locales.
HDMI	High-Definition Multimedia Interface
CSI	Interfaz Serie para Cámaras
NMEA	National Marine Electronics Association
CPU	Unidad central de proceso
CSI	Interfaz serie para cámaras
LED	Diodo emisor de luz
GPIO	Entrada/Salida para propósito general
EAR	Relación de aspecto del ojo
2D	Dos dimensiones
3D	Tres dimensiones
PCB	Placa de circuito impreso
CI	Circuito integrado

Lista de Diagramas

Lista de Ecuaciones

Dedicado a:

A nuestras familias, amigos, y seres queridos que nos apoyaron todo este tiempo y nos brindaron de su incondicional afecto.

Capítulo 1: Introducción

1.1 Fundamentación

La problemática que queremos abordar es el robo de automóviles en la República Argentina dado que según las estadísticas presentadas por la Superintendencia de Seguros de la Nación (SSN), en el año 2018 se denunciaron más de 60 mil casos de robo de vehículos, que promediando serían unos 167 episodios por día, o uno cada 9 minutos. Además, de acuerdo con estadísticas del Centro de Experimentación y Seguridad Vial (CESVI), el 32% de los robos son realizados a mano armada, situación que puede ser muy adversa emocionalmente y de un factor decisivo cuando se trata de la vida de las personas.

Este proyecto fue pensado para ser utilizado como un complemento de seguridad dentro del vehículo, con el objetivo de detectar si el conductor está autorizado a conducir, ya sea él mismo o un grupo de personas habilitadas, y una vez detectado el rostro proceder al encendido manual, caso contrario, no podrá ser encendido. De esta forma, se obtiene un método sencillo y confiable de asegurar un vehículo frente a conductores no designados, enfrentar las situaciones de robo simplemente dando las llaves del auto, o llevar estrictamente un registro de uso.

Según la Organización Mundial de la Salud (OMS), más de 1.3 millones de personas mueren en accidentes de tráfico cada año, con el 94% de estos accidentes fatales causados por errores humanos. Entre 20 y 50 millones de personas sufren traumatismos no mortales, y muchos de esos traumatismos provocan una discapacidad. Uno de estos errores humanos puede ser la pérdida del control del automóvil debido a somnolencia. La somnolencia es un estado de letargo, donde la persona presenta pesadez o actos involuntarios causados por una presencia profunda del sueño, pudiendo llegar a quedarse dormido en situaciones o momentos inapropiados. Este tipo de errores pueden llegar a ser mortales para la persona en cuestión, e incluso a terceros, debido a la posibilidad de incluir en el accidente a otros automovilistas.

El sistema que implementamos, además, ofrece la posibilidad de detectar un posible estado de somnolencia a los conductores, mediante un análisis de la frecuencia del parpadeo, emitiendo alarmas cada vez que las detecte para que detenga la marcha del vehículo. Como complemento, también se diseñó que haya comunicación entre el sistema y los usuarios mediante la conexión de un modem 3G USB, de forma tal que se pueda saber la ubicación del automóvil, que personas han estado utilizando el vehículo con fecha y hora, y el pedido de fotos del interior del habitáculo. Esta comunicación puede ser opcional para cada usuario ya que, si no lo desea, simplemente no se conecta ningún modem y trabaja normalmente con el reconocimiento y detección de somnolencia.

1.2 Sistemas de seguridad actuales

En la actualidad no hay muchos vehículos que cuenten con sistemas de seguridad basados en el análisis facial de los conductores, pero aun así muchas empresas se encuentran en investigación para llevar a sus autos este tipo de tecnología.
Entre los vehículos que hoy en día ya cuentan con sistemas similares se encuentra el M-Byte, fabricado por la empresa china de automóviles eléctricos Byton. Al subir una persona al vehículo, se la identifica y se carga su perfil personal, adaptando todas las funciones a sus necesidades, como ser los asientos, la temperatura y preferencias de la pantalla principal.

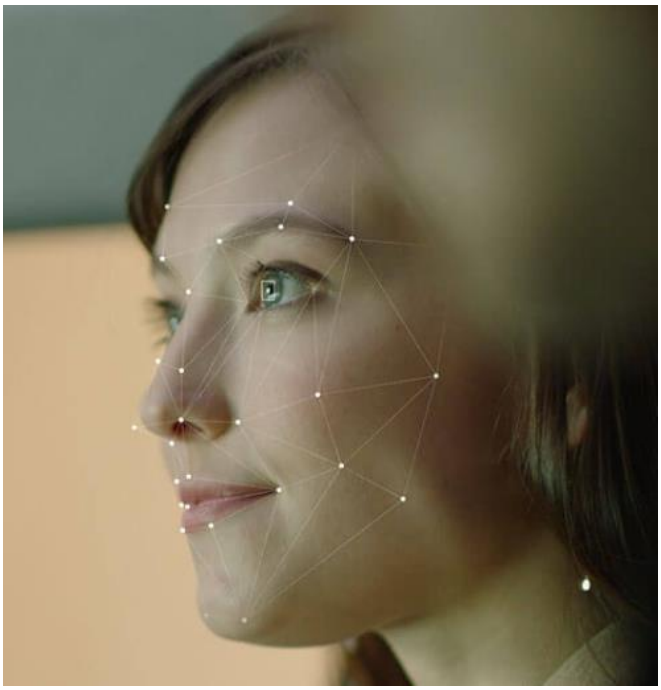

Figura 1 – Experiencia en el reconocimiento del Byton M-Byte [1]

Desde Subaru, el fabricante japonés, también encontramos un vehículo equipado con un sistema de reconocimiento facial. El Subaru Forester cuenta con el sistema de monitoreo del conductor – Driver Focus, que vigila al conductor atentamente realizando un seguimiento de la actividad ocular para calcular su nivel de somnolencia, emitiendo alarmas sonoras si detecta un caso extremo de dicho estado. A su vez, si el sistema detecta una falta de atención a la carretera, o si no está mirando al frente por un determinado tiempo, se emitirán alarmas sonoras para que la persona vuelva a enfocarse en el camino.

---

1 https://www.byton.com/m-byte/experience

El Subaru Forester también cuenta con un registro mediante el reconocimiento de rostros de conductores. El sistema es capaz de almacenar hasta 5 conductores con sus preferencias en la conducción, como ser la posición del asiento del conductor, ajustes climáticos, y configuraciones de pantalla.

Figura 2 – Sistema DMS del Subaru Forester [2]

Del fabricante estadounidense Cadillac se destaca el modelo CT6, que cuenta con Super Cruise, un asistente de conducción que es capaz de operar el vehículo con manos libres, siempre y cuando el conductor preste atención al camino y no use dispositivos de mano que puedan generar distracciones.

Para ello, el sistema realiza un seguimiento de la cabeza para asegurar que los ojos estén puestos sobre la carretera. En el caso de que el conductor no esté prestando atención suficiente al camino, se emitirán alarmas visuales y sonoras según sea el nivel de distracción, a mayor tiempo, mayor será el nivel de alarmas emitidas.
En el primer nivel de distracción, la barra de luces del volante parpadea en verde para indicar que el conductor vuelva a prestar atención a la carretera.

[2] https://www.subaru.com.au/why-subaru/technology

Figura 3 - Primer Alerta del sistema Super Cruise [3]

Si la barra de luces del volante parpadea en verde demasiado tiempo y el sistema determina que el conductor sigue sin ubicar la vista sobre el camino, la barra de luces del volante parpadea en rojo para notificarle que mire a la carretera y conduzca el vehículo manualmente.

Figura 4 – Segunda Alerta del sistema Super Cruise [4]

[3, 4] https://www.cadillac.com/ownership/vehicle-technology/super-cruise

En el caso de que la barra de luces del volante haya parpadeado en rojo durante demasiado tiempo, se escuchará un mensaje de voz de alerta. El conductor debe hacerse cargo de la dirección de inmediato y caso contrario, como último recurso, el vehículo irá disminuyendo la velocidad en su carril de circulación hasta detenerse por completo.

Figura 5 – Tercera Alerta del sistema Super Cruise [5]

1.3 Metodología de reconocimiento facial

El proceso de realizar un reconocimiento facial se puede dividir en diversas etapas. Se extraen imágenes de la persona no identificada desde una cámara y se analizan sus características faciales, para luego compararlas con una base de datos y determinar la identificación de la persona. El proceso general puede detallarse en el siguiente diagrama en bloques.

[5] https://www.cadillac.com/ownership/vehicle-technology/super-cruise

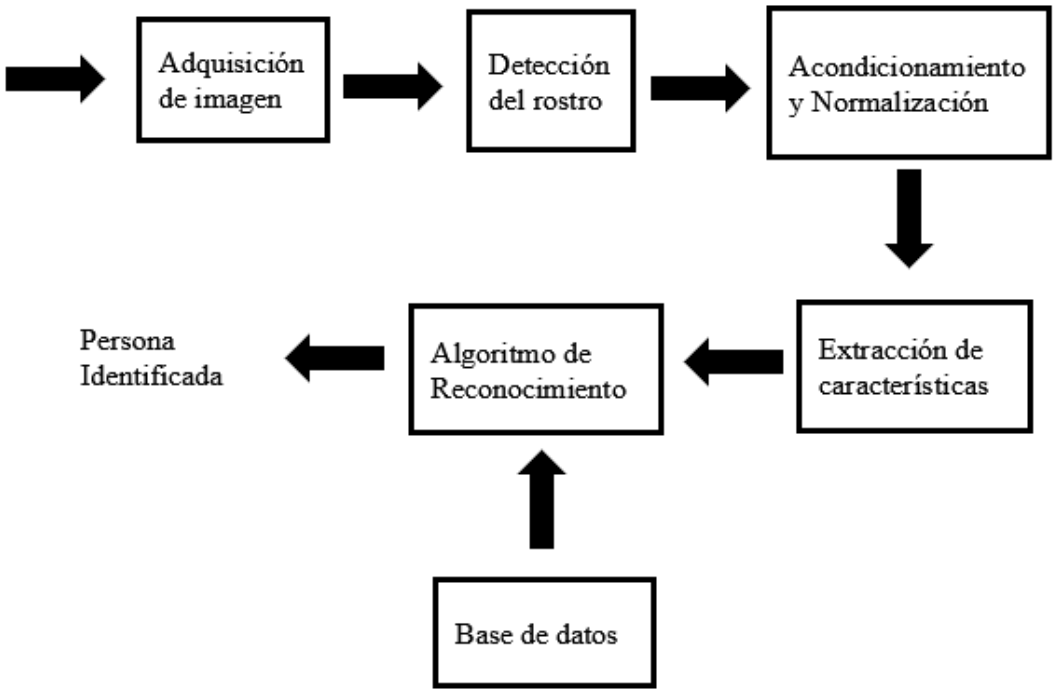

Diagrama 1 – Diagrama en bloques de realizar un reconocimiento

- Detección del rostro: localiza el rostro en la imagen

- Acondicionamiento y normalización: normaliza la imagen mediante diferentes métodos. Por ejemplo, transformando la imagen a una con escala de grises, reducción, ecualización.

- Extracción de características: dependiendo del algoritmo utilizado se proporciona información del rostro para distinguirlo de otros, por ejemplo, ojos, boca, rostro general, etc.

- Algoritmo de Reconocimiento: las características obtenidas previamente son comparadas con las que se posee en la base de datos. En caso de una similitud se obtiene el nombre de la persona en cuestión, o caso contrario, se informa de un desconocido.

Capítulo 2: Desarrollo

2.1 Software utilizado

2.1.1 Python

Es un lenguaje de programación interpretado de alto nivel, interactivo y orientado a objetos. El código fuente está disponible bajo Licencia Publica General de GNU. La sintaxis de Python es muy limpia, con énfasis en la legibilidad y utiliza palabras clave estándar en inglés.
Python fue creado a finales de la década de los años 80 por un europeo llamado Guido Van Rossum en los Países Bajos. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C.
Su formato no utiliza corchetes para delimitar bloques, sino que son las tabulaciones y los espacios los que definen en que bloque se encuentra una instrucción. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos '!', '||' y '&&' en Python se escriben 'not', 'or' y 'and', respectivamente.
Por defecto, Python viene instalado en el sistema operativo de Raspbian para Raspberry Pi.

Figura 6 – Logo de Python [6]

2.1.2 OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de aprendizaje por computadora y visión por computadora de código abierto. Al ser un producto con licencia BSD, OpenCV facilita a las empresas a utilizar y modificar el código. Lanzado oficialmente en 1999, el proyecto OpenCV fue inicialmente, una iniciativa de Intel Research.
La biblioteca posee más de 2000 algoritmos optimizados que se pueden utilizar para detectar y reconocer caras, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de cámara, etc.

---

6 https://brochure.getpython.info/

Es actualmente utilizado por grandes compañías como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota. Las distintas aplicaciones de OpenCV abarcan desde unir imágenes de streetview juntas, detectar intrusiones en videovigilancia, monitorear equipos mineros en China y mucho más.
OpenCV tiene interfaces C++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS. El software se inclina principalmente hacia aplicaciones de visión en tiempo real gracias a su eficiencia en recursos computacionales.

Figura 7 – Logo de OpenCV [7]

2.1.3 Dlib

Dlib es un kit de herramientas moderno de C++ que contiene algoritmos de aprendizaje automático y herramientas para crear software complejo en C++. Su uso se encuentra en la industria, la robótica, dispositivos integrados, teléfonos móviles y grandes entornos informáticos de alto rendimiento. La licencia de código abierto de Dlib permite usarla en cualquier aplicación de forma gratuita.
Desde que comenzó el desarrollo en el año 2002, Dlib ha crecido para incluir una amplia cantidad de herramientas. A partir del 2016, contiene componentes para manejar redes, interfaces gráficas de usuario, estructuras de datos, álgebra lineal, procesamiento de imágenes, minería de datos, y muchas otras tareas.

Figura 8 – Logo de Dlib [8]

Esta herramienta nos será muy útil en la detección de somnolencia, ya que ofrece interesantes funciones para determinar qué tan cerrados están los ojos y actúan en base a ello.

---

7 https://opencv.org/
8 http://dlib.net/

2.2 Clasificador de Haar para detección de objetos

La detección de objetos usando los clasificadores en cascada basados en características de Haar es un método de detección de objetos efectivo propuesto por Paul Viola y Michael Jones en su artículo, "Detección rápida de objetos usando una cascada mejorada de características simples" en 2001. Es un enfoque basado en el aprendizaje automático donde La función en cascada se forma a partir de muchas imágenes positivas y negativas. Luego se usa para detectar objetos en otras imágenes.

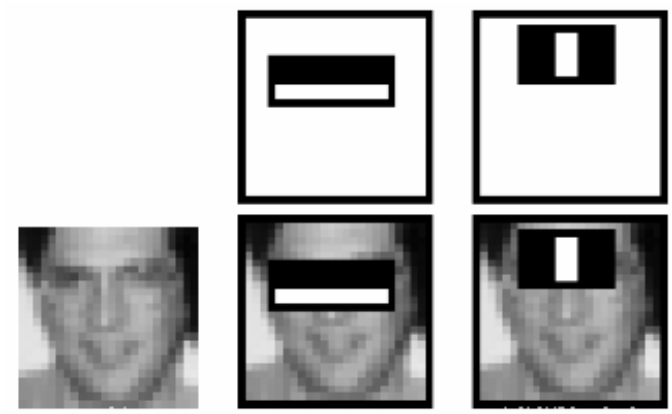

Figura 9 – Proceso de detección utilizando clasificadores de Haar [9]

Se trata de un clasificador basado en árboles de decisión para realizar el reconocimiento de un rostro en una imagen. El proceso consta de recorrer completamente la imagen tomando pequeñas ventanas de la imagen original, y en el caso en que una de esas ventanas no corresponda con regiones de un rostro se desecha y no vuelve a ser procesada, obteniendo así un método rápido y sencillo. Para esto, Viola-Jones utiliza la Cascada de Clasificadores que agrupa las características en diferentes etapas de los clasificadores y aplica una por una a la ventana. Si una ventana falla en la primera etapa se elimina y no se evalúan las características restantes en esa ventana. Si pasa, se aplica la segunda etapa de funciones y continúa el proceso. Finalmente, la ventana que pasa por todas las etapas es una región donde se encuentra un rostro.

Las primeras etapas de la cascada son simples y descartan rápidamente cualquier región que no presenten las características generales de un rostro humano, y luego las etapas siguientes son incrementalmente más selectivas y complejas, donde los recursos computacionales aumentan.

---

9 https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

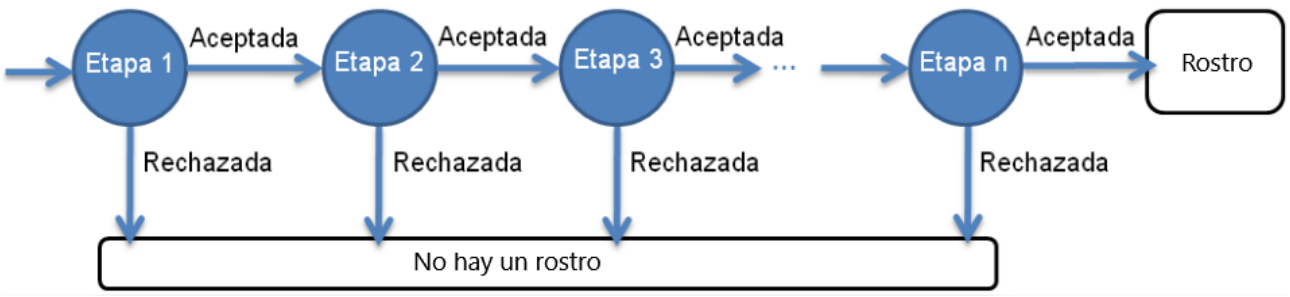

Figura 10 – Estructura en cascada del detector facial

En OpenCV se pueden usar diversos clasificadores para detectar diferentes regiones de un rostro. Cada uno de ellos dispone de un archivo específico, y el uso de uno u otro dependerá en gran medida de la aplicación que se busca. En nuestro caso, en base a experiencias y pruebas, se pudo ver que el detector de rostros Haar Rápido (haarcascade_frontalface_alt2) fue el que mejor resultados nos brindó, por lo que en este proyecto será el encargado de detectar a personas.

TIPO DE CLASIFICADOR	**NOMBRE ARCHIVO XML**
Detector de rostro	haarcascade_frontalface_default.xml
Detector de rostro (Haar rápido)	haarcascade_frontalface_alt2.xml
Detector de rostros de perfil	haarcascade_profileface.xml
Detector de ojos	haarcascade_lefteye_2splits.xml
Detector de boca	haarcascade_mcs_mouth.xml
Detector de nariz	haarcascade_mcs_nose.xml
Detector de persona completa	haarcascade_fullbody.xml

Tabla 1 – Tipo de clasificadores y sus respectivos archivos

2.3 Algoritmos de Reconocimiento

Para el reconocimiento e identificación de los rostros se utilizó la biblioteca de visión artificial de OpenCV, que contiene la clase FaceRecognizer. Los algoritmos disponibles actualmente son:

- Análisis de Componentes Principales (PCA), Eigenfaces
- Análisis de Discriminantes Lineales (LDA), Fisherfaces
- Histogramas de Patrones Binarios Locales. (HLBP)

Para nuestro proyecto, se prefirió utilizar el algoritmo de HLBP. Este método fue diseñado para la descripción de texturas. Se asigna una etiqueta a cada píxel de una imagen, el píxel central es comparado con sus pixeles vecinos dando como resultado un valor de su etiqueta. Después se usa los valores de las etiquetas de todos los pixeles para formar un histograma y este es utilizado como un descriptor de textura.

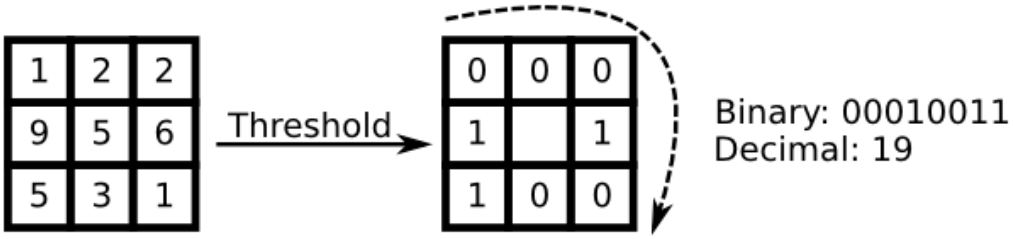

Figura 11 – Operador LBP [10]

Para obtener el valor del píxel central se realiza una simple comparación:

- Valor umbral píxel vecino ≥ Valor central píxel, se asigna un 1.
- Valor umbral píxel vecino < Valor central píxel, se asigna un 0.

Se transforma ese número binario a un número decimal y ese es el histograma o etiqueta para ese píxel central.
Para obtener un número de vecinos variables y establecer un radio cualquiera, se ubican los vecinos en forma circular y se aplica una interpolación bilineal para el caso de que algún píxel se encuentre fuera de la imagen. Así, utilizando la notación (P, R) se indica la cantidad de puntos vecinos, y el radio del circulo sobre que se ubican.

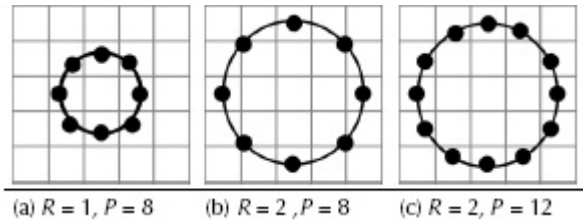

Figura 12 – Distintos radios y vecinos utilizando el operador LBP [11]

En la siguiente imagen puede observarse el resultado de haber aplicado el operador de Patrones Binarios Locales a un rostro donde se aprecia la textura final que se consigue.

10 https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

11 https://revistas.udistrital.edu.co/index.php/Tecnura/article/download/10086/11407?inline=1

Figura 13 – Resultado de haber aplicado el operador LBP [12]

[12] https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

2.4 Diagrama general del Proyecto

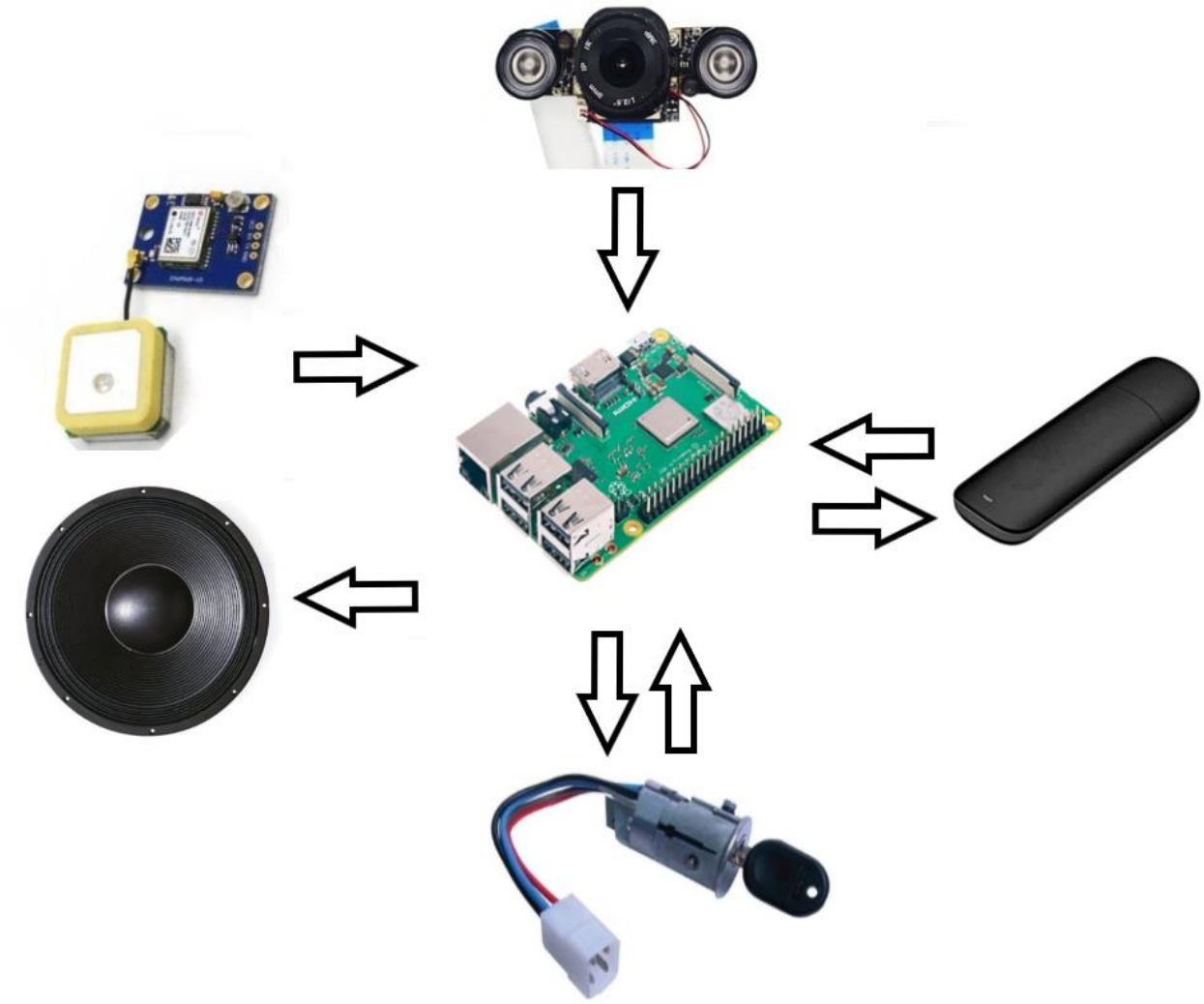

Figura 14 – Esquema de E/S del sistema

2.6 Descripción general

2.6.1 Raspberry Pi 3

Es una mini computadora de bajo coste y dimensiones, desarrollada en el Reino Unido por la Fundación Raspberry Pi. Como sistema operativo, la placa cuenta con Raspbian, que es una versión de Linux basada en Debian y especialmente desarrollada para Raspberry siendo uno de los más populares a la hora de usar este pequeño ordenador. Viene preinstalado con software educativo, para programación y para uso general. La placa utilizada para este proyecto es la versión 3 B, lanzada en febrero del año 2016.

Entre sus características podemos encontrar que posee un procesador de 64 bits Broadcom BCM2837 de 4 núcleos operando a 1.2 [GHz]. Dispone de 1 [GB] de RAM, conectividad LAN inalámbrica y Bluetooth de baja energía, 40 pines de propósito general (GPIO), 4 puertos USB, puerto HDMI, salida estéreo de 4 polos y puerto de video compuesto y puerto CSI para conectar una cámara Raspberry Pi.

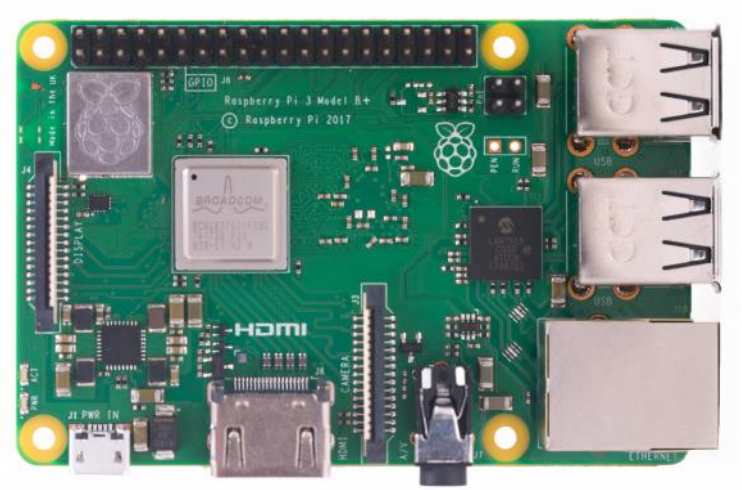

Figura 15 – Raspberry Pi 3 B [13]

2.6.2 Cámara IR

Es una cámara de 5 megapíxeles que utiliza el sensor OmniVision OV5647. Se conecta mediante un cable plano al puerto CSI de la Raspberry Pi. Posee dos LED infrarrojos de 3 [W] cada uno para otorgar una mayor capacidad de visión en ambientes pocos iluminados o bajo una oscuridad total. Además, presenta un lente focal variable.

---

13 https://in.element14.com/raspberry-pi/raspberrypi3-modb-1gb/sbc-raspberry-pi-3-mod-b-1gb-ram/dp/2525226

Figura 16 – Cámara infrarroja para Raspberry Pi

2.6.3 Módulo GPS

El Sistema de Posicionamiento Global (GPS) es un sistema de navegación por satélite formado por una red de 24 satélites colocados en órbita por el Departamento de Defensa de Estados Unidos. El GPS fue pensado originalmente para aplicaciones militares, pero en la década de 1980, el gobierno determinó que el sistema esté disponible para el uso civil. Funciona en cualquier condición meteorológica, en cualquier parte del mundo, las 24 horas del día. Los 24 satélites que conforman el segmento espacial del GPS orbitan la tierra cerca de 20.200 [Km] por encima de la superficie terrestre.

Figura 17 – Representación de los satélites orbitando la Tierra [14]

14 https://www.aristasur.com/contenido/como-funciona-el-sistema-de-posicionamiento-gps

Los satélites GPS circundan la tierra dos veces al día en una órbita muy precisa y transmiten información de la señal a la tierra. Los receptores GPS toman esta información y utilizan la trilateración para calcular la ubicación exacta del usuario. La trilateración es un método matemático para determinar las posiciones relativas de objetos usando geometría de triángulos.
Un receptor GPS debe tener una señal de al menos 3 satélites para calcular una posición de 2D (latitud y longitud) y seguir el movimiento. Con cuatro o más satélites a la vista, el receptor puede determinar la posición 3D del usuario (latitud, longitud y altitud).

El protocolo bajo el que se envía la información es NMEA. Estas siglas vienen de National Marine Electronics Association, una asociación fundada en 1957 por un grupo de fabricantes de electrónica para obtener un sistema común de comunicación entre las diferentes marcas de electrónica naval. Poco a poco se fueron sumando todos los fabricantes a este estándar, entre estos están fabricantes de aparatos electrónicos marinos tales como sonares, anemómetros, girocompás, pilotos automáticos, y receptores GPS entre otros instrumentos.

La salida de los módulos GPS proporciona mensajes NMEA estándar, tales como:

- GGA: proporciona datos relacionados, como la posición, la hora, el número de satélites en uso, etc.
- GSA: proporciona el modo de funcionamiento del receptor, en general los datos de satélite activos.
- GSV: proporciona la información de los satélites observables, tales como números PRN, elevación, azimut, SNR, y el número de satélites a la vista.
- RMC: proporciona los datos mínimos para el GPS, como el tiempo, la posición, velocidad, el rumbo.
- VTG: proporciona la velocidad y el rumbo con respecto a tierra.
- GLL: proporciona la posición geográfica en latitud/longitud, los datos de navegación y el estado del satélite.
- ZDA: proporciona la hora UTC, la fecha y la zona horaria local.

El modelo adquirido y utilizado es GY-Neo6mv2:

Figura 18 – Modulo GPS

Cuenta con una antena de gran potencia, así como una memoria EEPROM para guardar datos y una batería para hacer el respaldo de la configuración del módulo. Su tensión de alimentación puede ser entre 3.3 [V] y 5 [V], consumiendo una corriente típica de 45 [mA]. La velocidad de comunicación predeterminada es de 9600 [bps]

2.6.4 Modem USB 3G

Un modem USB 3G es un producto de comunicaciones que se conecta al puerto USB de un equipo y recibe la señal de Internet a través de una red 3G. Se prefirió el uso de estos dispositivos debido a su simplicidad de configuración, acceso, costo y velocidad de internet que puede proveer.

Figura 19 – Modem 3G USB

2.6.5 Conversor DC-DC

La Raspberry Pi se debe alimentar con una fuente de tensión continua de 5 [V] y al menos 2 [A] de corriente, por lo que se adquirió un conversor Step-Down DC-DC de 15 [W] donde la entrada son los 12 [V] provenientes de la batería del automóvil. La tensión de entrada puede variar de 6.3 [V] a 22 [V], y la eficiencia de conversión ronda el 95%.
La temperatura de trabajo del conversor es de - 40 [ºC] a 85 [ºC].

Figura 20 – Conversor Step-Down DC-DC

Dado que a la salida el conversor es capaz de entregar una potencia de 15 [W], se pudo determinar que con ese valor alcanza para alimentar la placa de la Raspberry Pi, y demás elementos que son energizados por dicho dispositivo.

2.6.6 Corte del motor de arranque

El sistema será el encargado de proporcionar el arranque una vez que se reconoce a las personas autorizadas, por lo que se debe lograr un corte seguro del motor de arranque.
El corte se realiza en el cable de la llave de IGNICION, el cual se ingresa a un circuito basado en un relé que, en el momento que se autoriza el encendido, se conmuta el dispositivo proporcionando que vuelva haber una señal sobre el cable de IGNICION.

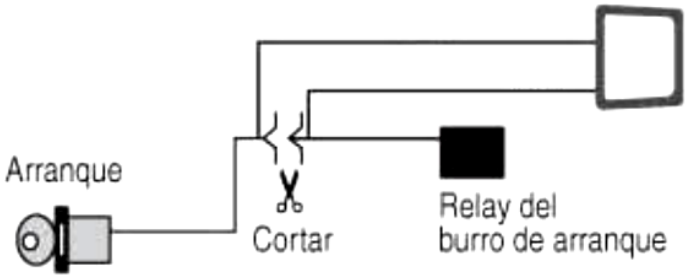

Figura 21 – Ubicación del corte a realizar

El motor de arranque es un motor eléctrico que se encarga de transformar la energía eléctrica en un giro de cigüeñal para que el motor pueda ponerse en marcha. El movimiento transmitido al cigüeñal se realiza mediante el conjunto de piñón que se encuentra en el eje de giro del motor eléctrico junto con la corona dentada situada en la parte exterior del volante de inercia.

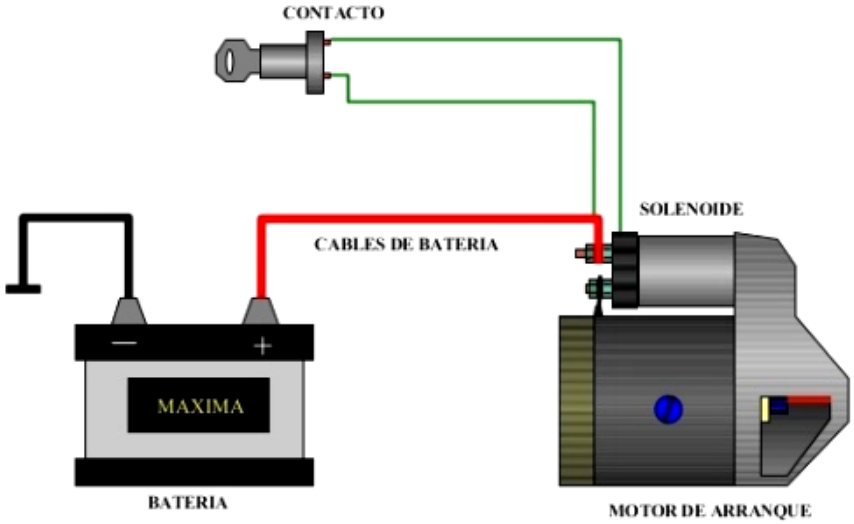

Figura 22 – Sistema de arranque mediante solenoide

Finalmente, hará falta un elemento que acople el piñón con la corona para hacer girar al cigüeñal y lo desacople cuando el motor se haya puesto en funcionamiento. Este elemento es el contactor o solenoide.
Actualmente los motores de arranque cuentan con un electroimán que funciona con corriente continua, y se alimentan de la batería. Del motor de arranque, un cable fino irá conectado desde la llave de contacto hasta el relé o solenoide funcionando como interruptor, mientras que un cable más grueso lo alimentará desde la batería una vez que se haya accionado el relé. Cuando giramos la llave de contacto, la corriente de la batería pasa al relé de arranque, produciendo un efecto de palanca sobre el piñón de arrastre del motor de arranque que permite que se acople al engranaje de la corona del volante del motor térmico para proporcionarle movimiento, por tanto, su función es vencer la resistencia inicial de los componentes del motor al arrancar.

2.7 Software implementado

2.7.1 Captura de video

En este punto nos enfocaremos en la captura de imágenes proveniente de la cámara para utilizarlas tanto para reconocer personas y luego en detector de somnolencia, como así también, para agregar personas a la base de datos del sistema.
Para nuestro propósito, el código implementado hace uso de la librería ‘imutils’, que inicializa el flujo de imágenes desde la cámara de Raspberry Pi, con una resolución fija de 720 x 560 pixeles. Luego la función read() devuelve el frame obtenido en la variable img.

```
from imutils.video import VideoStream
cam = VideoStream(resolution=(720,560), framerate=21, usePiCamera=True).start()
img = cam.read()
```

Figura 23 – Captura de imagen

Figura 24 – Captura de imagen

2.7.2 Detección de rostros

La detección de rostros se realiza, como bien se mencionó antes, mediante la utilidad de OpenCV. El código implementado utiliza el detector “frontalface_alt2”. Un punto para destacar es la obligación de utilizar frames en escala de grises para realizar la detección con el clasificador de Haar.

La función específica utilizada es la siguiente, y posee los siguientes argumentos:

detectMultiScale (Mat imagen, double factor escala, int vecinos cercanos, int banderas, Size tamaño mínimo, Size tamaño máximo)

Donde:

- Imagen: Imagen en escala de grises donde se desea detectar los objetos.
- Objeto: vector de rectángulos donde cada rectángulo contiene el objeto detectado.
- Factor escala: este parámetro determina el tamaño de los rostros a buscar en la imagen, donde valores típicos son 1.1 o 1.2. A menor valores, más precisa la detección pero será más lenta.
- Vecinos cercanos: este parámetro determina la fiabilidad de un rostro detectado, normalmente un valor de 3 es utilizado como valor estándar.
- Tamaño mínimo: tamaño mínimo del rostro a buscar.
- Tamaño máximo: tamaño máximo del rostro a buscar.

```
import cv2
# Definimos el tamaño mínimo a reconocer una cara
minW = 80
minH = 80
fn_haar = '/home/pi/recoFacial/haarcascade_frontalface_alt2.xml'
detector = cv2.CascadeClassifier(fn_haar)
faces = detector.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 6,
        minSize = (minW,minH))
```

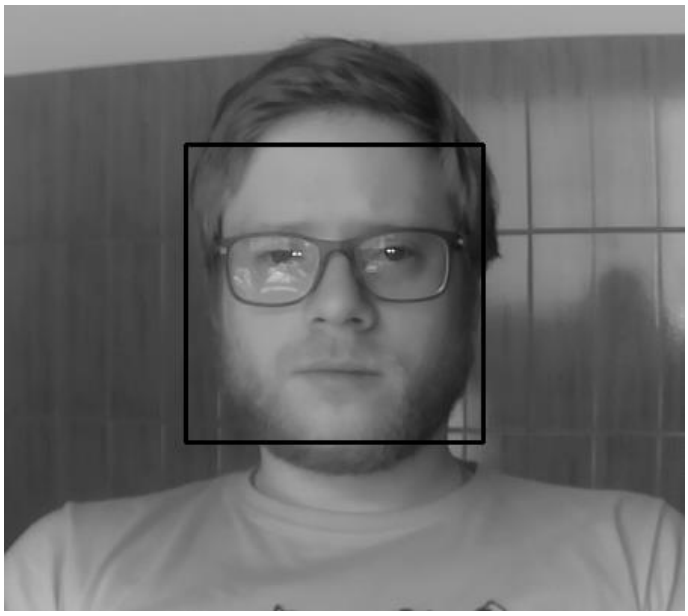

Figura 25 – Detección de un rostro

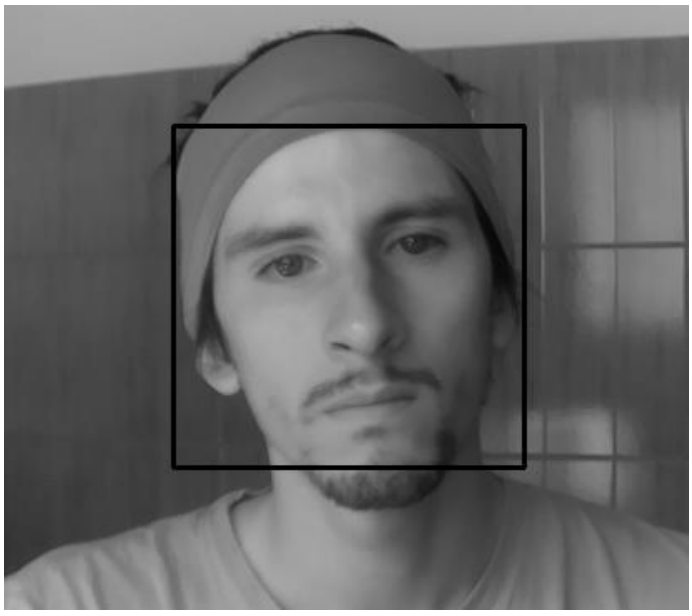

Figura 26 – Detección de un rostro

2.7.3 Procesamiento de las imágenes

Una vez detectado los rostros, es necesario procesar las imágenes obtenidas para llevarlas a un formato adecuado para el reconocimiento de personas. Lo primero que se realiza es llevar el frame original obtenido de la cámara a una imagen en escala de grises, luego se realiza una ecualización para otorgar mayor contraste. Finalmente se redimensionan a un tamaño de 144 x 168 pixeles.
El objetivo de ecualizar las imágenes es llevar los píxeles que estaban agrupados sobre un rango pequeño de intensidades, a un rango mayor de intensidad para mejorar el contraste.

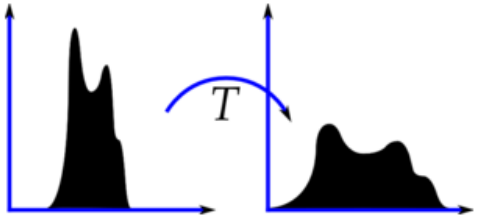

Figura 27 – Proceso de ecualización [15]

Por ejemplo, tomemos la siguiente figura:

Figura 28 – Figura de muestra para ecualizar [16]

Si aplicamos la ecualización de histogramas a la imagen anterior se obtiene:

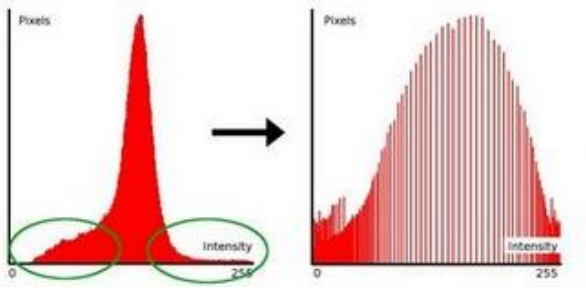

Figura 29 – Rango de intensidades [17]

---

[15] https://docs.opencv.org/3.4/d5/daf/tutorial_py_histogram_equalization.html

[16], [17] https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

Se puede ver como la primera imagen presenta un rango muy pequeño de intensidades, y en sus extremos se nota una gran cantidad de píxeles de muy baja intensidad (círculos verdes). Luego de la ecualización, se extiende dicho rango, como se aprecia en la imagen de la derecha, y la figura resultante es:

Figura 30 – Imagen resultante luego de aplicar la ecualización [18]

Así, volviendo a nuestro proyecto, las imágenes que obtuvimos mediante este proceso fueron:

Figura 31 – Imagen en escala de grises ecualizada

18 https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

Figura 32 – Imagen en escala de grises ecualizada

Para el caso del algoritmo donde se agregan personas a la base de datos, antes de redimensionarlas, se les aplica una máscara elíptica para borrar todo el contenido que esté por fuera del rostro detectado, y no genere información basura. Con esto realizado, se procede a guardar los rostros en una carpeta específica para la base de datos, y en un formato ".pgm". Para un buen reconocimiento, se encontró experimentalmente que son necesarias unas 200 fotos por persona, en diferentes distancias y condiciones lumínicas.

A su vez, si la persona utiliza anteojos de forma habitual, es necesario una serie de fotos del individuo donde se le captura el rostro con y sin lentes, dando un promedio de 500 fotos para este caso.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
equal = cv2.createCLAHE()
equ = equal.apply(gray)

mask = np.zeros_like(equ)
result = np.bitwise_and(equ,mask)

res = cv2.resize(result[y:y+h,x:x+w],(144,168))
cv2.imwrite("dataset/Usuario." + str(face_id) + '.' + str(count) + ".pgm", res)
```

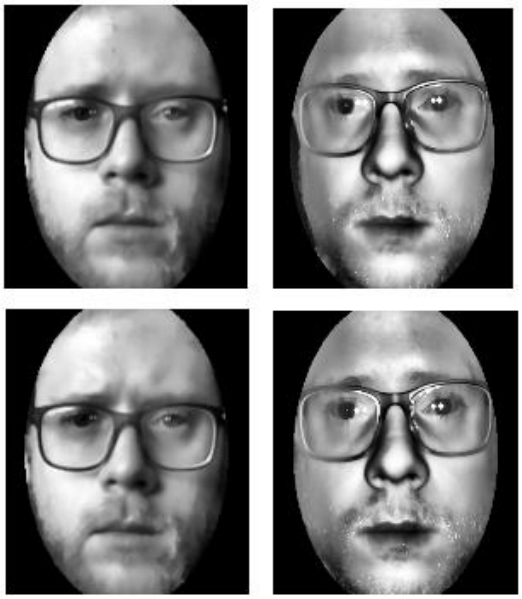

Figura 33 – Rostro ecualizado, recortado y con máscara elíptica

Figura 34 - Rostro ecualizado, recortado y con máscara elíptica

Aquí, nuevamente, el objetivo de realizar una máscara elíptica alrededor del rostro es eliminar parte del cabello, accesorios, o cualquier tipo de información basura que no ayude a entrenar la base de datos.

2.7.4 Diagrama de flujo de agregar personas

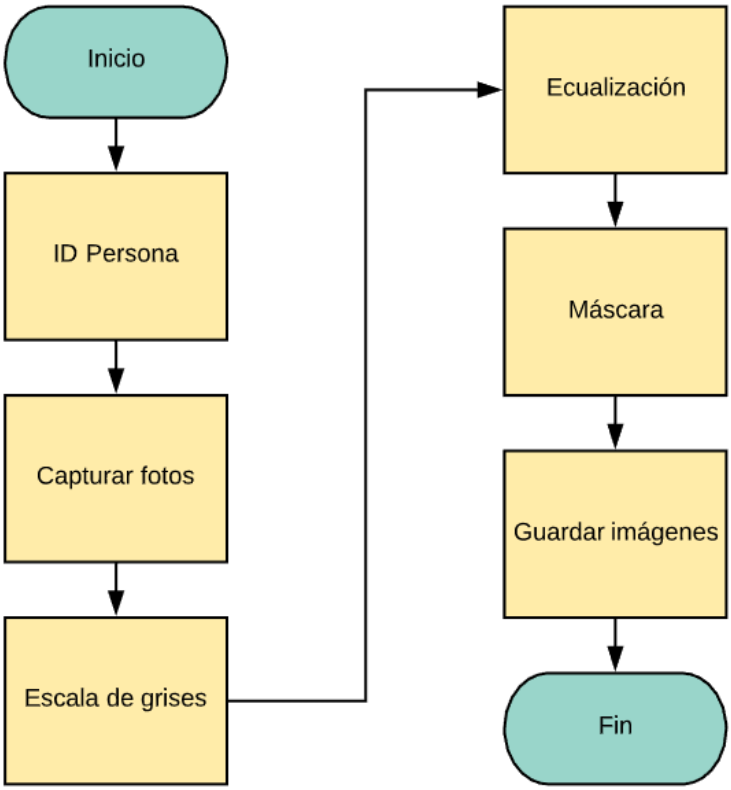

Diagrama 2 – Diagrama del algoritmo para agregar personas

2.7.5 Entrenamiento de la base de datos

Esta función será la encargada de entrenar el algoritmo de OpenCV con los rostros almacenados de cada usuario, para luego utilizarlo en el reconocimiento. Mas adelante se explicará con más detalle cómo trabaja la clase createLBPHFaceRecognizer(). Esto se hace directamente por una función específica de OpenCV. El resultado será un archivo de formato .yml que se guarda con el nombre "entrenador".

```
recognizer = cv2.face.createLBPHFaceRecognizer(2,8,8,8)

recognizer.train(faces,np.array(ids))
# Guarda el modelo "entrenador"
recognizer.save('/home/pi/recoFacial/entrenador.yml')
```

2.7.6 Diagrama de flujo del entrenamiento

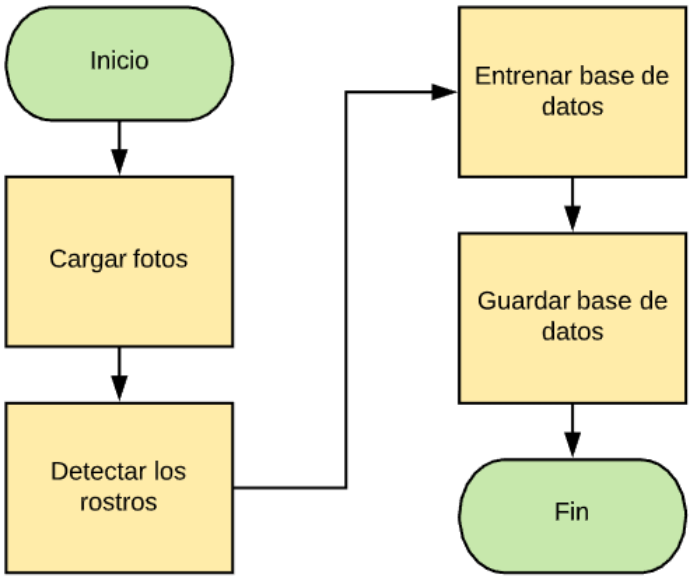

Diagrama 3 – Diagrama en bloques para entrenar la base de datos

2.7.7 Algoritmo de reconocimiento

El algoritmo para el reconocimiento facial utilizado es HLBP (histogramas de patrones binarios locales) como bien se mencionó anteriormente en el capítulo previo. OpenCV cuenta con la clase FaceRecognizer para implementar este algoritmo.

Esta clase cuenta con las funciones:

- Train: permite obtener un archivo donde se encuentran los valores de los histogramas de las imágenes en la base de datos.
- Update: permite actualizar el archivo con la información de los histogramas con nuevas imágenes.
- Predict: permite comparar el histograma de una nueva imagen con el archivo de los histogramas almacenado, indica cual es la imagen reconocida y entrega el valor de confianza de la predicción.
- Write/Save: guarda el archivo en formato YAML o XML.
- Read/Load: carga el archivo, para tenerlo como referencia en la predicción.
- Umbral: permite establecer un valor base para la predicción.

La utilización de Write o Save, como así también, de Read o Load depende de la versión instalada de OpenCV, ya que para versiones antiguas se usaban Write y Read. En versiones modernas se modificaron y pasaron a ser Save y Load.

El algoritmo que presenta OpenCV en la clase FaceRecognizer es el siguiente:

```
Ptr<FaceRecognizer> modelo =createLBPHFaceRecognizer (radio, vecinos, x, y);
```

Donde:

- Radio: es el radio del círculo para la construcción de LBP.
- Vecinos: número de muestras alrededor del círculo.
- X: regiones a dividir en el eje x.
- Y: regiones a dividir en el eje y.

En nuestro software implementado, los valores utilizados para cada argumento fueron: radio = 2, vecinos = 8, x = 8, y = 8. Esto significa que la imagen de 144x168 pixeles fue dividida en 64 regiones, para determinar el histograma de cada píxel utilizando un radio de círculo igual 2, utilizando 8 muestras alrededor.

Cuando se compara un rostro que ha sido captado con el archivo entrenado, se utiliza la función predict(), esta función luego de haber hecho las operaciones respectivas entrega la identificación del rostro y un valor de confianza de dicha identificación.
La identificación entregada es el número asignado cuando se entrenó las imágenes, por ejemplo, si se entrena los rostros de ‘usuario1’ con el número 1, y si la imagen que se va a comparar es la del usuario1, esta función entregará un 1 y un valor de confianza, este valor de confianza es mayor cuando se acerca a cero, cuando es una imagen que no ha sido reconocida la función entrega -1.

Dado que este valor de confianza en la práctica no suele ser muy amigable para trabajar, se lo convierte a un valor en porcentaje de semejanza, utilizando la ecuación 1. Por lo que una predicción perfecta arrojaría un valor de 100%.

```
nombres = ['NN', 'Ezequiel', 'Gaston', 'Des'…]

recognizer = cv2.face.createLBPHFaceRecognizer(2,8,8,8)
id, confidence = recognizer.predict(res)

persona = nombres[id]
```

$$semejanza = 100 - confidence$$

Ecuación 1 – Conversión de confianza en porcentaje

En la variable persona, entonces, aparecerá el nombre del individuo en cuestión basado en la predicción, y en la variable semejanza una estimación de “que tan parecido” es el rostro analizado con la predicción dada.

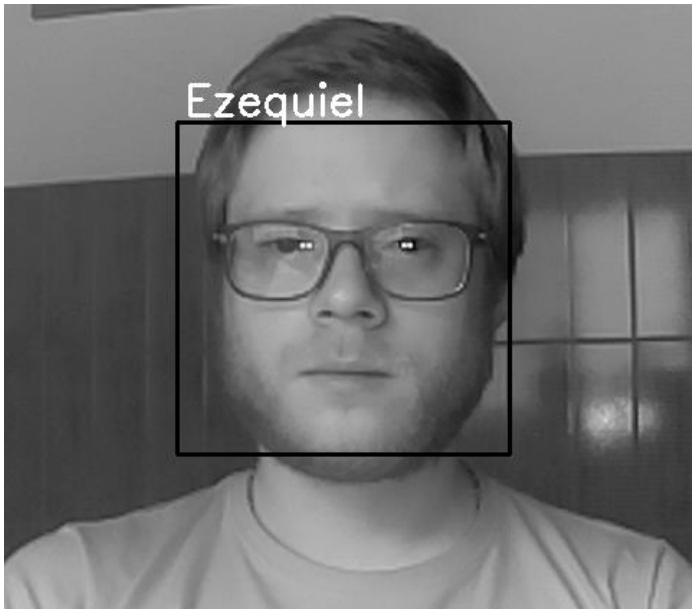

Figura 35 – Persona reconocida

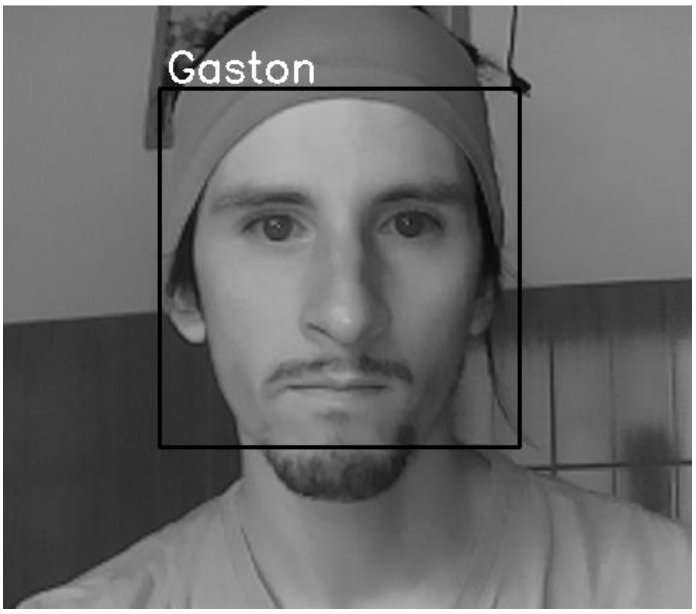

Figura 36 – Persona reconocida

Para evitar posibles falsos positivos, es decir, que haya una predicción errónea de una persona, el sistema activará el encendido cuando haya sido posible detectar 3 veces el mismo rostro de una persona autorizada. Además de esto, se emite una alarma de bienvenida y se actualiza el registro de conductores con la fecha y hora del reconocimiento.

```
If contadorp == 3:
        os.system('mpg321 /home/pi/recoFacial/bienven.mp3')
        cv2.imwrite("captura.png", img)
        archivo = open("/home/pi/recoFacial/lista.txt", "a")
        archivo.write(id + time.strftime("      \t\t%d/%m/%y  %H:%M:%S\n"))
        archivo.close()
        break
```

2.7.8 Diagrama de flujo de reconocer personas

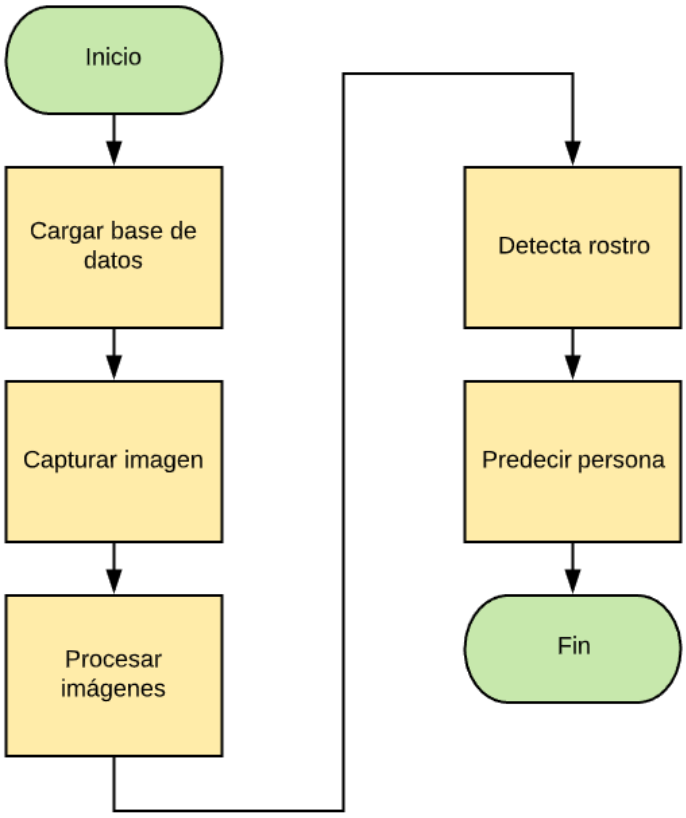

Diagrama 4 – Diagrama en bloques de reconocer personas

2.7.9 Alertas y pedidos por Telegram

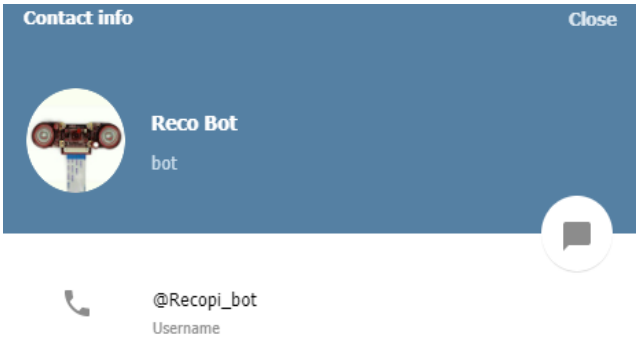

Figura 37 – Perfil del Bot creado en Telegram

El sistema implementado tiene la capacidad de comunicarse con el usuario final mediante la app de Telegram. Es así como, en caso de que se detecte síntomas de somnolencia el sistema avisa instantáneamente al supervisor del vehículo de la situación. Además, se implementó una serie de comandos que el dispositivo puede entender y ejecutar determinadas acciones. Estos comandos son:

- /foto – Pedir una foto instantánea
- /lista – Pedir una lista de conductores autorizados con fecha y hora
- /mapa – Pedir la ubicación del vehículo mediante GPS

La lista enviada estará en formato '.txt' con lo cual puede ser visualizada con cualquier programa desde una computadora.

Figura 38 – Recepción de lista de conductores mediante Telegram Web

```
bot = telepot.Bot('ID Bot')
if command == '/foto':
        img = cam.read()
        res = cv2.resize(img,(144,168))
        cv2.imwrite("/home/pi/recoFacial/captura.png", res)
        try:
            bot.sendPhoto(chat_id, photo=open('/home/pi/recoFacial/captura.png', 'rb'))
        except:
            print("")
    if command == '/mapa':
        try:
            bot.sendMessage(chat_id, str(lat) + str("      ") + str(long))
        except:
            print("")
    if command == '/lista':
        try:
            bot.sendDocument(chat_id, document=open("home/pi/recoFacial/lista.txt"))
        except:
            print("")

MessageLoop(bot, handle).run_as_thread()
```

El mensaje de alerta cuando se detecta somnolencia se presenta a continuación:

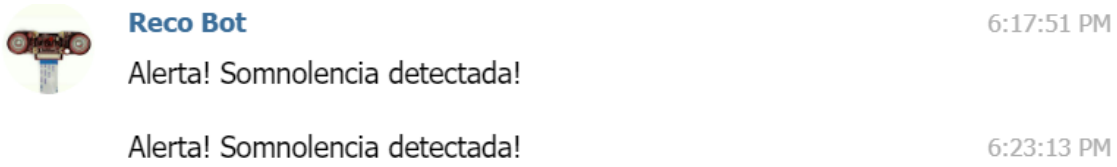

Figura 39 – Alertas enviadas por el Bot

Para que el sistema reconozca las órdenes que se le piden, se analiza el mensaje recibido y en caso de pertenecer a los comandos establecidos anteriormente, se realizan las acciones correspondientes. Cuando el comando recibido es "/foto" el programa captura una imagen de la cámara y la redimensiona a un tamaño menor, esto es para evitar mandar imágenes grandes al utilizar el modem 3G, permitiendo el ahorro de datos.

Figura 40 – Recepción de imagen mediante comando

2.7.10 GPS

La comunicación del módulo GPS con la Raspberry Pi es mediante puerto serie, por lo tanto, para recibir la información es necesario configurar el puerto y luego leerlo de forma reiterada. El mensaje NMEA que recibe el sistema será GGA que contiene la información de latitud y longitud, que luego con la librería "pynmea2" son extraídos para almacenarlos en sus respectivas variables.

```
Import pynmea2
import serial
ser = serial.Serial("/dev/ttyAMA0", 9600, timeout=0.5)
data = ser.readline()
data = data.decode("utf-8","ignore")
if data.find('GGA') > 0:
```

```
msg = pynmea2.parse(data)
lat = msg.latitude
long = msg.longitude
```

La ubicación GPS remitida posee las coordenadas de latitud y longitud, con lo cual fácilmente se copia y pega en el buscador y se obtiene el sitio aproximado donde se encuentra el vehiculo.

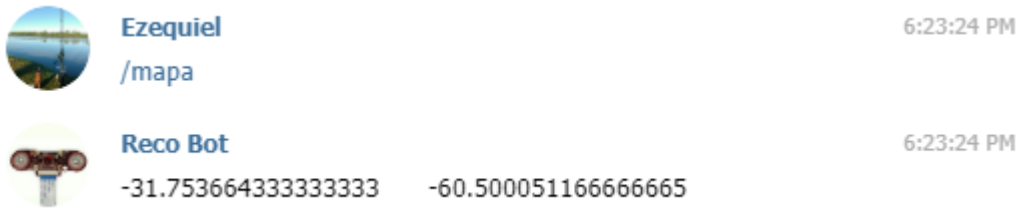

Figura 41 – Recepción de ubicación GPS mediante comando en Telegram Web

Figura 42 – Ubicación aproximada en Google Maps

2.7.11 Detección de somnolencia

La detección de somnolencia se realiza mediante la utilización de dos librerías: OpenCV que detecta los rostros, y Dlib que se encarga de obtener las partes relevantes de la cara, que en nuestro caso serán los ojos.

Como se puede apreciar en la siguiente imagen, Dlib otorga un conjunto de puntos que definen un rostro, por ejemplo, el ojo derecho estará definido por los puntos 37 al 42 y el ojo izquierdo por los puntos 43 al 48.

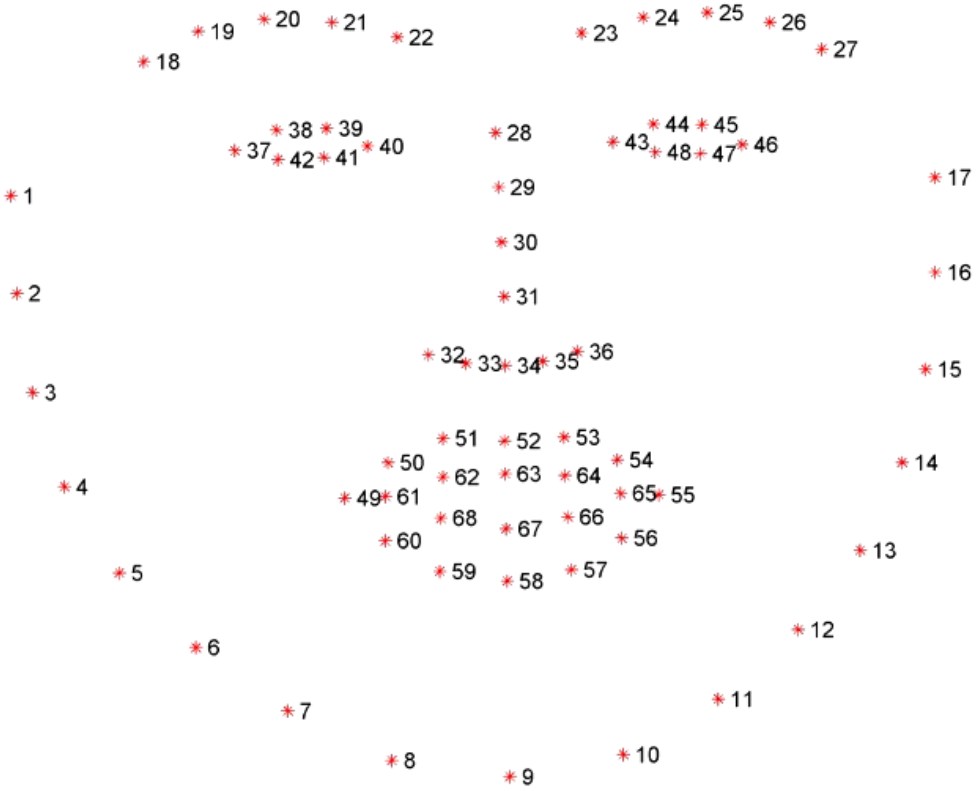

Figura 43 - Visualización de las 68 coordenadas faciales [19]

Para determinar el estado de somnolencia del conductor en este sistema, se utiliza el cálculo de una métrica llamada relación de aspecto del ojo (EAR). Este método hace el procesamiento mucho más simple ya que no implica una combinación de la localización ocular, utilización de un umbral para encontrar el blanco de los ojos y determinación de si la región blanca de los ojos desaparece por un período de tiempo lo que indica un parpadeo.

[19] https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

La implementación del código se basa en crear un objeto Dlib del rostro encontrado para determinar los puntos de referencia faciales en coordenadas X-Y, luego se convierten a un vector NumPy para su posterior calculo.

```
Rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
shape = predictor(img, rect)
shape = face_utils.shape_to_np(shape)
```

La relación de aspecto del ojo es, en cambio, una solución mucho más elegante que implica un cálculo muy simple basado en la relación de distancias entre los puntos de referencia faciales de los ojos.

$$EAR = \frac{\|p2 - p6\| + \|p3 - p5\|}{2\|p1 - p4\|}$$

Ecuación 2 – Cálculo de la relación de aspecto del ojo [20]

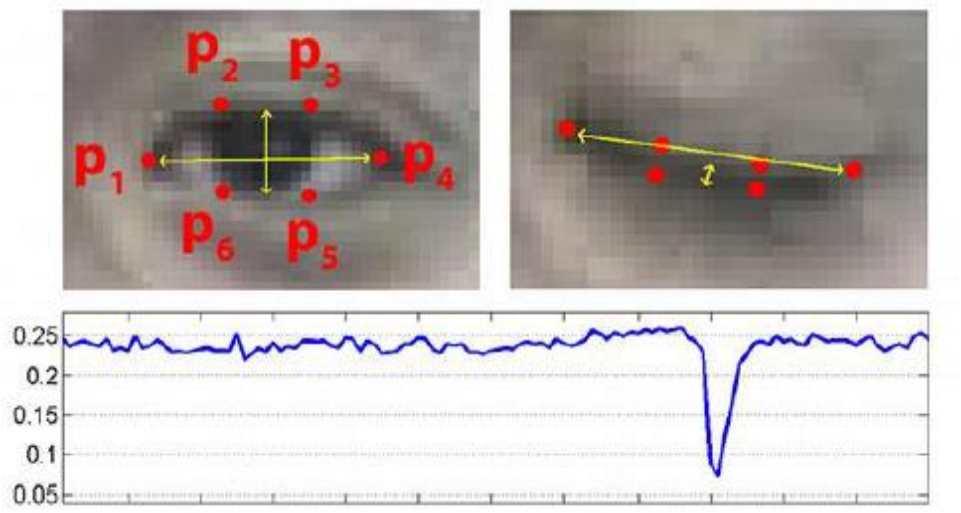

Figura 44 - Puntos de referencia del ojo cuando el ojo está abierto [21]

Una vez hecho esto, se extraen las coordenadas del ojo izquierdo y derecho y se realiza el cálculo de la relación de aspecto para ambos ojos.

```
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
```

[20], [17] https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

Con esto ya es posible calcular la relación de aspecto mediante la función eye_aspect_ratio(). Se calcula la distancia entre los dos puntos de referencia verticales del ojo y luego la distancia entre los puntos de referencia horizontal del mismo.
Por último, se combina tanto el numerador como el denominador para llegar a la relación de aspecto del ojo final.

```
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)

def eye_aspect_ratio(eye):
A = euclidean_dist(eye[1], eye[5])
B = euclidean_dist(eye[2], eye[4])
C = euclidean_dist(eye[0], eye[3])
ear = (A + B) / (2.0 * C)
return ear
```

Una vez calculada la relación de aspecto de ambos ojos, se promedia los dos valores para obtener una mejor estimación del parpadeo debido a que una persona realiza el parpadeo de ambos ojos simultáneamente.

Si la relación de aspecto del ojo cae por debajo de un cierto umbral y luego se eleva por encima del mismo, entonces el conductor realizo un parpadeo. La constante UMBRAL_RA es este umbral, su valor se debe ajustar realizando pruebas y hallando el valor que mejor funcione.
En caso de detección de somnolencia se activará la alarma que emitirá una señal de alerta sonora y enviando un aviso mediante Telegram. El audio reproducido es un archivo mp3 almacenado en el directorio del proyecto y es utilizado por el programa mpg321, un reproductor de mp3 de línea de comandos gratuito.

```
Ear = (leftEAR + rightEAR) / 2.0

if ear < EYE_AR_THRESH:
        counter += 1
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
            os.system('mpg321 /home/pi/recoFacial/cuidado.mp3')
            if ALARM_ON == 0:
                ALARM_ON = 1
                try:
                    bot.sendMessage(chat_id=9183773XX, text="Alerta! Somnolencia
                    detectada!")
                except:
                    print("")
```

Luego existe en este código otra constante importante de destacar, FRAMES_CONSEC. Este parámetro se establece en 11, para indicar que deben ocurrir 11 cuadros sucesivos con una relación de aspecto de ojo inferior a EYE_AR_THRESH para considerar que el conductor se encuentra en un estado de somnolencia. Es importante destacar que el tiempo en que ocurran estos frames o cuadros, va a depender de la velocidad de procesamiento del sistema y la carga de recursos que posea en esos momentos.
COUNTER es un contador que lleva la cuenta del número total de fotogramas sucesivos que tienen una relación de aspecto de ojo inferior a EYE_AR_THRESH. Si este valor supera a FRAMES_CONCEC el sistema procede a la activación de la alarma correspondiente.

Figura 45 – Detección de ojos abiertos

Figura 46 – Detección de ojos cerrados

Figura 47 – Detección de ojos abiertos utilizando anteojos

Figura 48 – Detección de ojos cerrados utilizando anteojos

2.7.12 Conexión mediante Modem USB 3G

Para realizar la conexión 3G se utiliza el programa Sakis3G y UMTSKeeper. El primero es el encargado de utilizar los argumentos de configuración para la conexión y el segundo es el encargado de comprobar constantemente el estado, y en caso de perder la conexión, restablecerla.

Adicionalmente, se añade la siguiente línea de comando al archivo rc.local ubicado en el directorio /etc, lo que tiene como objetivo iniciar la conexión durante el arranque del sistema y mantener el proceso en ejecución.

```
PATH/umtskeeper –sakisoperators “USBINTERFACE=’0’ OTHER=’USBMODEM’ USBMODEM=’12d1:XXXX’ APN=’apn_proveedor’ APN_USER=’user’ APN_PASS=’user’” –sakisswitches “–sudo –console” –devicename ‘Huawei’ –log –silent –monthstart 8 –nat ‘no’ &
```

2.7.13 Inicio automático

La forma de ejecutar comandos o programas al inicio del sistema de forma automática puede realizarse mediante la utilización de servicios. Dicho de otra forma, el programa principal es agregado a un servicio propio, donde fácilmente luego se puede iniciar, detener o deshabilitar desde la línea de comandos.

Por lo tanto, primero se debe crear un archivo “.service” y agregar los argumentos necesarios para el inicio, en nuestro caso el archivo creado es “reco.service”.

```
[Unit]
Description=Reconocimiento Facial
After=multi-user.target

[Service]
ExecStart=/usr/bin/python3  home/pi/recoFacial/reco.py
Type=idle
Restart=always

[Install]
WantedBy=multi-user.target
```

Esto define un nuevo servicio llamado "Reconocimiento Facial" y estamos solicitando que se inicie una vez que el entorno multiusuario esté disponible, luego de cada inicio. El parámetro "ExecStart" se utiliza para especificar el programa que queremos ejecutar. “Type” se establece en inactivo para garantizar que el comando ExecStart se ejecute solo cuando todo lo demás se haya cargado. Entonces, en este caso, el servicio ejecutará “reco.py” desde nuestro directorio de trabajo “/home/pi/recoFacial”.

Luego se pueden utilizar los siguientes comandos para iniciar, detener, habilitar, o deshabilitar el servicio respectivamente:

```
sudo systemctl start reco.service
sudo systemctl stop reco.service
sudo systemctl enable reco.service
sudo systemctl disable reco.service
```

Finalmente, el directorio del proyecto de reconocimiento final y detector de somnolencia se muestra en la imagen siguiente. Contiene, entre otras cosas, la base de datos de las personas, archivos necesarios para los detectores, y scripts en Python de los códigos implementados.

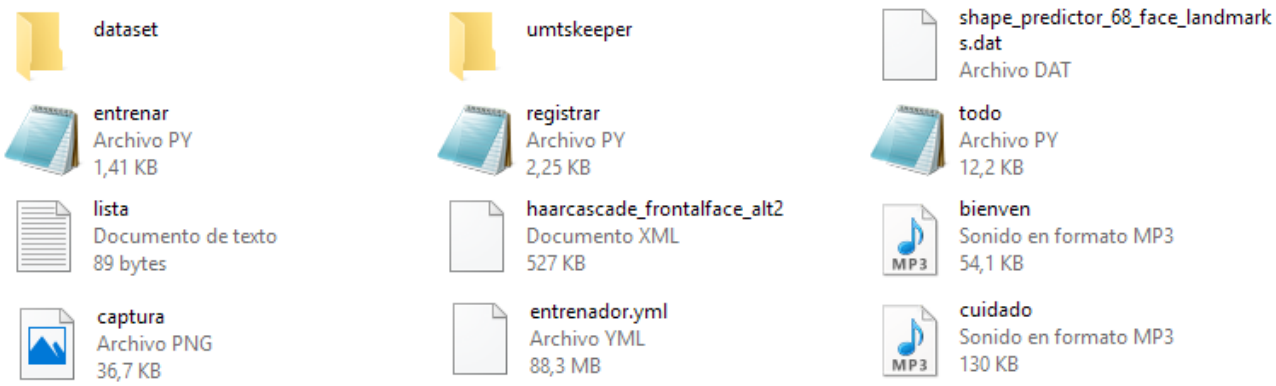

Figura 49 – Directorio del proyecto

2.7.14 Diagrama de flujo del proyecto total

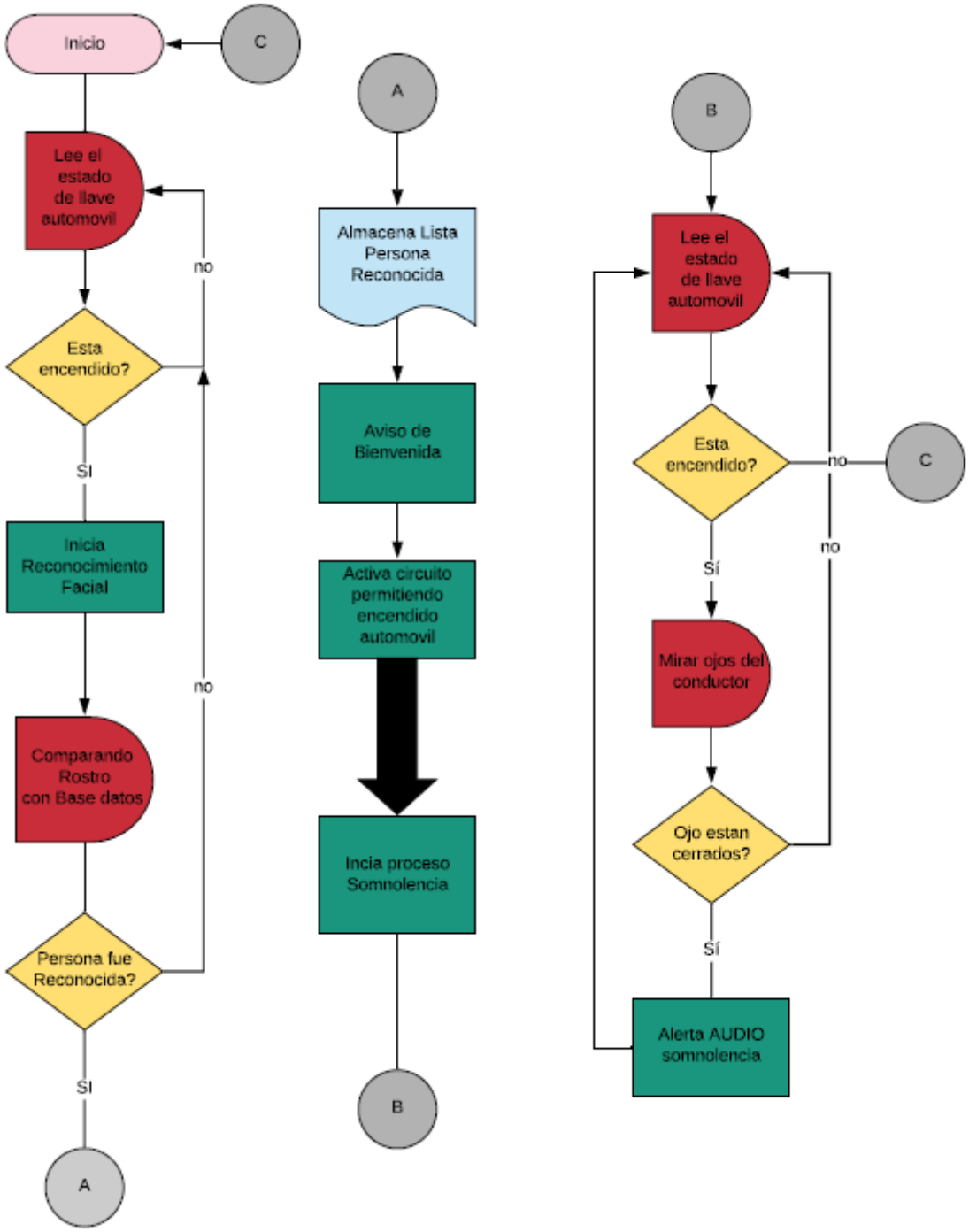

Diagrama 5 – Diagrama en bloques del proyecto general

2.8 Hardware adicional

Debido a las necesidades que requería el proyecto, se diseñó un circuito impreso en PCB para realizar diferentes acciones, como por ejemplo, poder leer el estado de contacto del vehículo y amplificar la salida de audio de las alertas.

2.8.1 Amplificador de audio

La señal de alerta que emite la Raspberry Pi no posee la corriente necesaria para alimentar un parlante y poder escuchar los audios, por lo que se implementó un amplificador con el integrado LM386. La salida del canal izquierdo y derecho se suman en la entrada no inversora del amplificador y a la salida se le conecta un parlante de 3 [W] y 4 [Ω].

El circuito implementado se muestra a continuación:

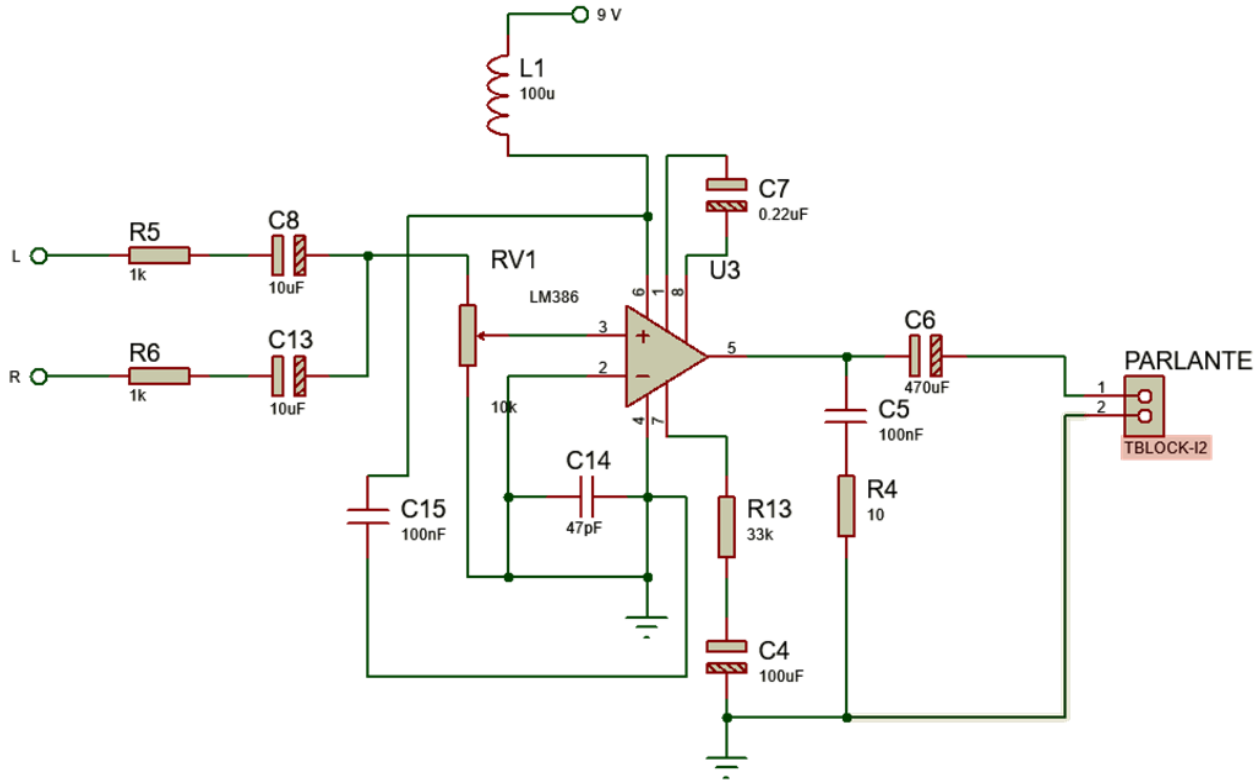

Figura 50 – Esquema circuital del amplificador implementado

2.8.2 Lector del estado de llave contacto

Con el circuito que se presenta a continuación se procedió a leer el estado en que se encuentra la llave de contacto. Los estados de la llave varían con cada vehículo, pero por lo general se encuentran STOP – ACC – MARCHA – IGNICION. El estado que necesitamos conocer es el de MARCHA, ya que en ese momento se inicia el

reconocimiento de personas y luego el de somnolencia, siempre y cuando, se haya autorizado el encendido y la llave siga en contacto de MARCHA.

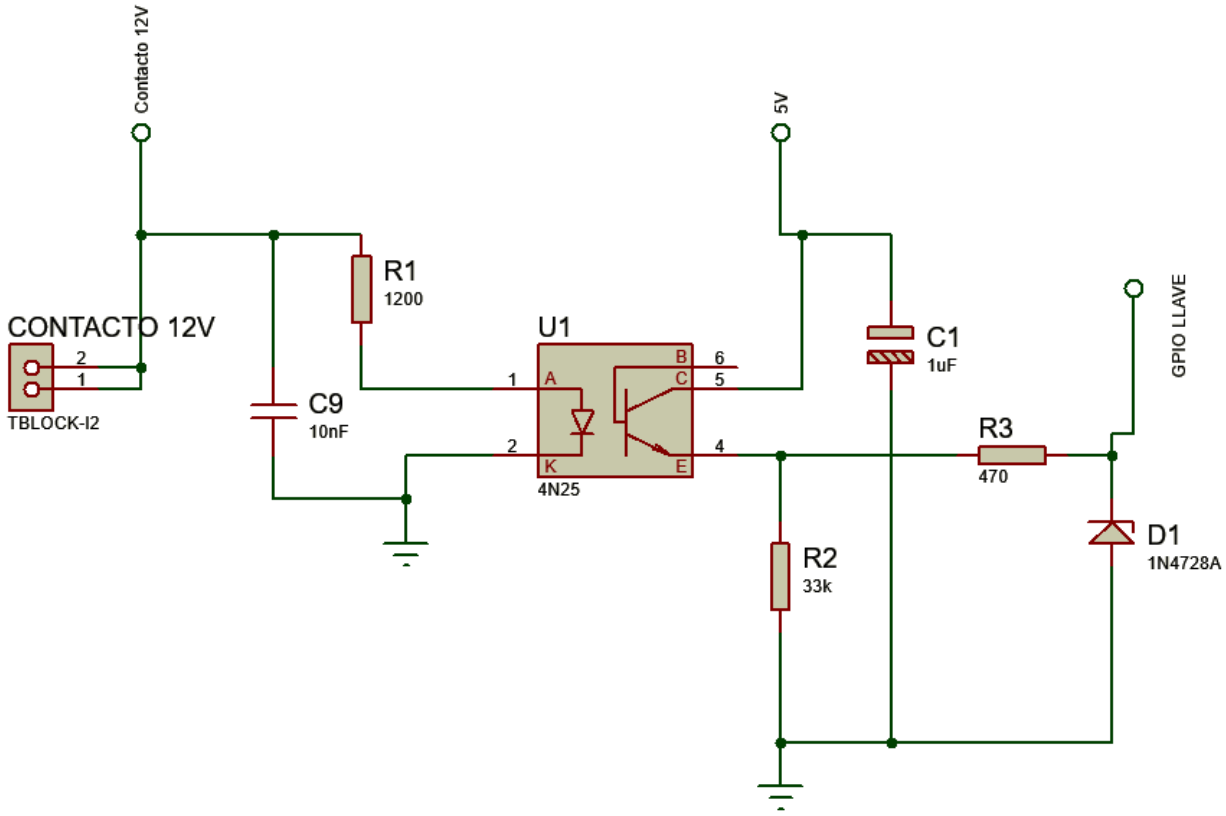

Figura 51 – Esquema circuital del lector de llave contacto

En la programación, el pin 17 de la GPIO se configura como entrada y es el encargado de leer constantemente el estado de la llave.

```
GPIO.setup(17, GPIO.IN)
estado = GPIO.input(17)
time.sleep(0.1)
```

2.8.3 Autorización del arranque

En este caso, el circuito implementado se basa en un temporizador realizado con el circuito integrado 555, el cual recibe el pulso proveniente de la Raspberry (reconocimiento exitoso) para encender el auto, y mantiene ese pulso por 35 segundos, el cual hará conmutar un relé de protección que será el encargado de enviar la señal de ignición al sistema del motor de arranque.

$$T = 1{,}1 \cdot R \cdot C = 1{,}1 \cdot 68\ [K\Omega] \cdot 470\ [\mu F] \approx 35\ [s]$$

Ecuación 3 – Tiempo del pulso a la salida del CI

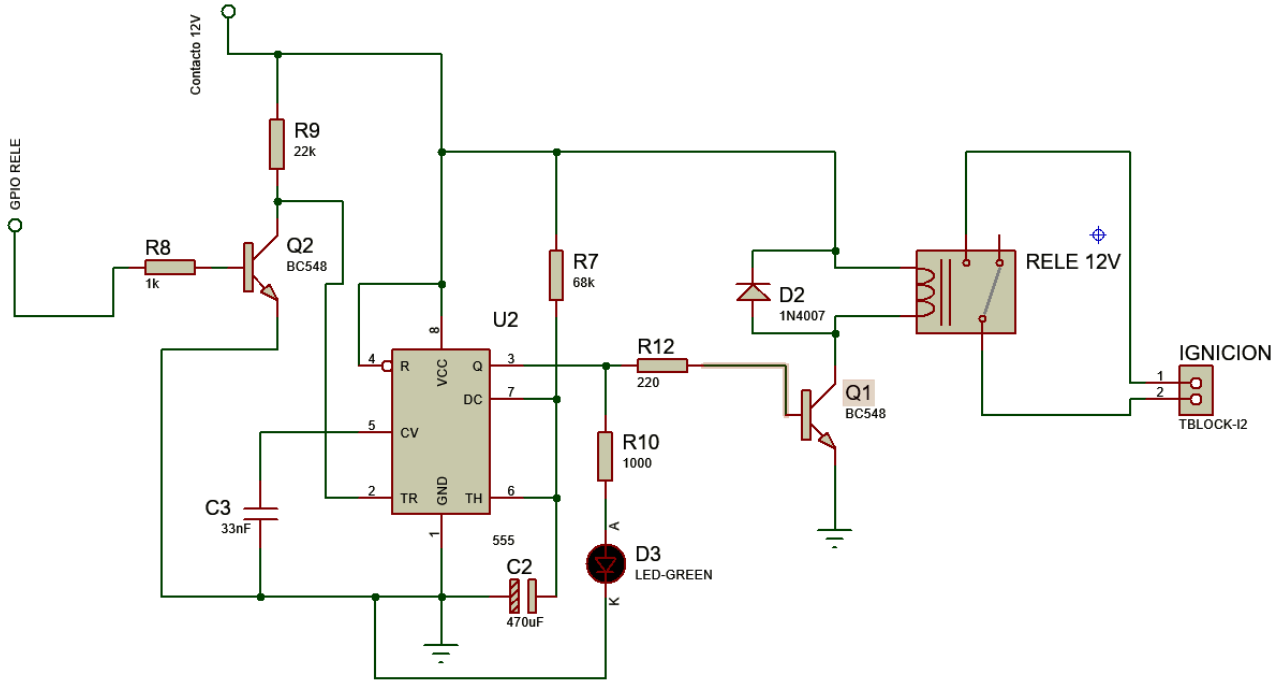

Figura 52 – Esquema circuital de la habilitación del arranque

El pin 18 de la GPIO configurado como salida será el encargado de enviar una señal de estado ALTO (HIGH) por 100 milisegundos, tiempo suficiente para que sea detectado por el 555 y extienda el pulso.

```
GPIO.setup(18, GPIO.OUT)
GPIO.output(18,GPIO.HIGH)
time.sleep(0.1)
GPIO.output(18,GPIO.LOW)
```

2.8.4 Apagado seguro

Para realizar el apagado del sistema de forma segura, ya sea para reiniciarlo, o desconectarlo del circuito eléctrico, se implementó un esquema con un pulsador, y la Raspberry al detectar el cambio de estado de la entrada por el pin 27 referido a la GPIO, detiene la ejecución del programa principal y ejecuta la instrucción de apagar.

```
GPIO.setup(27, GPIO.IN)
GPIO.setup(23, GPIO.OUT)
power = GPIO.input(27)
time.sleep(0.1)
GPIO.output(23,GPIO.HIGH)
if power == True:
    break
```

```
        …
        …
GPIO.output(23,GPIO.LOW)
cam.stop()
os.system('sudo shutdown now')
```

El pin 23 es utilizado como una salida que alimenta un LED el cual solo sirve como control para saber que el código principal está siendo ejecutado.

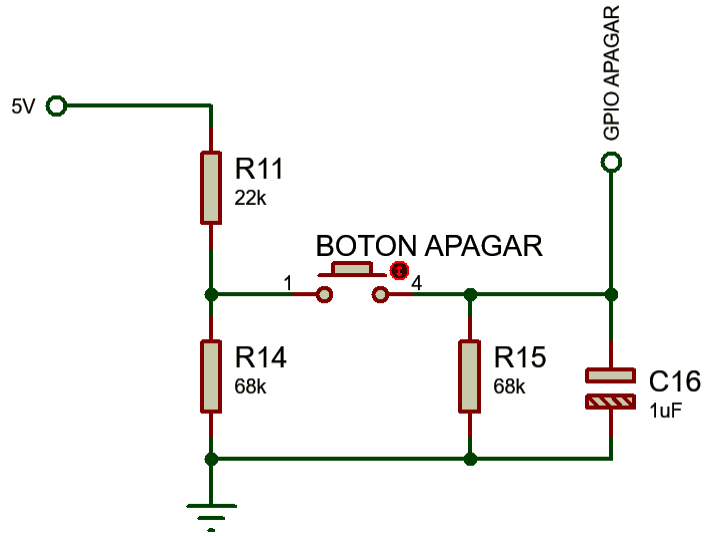

Figura 53 – Circuito implementado para forzar el apagado

2.9 Diseño Completo

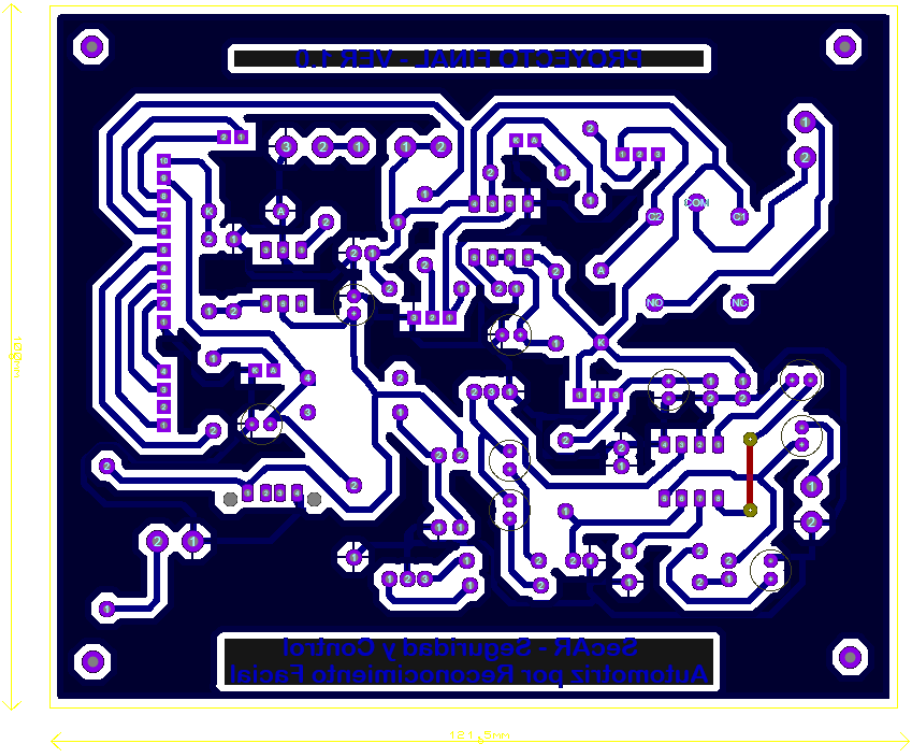

Figura 54 – Diseño del PCB

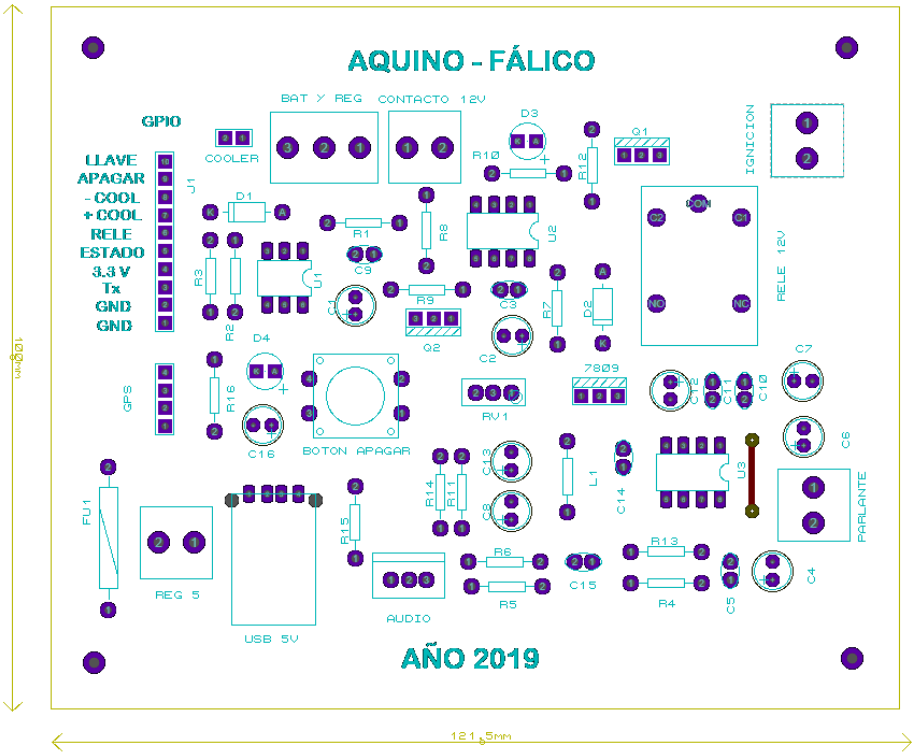

Figura 55 – Ubicación de los componentes en el diseño

Figura 56 – Vista superior del PCB en 3D

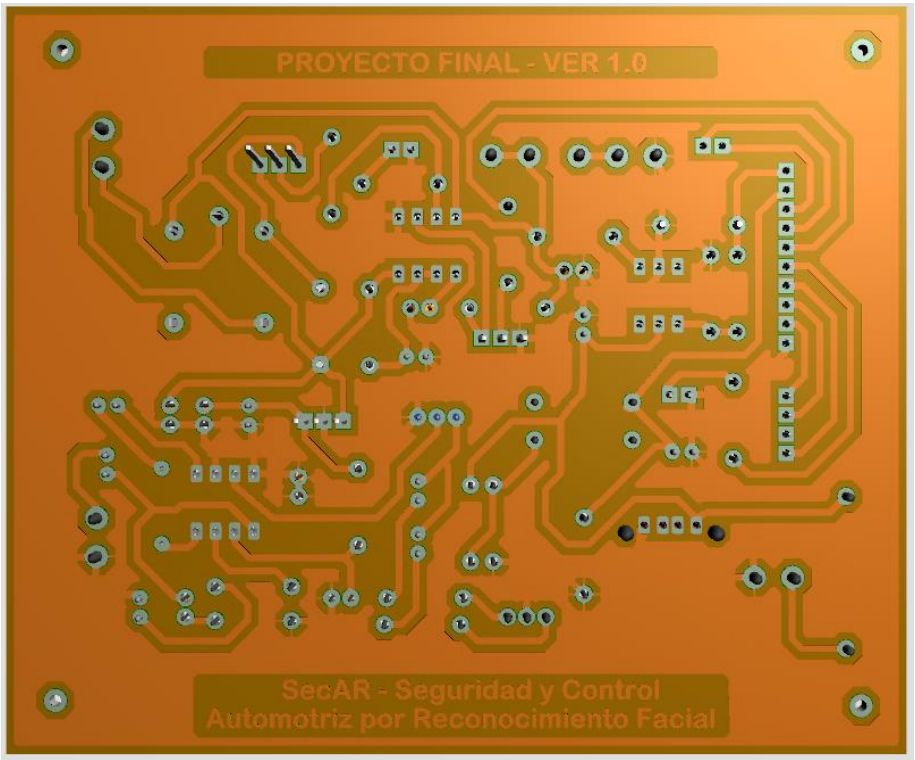

Figura 57 – Vista inferior del PCB en 3D

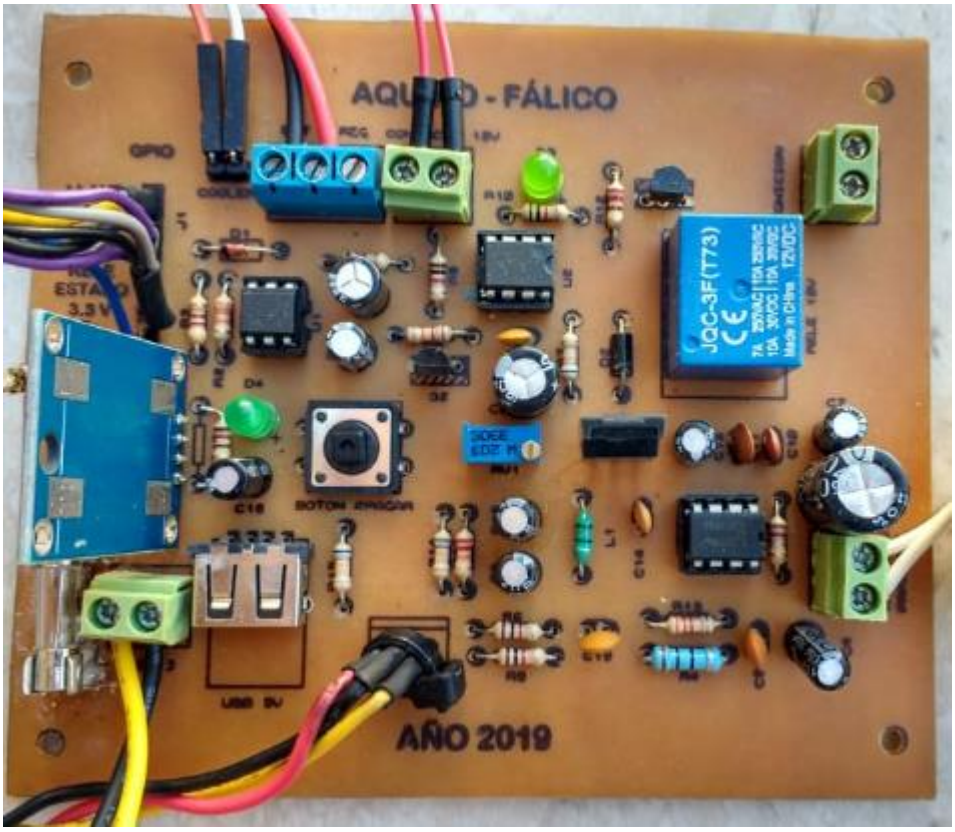

Figura 58 – Placa PCB realizada

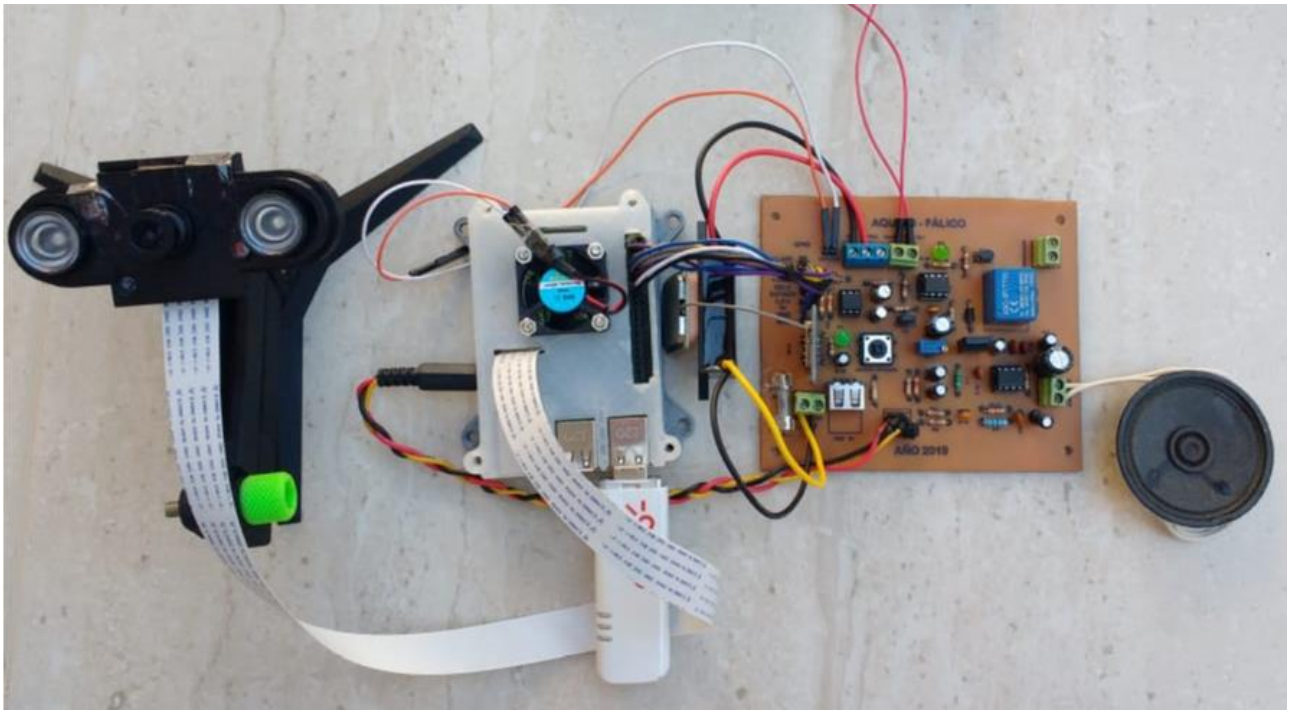

Figura 59 – Proyecto completo

Capítulo 3: Resultados

Todas las pruebas y experiencias han sido realizadas en un rango de distancia aproximado de 60 [cm] a 1 [m], siendo estas distancias las más frecuentes entre la posición de la cámara y los conductores.

3.1 Resultado de la detección

A continuación, se muestran los resultados de aplicar la detección de rostros sobre una base de 200 muestras, tomadas en el día y en la noche.

Rostros		
Día		
Cantidad de capturas	**Detecciones exitosas**	**Aciertos [%]**
200	200	100
Noche		
Cantidad de capturas	**Detecciones exitosas**	**Aciertos [%]**
200	195	97,5

Tabla 2 – Tabla de porcentaje en la detección de rostros

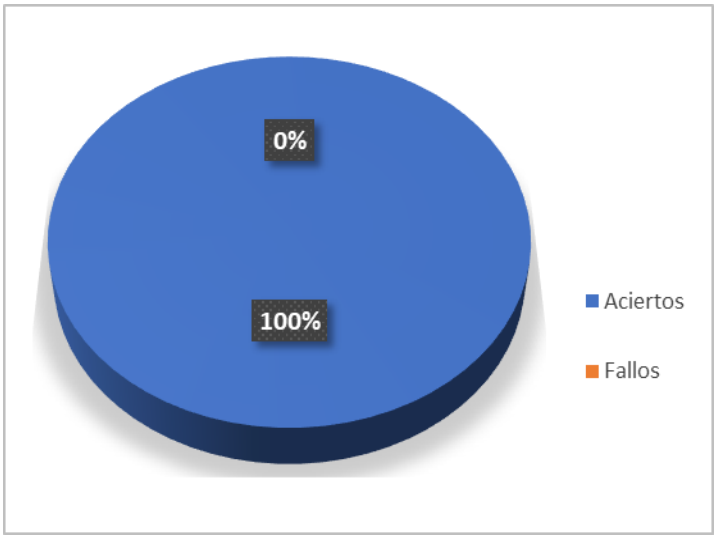

Figura 60 – Porcentaje en la detección de rostros en el día

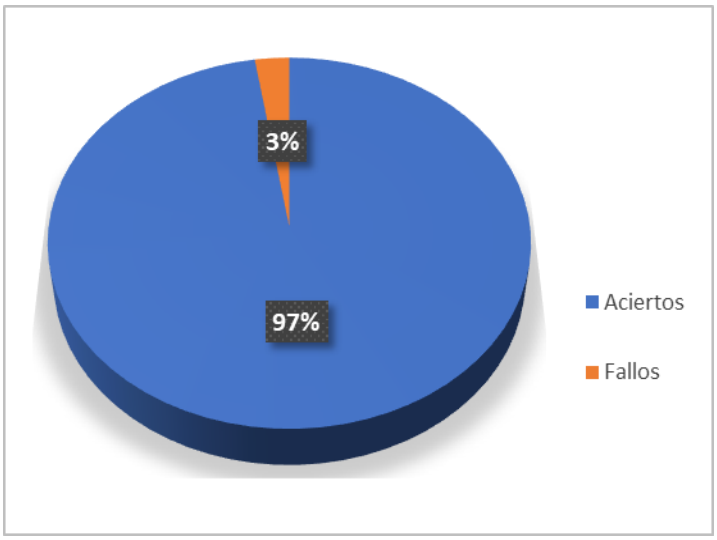

Figura 61 - Porcentaje en la detección de rostros en la noche

3.2 Resultado del reconocimiento

En este punto, de detallará el porcentaje de aciertos en el reconocimiento de personas en la base de datos. Para ello, las muestras se realizaron durante el día y la noche, tomando a los autores como objetivo, y sobre un grupo de personas al azar para el correcto funcionamiento de una detección registrada como “desconocido”.

Gastón		
Día		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	81	81
Noche		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	72	72

Tabla 3 – Tabla de porcentaje de reconocer al usuario Gastón

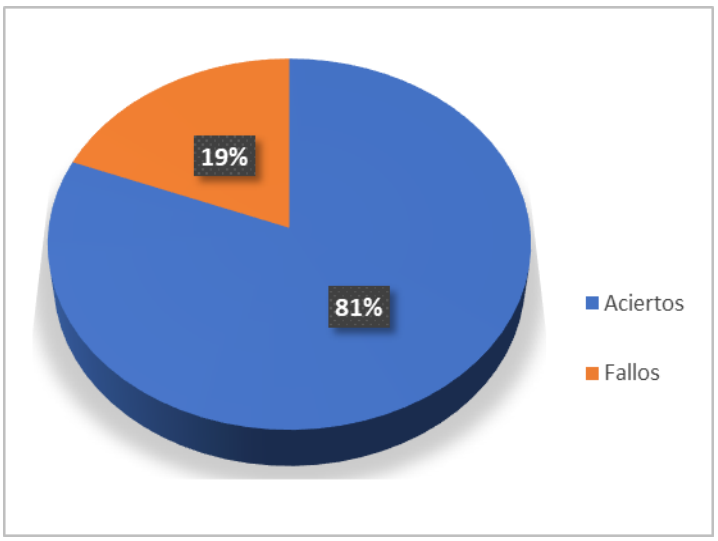

Figura 62 - Porcentaje de reconocimiento en el día para el usuario Gastón

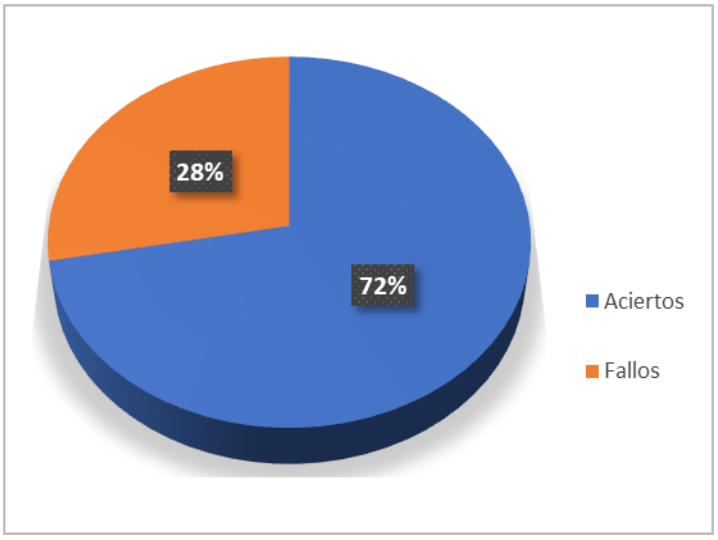

Figura 63 - Porcentaje de reconocimiento en la noche para el usuario Gastón

Ezequiel (LENTES)		
Día		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	92	92
Noche		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	88	88

Tabla 4 – Tabla de porcentaje de reconocer al usuario Ezequiel con anteojos

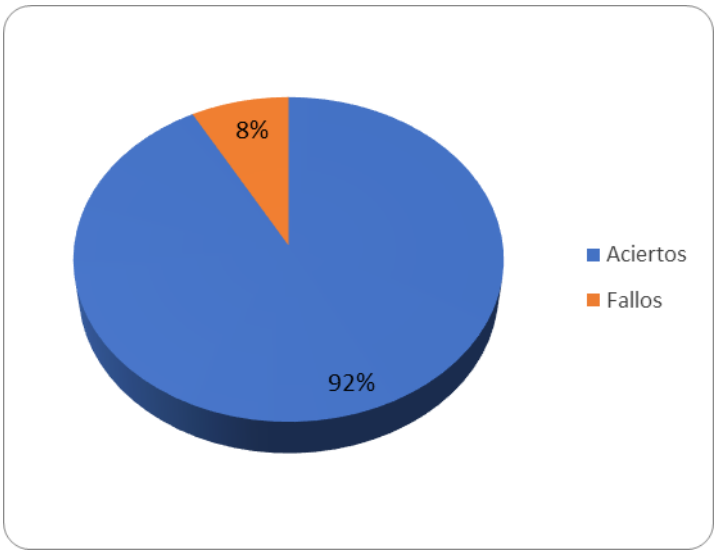

Figura 64 - Porcentaje de reconocimiento en el día para el usuario Ezequiel (Lentes)

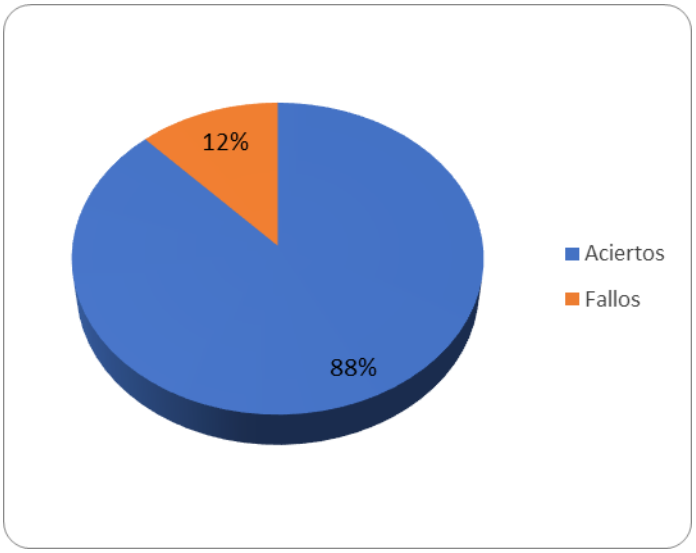

Figura 65 - Porcentaje de reconocimiento en la noche para el usuario Ezequiel (Lentes)

Ezequiel (SIN LENTES)		
Día		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	85	85
Noche		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	95	95

Tabla 5 - Tabla de porcentaje de reconocer al usuario Ezequiel sin anteojos

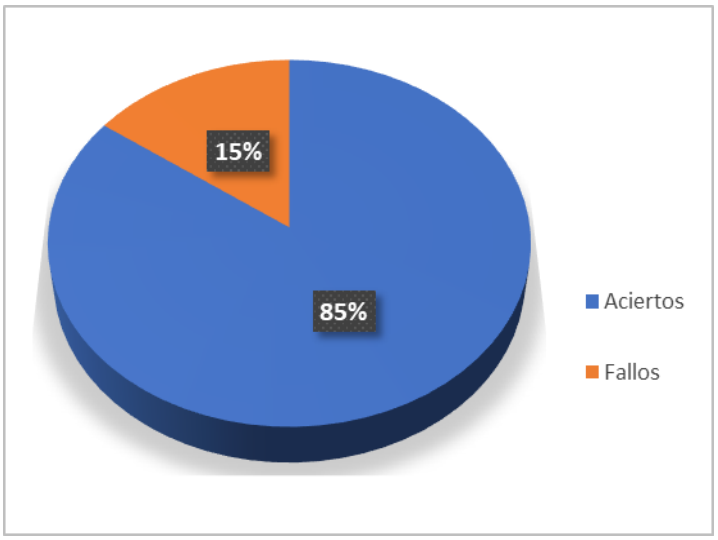

Figura 66 - Porcentaje de reconocimiento en el día para el usuario Ezequiel (Sin lentes)

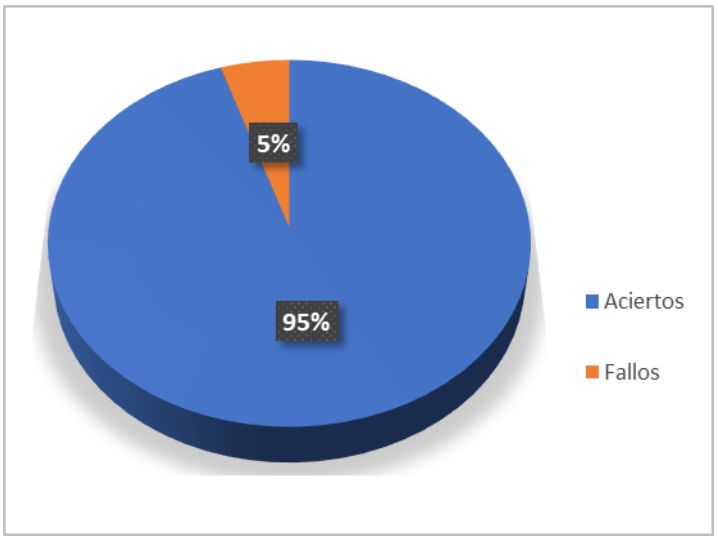

Figura 67 - Porcentaje de reconocimiento en la noche para el usuario Ezequiel (Sin lentes)

Desconocidos		
Día		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	87	87
Noche		
Cantidad de capturas	**Reconocimiento exitoso**	**Aciertos [%]**
100	83	83

Tabla 6 – Tabla de porcentaje de reconocimiento a desconocidos

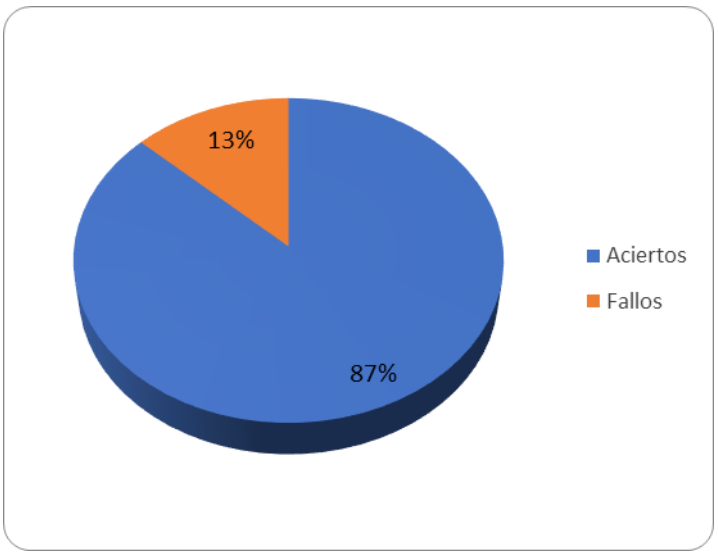

Figura 68 - Porcentaje de reconocimiento a desconocidos en el día

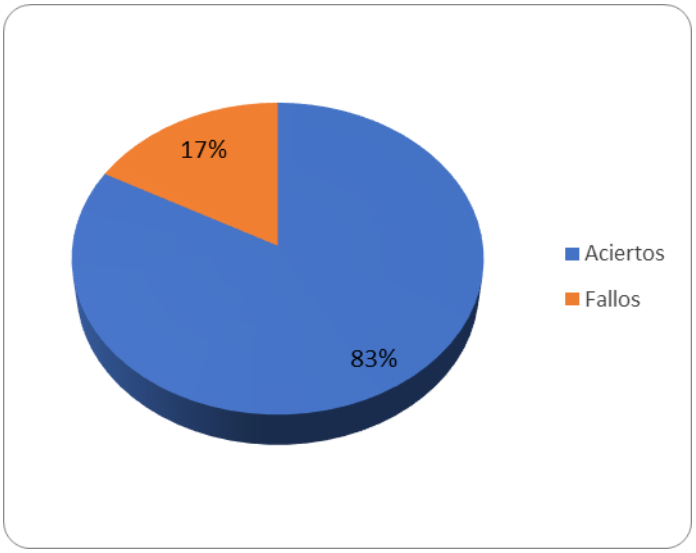

Figura 69 - Porcentaje de reconocimiento a desconocidos en la noche

Además, se cronometró el tiempo que requería el sistema para permitir el arranque del auto a una persona autorizada, obteniendo los siguientes datos para una serie de 20 muestras:

Muestra	Tiempo[s]	Promedio[s]
1	2	2,6195
2	2,2	
3	2,1	
4	3	
5	2,7	
6	2,3	

7	3,1	
8	3,4	
9	3	
10	2	
11	2,76	
12	2,03	
13	2,8	
14	3,78	
15	2,54	
16	2,55	
17	2,46	
18	2,68	
19	2,13	
20	2,86	

Tabla 7 – Tiempos de demora en habilitar el arranque

3.3 Resultado de control de somnolencia

El análisis se realiza en el día y en la noche, y considerando el caso de la utilización de anteojos para conducir. A continuación, se presenta el estudio de somnolencia en el día y sin uso de lentes.

Somnolencia (sin lentes)		
Día		
Cantidad de capturas	**Detección exitosa**	**Aciertos [%]**
50	39	78
Noche		
Cantidad de capturas	**Detección exitosa**	**Aciertos [%]**
50	35	70

Tabla 8 – Tabla de porcentaje en detectar somnolencia sin anteojos

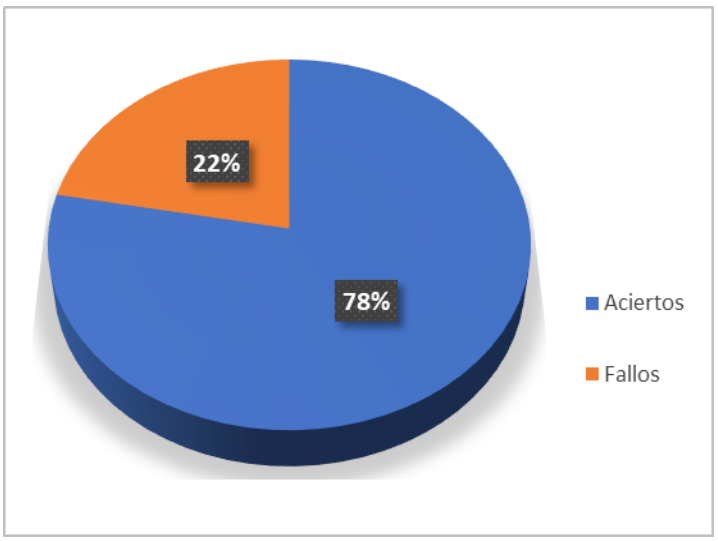

Figura 70 - Porcentaje en detectar somnolencia sin anteojos en el día

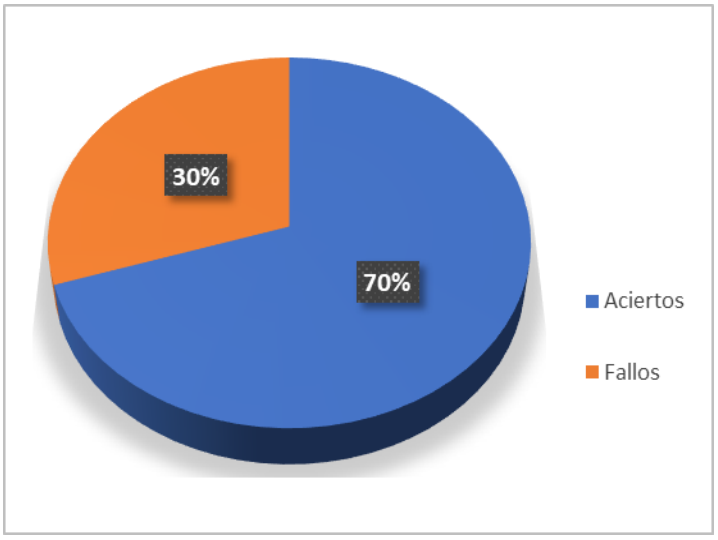

Figura 71 - Porcentaje en detectar somnolencia sin anteojos en la noche

Se tomaron 100 muestras para determinar el tiempo de detección de somnolencia, descartando del promedio, aquellas muestras que superan los 4[s] debido a que consideramos que la detección debe estar por debajo de ese valor. Este valor de detección depende principalmente de la velocidad a la que esté trabajando el CPU y la velocidad de frames por segundo en la toma de imágenes.

Día (sin lentes)				Noche (sin lentes)			
Muestra	**Tiempo[s]**	**Muestra**	**Tiempo[s]**	**Muestra**	**Tiempo[s]**	**Muestra**	**Tiempo[s]**
1	2,93	26	2,5	1	3,5	26	2,9
2	2,89	27	2,9	2	4,2	27	2,7
3	7,65	28	2,8	3	7	28	2,8
4	8,3	29	2,5	4	5,3	29	2,7
5	3,15	30	2,9	5	2,7	30	6
6	6	31	2,74	6	3,3	31	3,2
7	5	32	2,89	7	3,9	32	3,1
8	3,1	33	2,4	8	2,9	33	6
9	7	34	2,7	9	2,7	34	5
10	12	35	2,6	10	4,2	35	8,5
11	3,7	36	2,5	11	3	36	7,5
12	3,3	37	2,65	12	3,5	37	5,5
13	4,8	38	4,5	13	7	38	3
14	7	39	2,7	14	2,5	39	3
15	2,6	40	2,5	15	11	40	2,9
16	2,5	41	3,25	16	3,3	41	3,2
17	3,2	42	3,1	17	7	42	4,8
18	2,9	43	2,9	18	2,3	43	5
19	2,95	44	3,1	19	3,9	44	2,7
20	2,5	45	4	20	2,75	45	2,8
21	3,03	46	2,7	21	2,8	46	2,7
22	2,7	47	5	22	2,9	47	4,2
23	2,6	48	3	23	2,8	48	4,5
24	3,3	49	4	24	3	49	2,8
25	2,66	50	9	25	2,8	50	2,9

Tabla 9 – Tiempos de demora en alertar la somnolencia sin anteojos

El mismo análisis se llevó a cabo para personas que utilicen anteojos, el único detalle a comentar fue que la cámara tuvo que ser ubicada por encima de la línea de la cabeza para que el haz emitido por los leds infrarrojos no se refleje sobre los cristales, lo que ocasionaba una falsa detección de ojos.

Somnolencia (con lentes)		
Día		
Cantidad de capturas	**Detección exitosa**	**Aciertos [%]**
50	46	92
Noche		
Cantidad de capturas	**Detección exitosa**	**Aciertos [%]**
50	43	86

Tabla 10 – Tabla de porcentaje en detectar somnolencia con anteojos

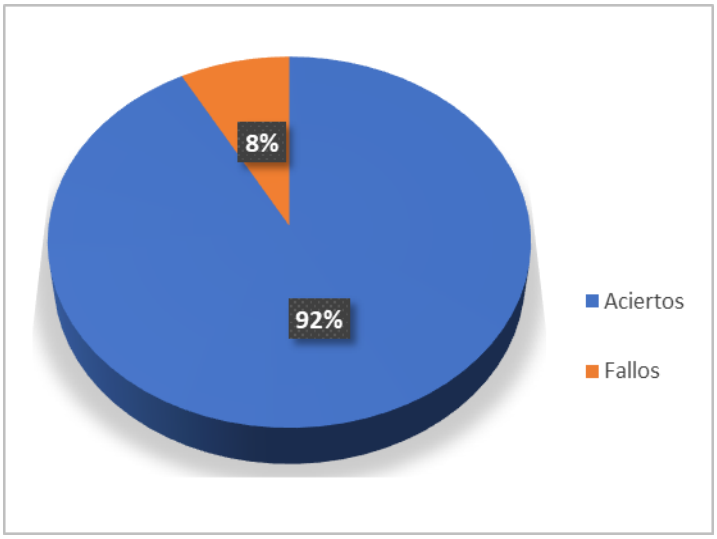

Figura 72 - Porcentaje en detectar somnolencia con anteojos en el día

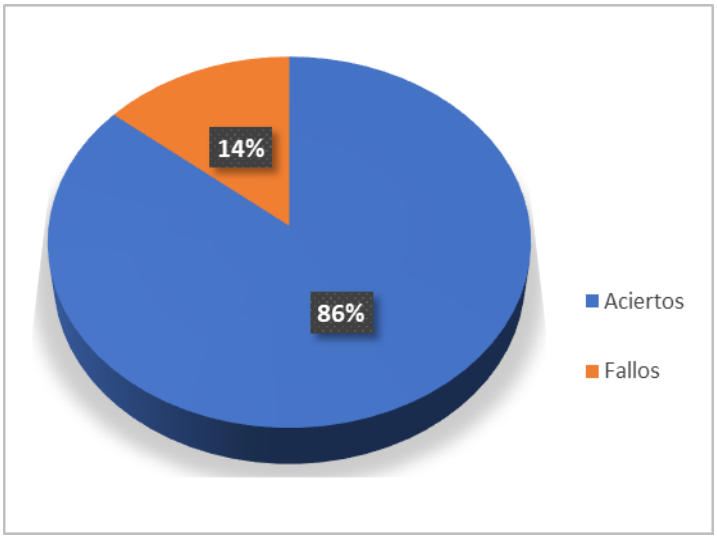

Figura 73 - Porcentaje en detectar somnolencia con anteojos en la noche

Dia (con lentes)				Noche (con lentes)			
Muestra	**Tiempo[s]**	**Muestra**	**Tiempo[s]**	**Muestra**	**Tiempo[s]**	**Muestra**	**Tiempo[s]**
1	1,91	26	2,1	1	2,46	26	3,23
2	2,38	27	2,43	2	2,37	27	3,08
3	2,2	28	2,26	3	2,36	28	2,67
4	2,48	29	2,3	4	2,9	29	5,72
5	2,49	30	2,3	5	3,48	30	4,55
6	2,08	31	10,3	6	4,86	31	3,39
7	2,11	32	2,33	7	3,18	32	2,28
8	2,39	33	2,6	8	3,25	33	2,18
9	3,13	34	2,52	9	3,03	34	4,95
10	2,92	35	2,47	10	3,03	35	3,77

11	3,3	36	2,44	11	3,41	36	2
12	2,44	37	2,42	12	3,26	37	2,4
13	2,61	38	2,4	13	3,19	38	3,01
14	2,28	39	2,58	14	3,16	39	3,1
15	2,54	40	2,91	15	2,73	40	2,74
16	2,43	41	3	16	2,68	41	2,46
17	2,36	42	2,77	17	2,48	42	2,29
18	2,43	43	3,13	18	2,51	43	2,43
19	2,56	44	4,48	19	2,4	44	9,57
20	2,39	45	4,11	20	8,77	45	2,47
21	2,4	46	3,43	21	2,6	46	5,48
22	2,88	47	3,69	22	3,62	47	3,05
23	2,99	48	2,74	23	2,75	48	2,53
24	2,78	49	3,25	24	3,38	49	3,23
25	2,78	50	4,73	25	2,5	50	2,61

Tabla 11 – Tiempo de demora en alertar somnolencia con anteojos

Finalmente, podemos presentar un promedio del tiempo que conlleva detectar somnolencia en este sistema:

Tiempo Promedio total en el Dia [s]
3,3254
Tiempo Promedio total en la Noche [s]
3,662

Tabla 12 – Tiempos promedio para detectar somnolencia

Capítulo 4: Análisis de Costos

4.1 Costos de fabricación

A continuación, mostraremos una tabla con los valores aproximados de costos para la fabricación de una unidad de producto:

Raspberry Pi 3 B	USD	69,17
Cámara Infrarroja	USD	41,50
Cooler y disipadores	USD	8,33
Cable Plano 60 cm	USD	8,17
Base y soporte	USD	13,33
Componentes Varios	USD	12,33
Módulo GPS con Antena	USD	14,17
Tarjeta SD 16 GB	USD	8,83
Parlante 3 W	USD	8,33
COSTO TOTAL x UNIDAD	**USD**	**184,17**

Tabla 13 – Costos de fabricación por unidad

El valor comercial de venta estimado es de USD 500,00 considerando el valor agregado por trabajo y diseño de hardware/software. Los gastos de adquirir un modem 3G USB con un plan de datos corre a cuenta del cliente.

A su vez, el precio de venta está limitado a una cantidad de 2 personas que se desean agregar a la base de datos. Superando esta cuantía, se cobra un extra de USD 15 por cada individuo.

Capítulo 5: Discusión y Conclusión

5.1 Conclusiones

Como conclusión del trabajo realizado podemos destacar que se diseñó un sistema sencillo y eficaz para reconocer personas y llevar un registro de uso, sin necesidad de hacer grandes modificaciones a la instalación eléctrica/mecánica del automóvil.
Por su parte, el sistema también presentó buenas prestaciones a la hora de detectar somnolencia, y en cuanto a la comunicación con el usuario final, se pudo utilizar herramientas gratuitas y de gran utilidad como es Telegram para el envío y recepción de comandos determinados. Además, podemos mencionar que el sistema puede ser instalado en cualquier tipo de vehículos, desde pequeños automóviles hasta grandes camiones y colectivos, ya que en estos últimos, pese a tener baterías de 24 [V] el circuito eléctrico en algún punto de la instalación los reduce a 12 [V].
Gracias a todas estas características, el sistema puede ser empleado como un simple complemento de seguridad a un automóvil particular o familiar, y como un sistema de control a grandes empresas de transporte o flota de vehículos.

Desde el punto de vista negativo, podemos mencionar que el sistema no es fiable en la detección de somnolencia al utilizar anteojos de visión donde se refleje mucha luz en ellos, y en anteojos de sol donde se dificulta la localización de los ojos. Otro punto a considerar es la gran cantidad de fotos que se deben tomar para entrenar la base de datos (300 aproximadamente para cada persona), ya que el reconocimiento de rostros es sensible a distintas condiciones de luminosidad y distancia, y aún así, se pudo observar que el proceso no es 100% confiable para determinar la identidad de un individuo, habiendo casos en los que se identifica por error que un desconocido pertenece a la base de datos.

5.2 Mejoras a futuro

Como mejoras para este proyecto podemos mencionar que desde el punto de vista del diseño del hardware y PCB se podría perfeccionar aún más. En cuanto al reconocimiento de personas, se podría investigar sobre nuevas herramientas para realizar una detección más precisa sobre las características faciales de las personas. Además de esto, trabajar sobre un detector de vida que funcione conjuntamente con el reconocimiento de los individuos, para evitar que una persona ajena pueda encender el vehículo utilizando fotografías impresas de los conductores autorizados.

Considerando la detección de somnolencia, podemos decir que una mejora a futuro sería añadir nuevos métodos de detección como, por ejemplo, bostezos y que la visual del conductor esté puesta sobre el camino. Además, dado que el sistema utiliza un módulo GPS, una mejora sería calcular la velocidad del automóvil y realizar la detección de somnolencia superando cierto valor, como ser por ejemplo, 30 [Km/h].

Por último, se podría investigar formas de reducir el consumo de energía para proveer al sistema de mayor cantidad de horas de funcionamiento mientras el vehículo se encuentra apagado, o bien, implementar mediante diferentes sensores un arranque inteligente del dispositivo a la hora de que los conductores entren al habitáculo para encender el motor, y así evitar un consumo de corriente innecesario en momentos que no se requiere. Otra mejora sería alimentar por separado los LED infrarrojos y controlar su encendido, ya que es otra fuente de consumo importante en el sistema, al ser de alta potencia lumínica.

Capítulo 6: Bibliografía

- Adrian Rosebrock. (2019). Obtenido de https://www.pyimagesearch.com/
- Cajas Idrovo; Viri Ávila, M. V. (2017). *Diseño e implementación de un sistema de seguridad vehicular mediante reconocimiento facial a través de visión artificial.* Cuenca, Ecuador.
- Documentación de OpenCV. (2019). Obtenido de https://docs.opencv.org/
- Elías de Mintaka. (2019). Obtenido de http://mintakaconciencia.net/squares/umtskeeper/
- España Tarira; Oña Paredes, L. G. (2018). *Implementación de un prototipo para la detección de signos de fatiga del conductor aplicando visión artificial en un vehículo liviano en la noche.* Quito, Ecuador.
- Fundación Raspberry Pi. (2019). Obtenido de https://www.raspberrypi.org/documentation/
- Nick Lee. (2018). Obtenido de https://telepot.readthedocs.io/en/latest/reference.html
- Oficina Nacional de Coordinación de USA. (2019). Obtenido de https://www.gps.gov/spanish.php
- OMS. (2018). Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/road-traffic-injuries
- OpenCV. (2019). Obtenido de https://opencv.org/
- Telegram. (2019). Obtenido de https://telegram.org/