Universidad Tecnológica Nacional

Proyecto Final

AVSA – Asistente vehicular de seguridad

Autores:

- Follonier, Maximiliano
- Peroni, Luciano

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

Facultad Regional Paraná

Agosto de 2018

Declaración de autoría:

Nosotros declaramos que el Proyecto Final “AVSA – Asistente Vehicular de Seguridad” y el trabajo realizado son propios. Declaramos:

- Este trabajo fue realizado en su totalidad, o principalmente, para acceder al título de grado de Ingeniero Electrónico, 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:

Agradecemos a la institución por la formación, a los docentes de la carrera por su constante disposición ante nuestras consultas y a nuestras familias por el apoyo incondicional.

Follonier, Maximiliano Andrés

Peroni, Luciano Nicolás

Universidad Tecnológica Nacional

Abstract

Facultad Regional Paraná

Ingeniero en Electrónica

AVSA – Asistente Vehicular de Seguridad Avanzada

Follonier, Maximiliano Andrés
Peroni, Luciano Nicolás

Abstract:
The frequent human errors in the driving of transport vehicles, such as, the use of the mobile phone or the excess of fatigue generate infinity of accidents.
This project tries to reduce these errors through the use of technological tools providing driver assistance.
The system is based on real-time image processing to monitor the actions of the driver, by detecting flicker and the position of the head. The OpenCV tool has been used to accurately detect the eye and calculate its aspect ratio. With this information, you can determine the opening or not of the eye, if it persists closed for a critical time indicates drowsiness. If this happens, the system will alert the driver.
The device consists of a low-cost computer board responsible for processing the images captured by the camera. The interaction with the driver of the vehicle is through a touch screen and an audio system, which are responsible for alerting in particular conditions, managing to predict the status of the driver in 90%.

Keywords: Image processing, blink detection, drowsiness, distraction.

Resumen:
Los recurrentes errores humanos en la conducción de vehículos de transporte tales como el uso del teléfono móvil, o el exceso de cansancio han generado hasta el día de hoy una infinidad de accidentes diarios.
Este proyecto buscó mitigar dichos errores mediante el uso de herramientas tecnológicas brindando una asistencia de manejo al conductor.
El sistema se basa en el procesamiento de imágenes en tiempo real para monitorear las acciones del conductor, mediante la detección de parpadeo y la posición de la cabeza. Se ha utilizado la herramienta OpenCV para detectar de forma exacta el ojo y calcular su relación de aspecto. Con esta información, se puede determinar la apertura o no del ojo, si el mismo persiste cerrado por un tiempo critico indica somnolencia. Si esto acontece, el sistema alertará al conductor.
El dispositivo final dispone de una placa computadora de bajo costo encargada de realizar el procesamiento de las imágenes capturadas por la cámara. La interacción con el conductor del vehículo es a través de una pantalla táctil y un sistema de audio, los cuales se encargan de alertarlo en condiciones particulares, logrando predecir el estado del conductor en un 90% de los casos.

Palabras Clave: Procesamiento de imagen, detección de parpadeo, somnolencia, distracción.

Reconocimientos:

Se agradece a los docentes de Proyecto Final por su dirección y ayuda constante, en especial por su orientación metodológica y por su continuo estimulo durante todo el proceso hasta al final del mismo.

Índice:

Lista de Figuras:

Lista de Tablas

Lista de Abreviaciones

OMS	Organización mundial de la salud
RAE	Real academia española
GPS	Global positioning system
MLL	Machine learning library
GNU	sistema operativo de tipo Unix
TIOBE	The Importance of Being Earnest
GUI	Interfaz gráfica de usuario
iOS	Sistema operativo móvil de la multinacional Apple
SVM	Support vector machines
HOG	Histograma de gradiente orientado
CPU	Unidad central de proceso
HD	High definition o alta definición
CSI	Interfaz serie para cámaras
LED	Diodo emisor de luz
UTC	Tiempo universal coordinado
GPIO	Entrada/Salida de propósito general
GTK	GIMP Tool Kit, biblioteca con objetos y funciones para interfaz gráfica
EAR	Relación de aspecto del ojo
UART	Transmisor-Receptor asíncrono universal
3D	Tres dimensiones
UV	*Radiación ultravioleta*

Lista de Símbolos

Δ	Delta, indica incremento
$\boldsymbol{\varphi}$	Alfabeto griego, Phi en minúscula
$\boldsymbol{\lambda}$	Alfabeto griego, Lambda
$\mathbf{\lvert\rvert}$	Valor absoluto
+	Indica adición
-	Indica sustracción
=	Indica igualdad
()	Indica agrupamiento
$\sqrt{}$	Raíz cuadrada

Dedicado a:

A nuestras familias por el apoyo incondicional

1. Introducción

1.1. Problemática:

Según un informe sobre la situación mundial de seguridad vial de la OMS (Organización mundial de la salud) cada día mueren en todo el mundo más de 3.000 personas por accidentes de tránsito, cifra que se concentra en un 85% en países de ingresos medios y bajos, entre los cuales se ubica la Argentina. Quedarse dormido al volante es una de las primeras causas de accidentes de tráfico en el mundo. Hasta un 25% de los accidentes se producen por esta causa, poniéndose al mismo nivel que los accidentes provocados por el consumo de alcohol.

En la Argentina mueren veinte personas por día en accidentes de tránsito, cinco de ellas por somnolencia. A ello se suma el uso del teléfono celular cuando se conduce, provocando con ello que el conductor aparte la vista del camino, o disminuya su atención. Esto último aumenta las probabilidades de sufrir accidentes de tránsito.

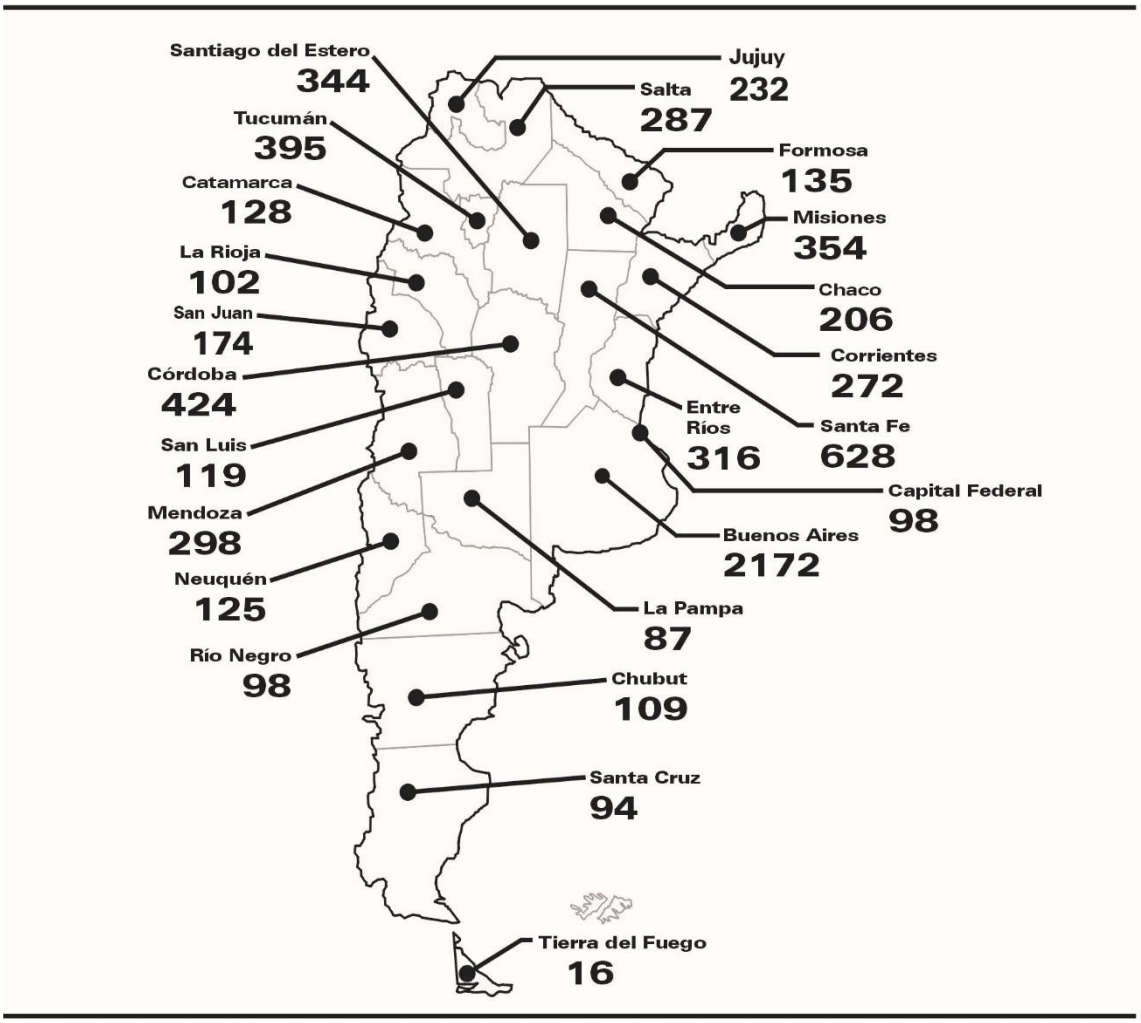

Fig. 1 Problemática del proyecto. Accidentes de tránsito anuales en el país

En el gráfico anterior puede observarse las muertes por accidentes de tránsito anuales en nuestro país. *(Asociación Civil Luchemos Por La Vida - Año 2017)*

Si se compara las estadísticas con otros países en un lapso correspondiente a los últimos años, la Argentina ha sido el único país donde no se han reducido las tasas de mortalidad, sino que han aumentado.

Año	1990	2000	2008	2012	2014	Porcentaje de disminución de muertos 1990-2014
Suecia	772	591	397	285	282	63%
Holanda	1.376	1.082	677	566	570	59%
Estados Unidos	44.599	41.495	37.423	33.561	32.675	27%
España	9.032	5.777	3.100	1.903	1.680	81%
Argentina	**7.075**	**7.545**	**8.205**	**7.485**	**7.613**	**0%**

Tabla 1 Problemática del proyecto. Tasa de mortalidad en Argentina y otros países

1.2. Distracción y somnolencia:

A pesar de lo expuesto anteriormente, existen pocos estudios que se centren en el conductor y en su estado. Un estudio publicado por el departamento de psicología de la universidad de Helsinki, Finlandia, llevó a cabo unos ensayos para comprobar si la realización de tareas durante la conducción, tal como hablar por celular, afecta su rendimiento. La conclusión del estudio fue que la capacidad de alerta de los conductores se veía afectada en alrededor de 0,5 segundos en términos de tiempo de reacción al accionar el freno y casi 1 segundo en términos de tiempo de evitar una colisión. *(D. Lamble, K. Tatu, M. Laakso, H. Summala, 2010)*.

Se estima que las distracciones están presentes en un 38% de los accidentes mortales. Viajando a una velocidad media de 100 km/h, una distracción de un segundo es equivalente a recorrer a ciegas una distancia de 27 metros, con dos segundos, 55 metros. Todos ello sin tener en cuenta el tiempo de reacción de la persona al observar un obstáculo y considerando que el vehículo posee una distancia de frenado.

La más peligrosa y frecuente de las distracciones, es realizar una llamada telefónica, marcar un número de teléfono o buscarlo en la agenda suele requerir un tiempo que puede llegar hasta los 10 segundos, aumentando en forma considerable la posibilidad de sufrir un accidente.

El foco distracción puede generarse por el propio conductor como en el caso citado anteriormente o bien sufrir una distracción por agentes externos. Los agentes externos abarcan desde contemplar un paisaje, un cartel o un evento de un tercero. Aquí entra en juego la personalidad y la responsabilidad de cada conductor.

Según un estudio de *"Castrol and Race"*, los acompañantes son causa del 85% de las distracciones. Los principales motivos son: hablar con el copiloto (60%), atender a los niños (12%), mirar a los acompañantes (10%), insultos hacia la manera de conducir (6,64%), los movimientos de los niños en la parte trasera (6,53%), cambiar de música (6,35%) y mantener una discusión (4%).

El otro factor que influye en la conducción es la somnolencia. Según la *RAE* "La somnolencia es la sensación de pesadez y torpeza de los sentidos motivadas por el sueño"

La somnolencia es un estado fisiológico con una inclinación a quedarse dormido. Técnicamente es distinta de la fatiga que es la falta de disposición de seguir realizando la misma actividad. Una persona puede estar fatigada sin estar somnoliento pero las condiciones que producen fatiga como la conducción de automóviles en grandes distancias desenmascaran la presencia de somnolencia fisiológica, pero no la causan *(Stutts, Wilkins, & Vaughn, 1999)*.

Los efectos provocados por la somnolencia son:

- Disminución de la capacidad de reacción
- Disminución de la atención
- Disminución de la coordinación psicomotora

Signos de somnolencia fisiológica:

- Bostezos repetidos
- Parpadeo frecuente
- Inclinación de la cabeza

El estudio de estos síntomas proporciona información valiosa para el desarrollo de sistemas de detección de somnolencia. La fatiga produce cambios en las ondas cerebrales, la actividad de los ojos varia, las expresiones faciales cambian, los movimientos de la cabeza disminuyen, el pulso cardiaco se decelera, la presión que se realiza sobre el volante disminuye, el número de veces que el conductor se toca la cara, la cabeza, la oreja y los ojos se incrementa, se inclina ligeramente la cabeza hacia un lado a consecuencia de la relajación muscular del cuello y los patrones de parpadeo se ven alterados.

El trabajo realizado por *(Stutts JC, Wilkins JW, Scott Osberg J, Vaughn BV. Driver risk factors for sleep-related crashes 2003).* concluye que los conductores que duermen aproximadamente 5 horas, aumentan el riesgo de tener un accidente de tráfico en 5 veces. Los factores más importantes para distinguir entre si un accidente ha sido consecuencia de la somnolencia, son la duración del último periodo de sueño y la cantidad de horas dormidas en las últimas 24 horas. Los horarios de los conductores tienen impacto en la fatiga de los mismos, particularmente en vehículos comerciales. El riesgo de sufrir un accidente debido a la fatiga está muy relacionado con las horas totales

de conducción y con factores propios del conductor: tener más de un trabajo, tener turno de noche y trabajar más de 60 horas semanales.

Otro estudio dice que cuando una conducción se vuelve monótona, los estímulos recibidos de la misma no cambian o bien los cambios son predecibles. Esta monotonía ha sido identificada como causa principal de somnolencia en una gran cantidad de estudios, resultando que los accidentes debidos a somnolencia son más comunes durante la noche, en autovías anchas y monótonas. El 40% de los accidentes relacionados con somnolencia ocurren en autovías y el 30% de los accidentes en carreteras rurales.

1.3. Sistemas de seguridad en el mercado actual

Una investigación conjunta entre el Centro Europeo de Investigación e Innovación de Ford y una universidad alemana ubicada en Anchen, ha permitido desarrollar un asiento de monitoreo del ritmo cardíaco a través de sensores que pueden verificar la actividad del corazón del conductor. Este asiento utiliza seis sensores embebidos con el fin de medir los impulsos eléctricos generados por el corazón. Los sensores utilizados pueden operar independientemente de lo que se encuentre vistiendo el conductor debido a que no requieren contacto directo.

Las pruebas del asiento demostraron que es posible alcanzar un alto grado de precisión en las lecturas de hasta 98 por ciento del tiempo transcurrido detrás del volante, a pesar de encontrarse en etapa de desarrollo. *(News Center Ford Motor Company, 2011)*.

Toyota Motor Corporation e Hino Motors Ltd, desarrollaron un sistema que consta de un alcoholímetro con el fin de detectar la presencia de alcohol y una cámara digital que toma fotografías del rostro del conductor. En el caso de que el resultado de la prueba sea positivo el sistema puede advertir de la situación al conductor o en un caso extremo bloquear la ignición en el automóvil.

El *National Highway Traffic Safety Administration* en Estados Unidos, diseño un sistema para camiones de detección de conductores adormecidos. El mismo hace uso de una cámara que detecta el porcentaje de clausura de los ojos del conductor. Se han generado 12 terabytes de video con 48000 horas de conducción. Durante las pruebas se evaluó a un conductor con anteojos y se descubrió que el sistema no es capaz de realizar diagnósticos fiables si el conductor se encuentra utilizando lentes.

Las principales marcas de automóviles tales como *Volvo, Mercedes Benz y Ford* ya incorporan sistemas de seguridad de cruce de carril o detección de colisión en sus modelos de alta gama, pero sin la posibilidad de obtener dicho sistema de forma individual para adaptarlo a cualquier automóvil. Cada una de las marcas con diferentes particularidades en su sistema:

Ford, incorporando un sistema que determina una posible colisión y tomando el control del vehículo y evitando la colisión.

Mercedes Benz, detecta cuando el vehículo tiende a cambiarse de carril y lo mantiene en este, monitoreando a su vez los automóviles próximos a él y su velocidad.

Volvo, incorpora un sistema de detección de colisión con frenado de emergencia en el cual el vehículo monitorea los vehículos precedentes avisando al conductor y si el mismo no reacciona se lleva a cabo el frenado de emergencia de forma automática.

1.4. Variables en la parametrización del manejo:

Existen diferentes propuestas para la asistencia en el manejo y la detección de fatiga o somnolencia en conductores, en su mayoría prototipos controlados en un simulador, los cuales no resuelven el problema general. La detección de distracción o somnolencia puede realizarse de varias formas:

- Medidas fisiológicas del conductor: Suele ser complejo encontrar un patrón de somnolencia mediante estas medidas y con el inconveniente de ser métodos invasivos.
- Medidas de comportamiento del conductor: mediante las medidas del comportamiento del conductor es posible establecer parámetros de conducción mediante técnicas no invasivas a través de procesamiento de imágenes.
- Medidas de conducción: Referentes a la forma de conducir, parametrizando movimientos y maniobras del vehículo.
- Híbridos: resultan de combinar medidas de conducción referentes al vehículo y analizar el comportamiento del conductor.

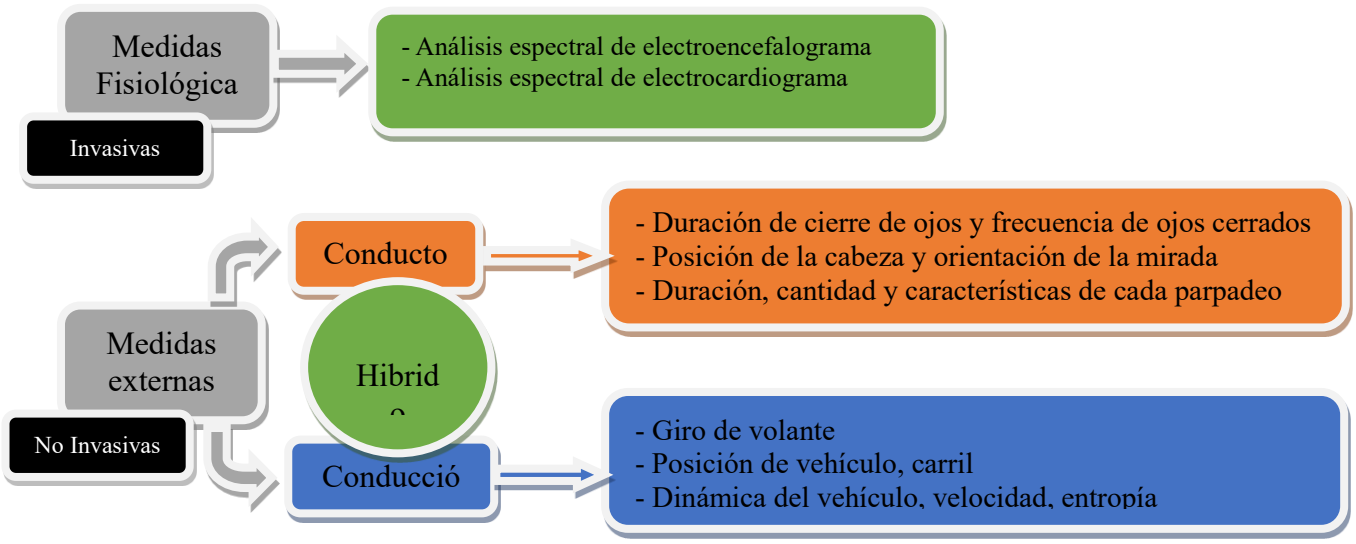

Fig. 2 Parametrización del manejo. Variables posibles a medir

Análisis espectral de electroencefalograma: mediante el análisis espectral de las actividades cerebrales es posible determinar la transición de despierto a dormido con cambios en las bajas frecuencias hasta los 20 Hz. En estado de somnolencia las ondas cerebrales α son reemplazadas por ondas θ.

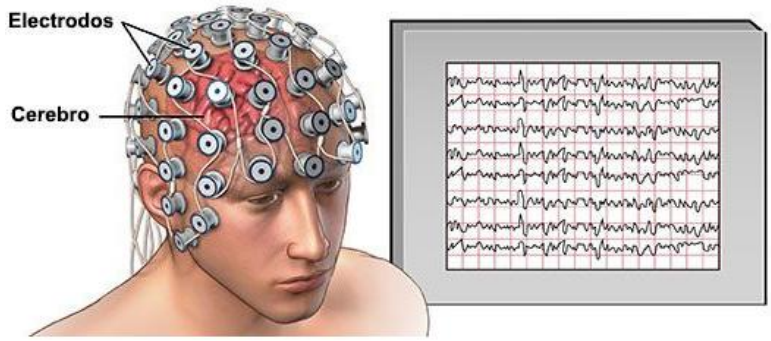

Fig. 3 Parametrización del manejo. Análisis de electroencefalograma

Análisis espectral de electrocardiograma: Mediante el espectro de potencia correspondiente al ritmo cardiaco se determina la correspondencia que existe entre el estado de la persona y su espectro, estableciendo las condiciones fisiológicas del conductor. *(Faber, J, "Detection of different levels of vigilance by EEG", 2004)*

Fig. 4 Parametrización del manejo – Ejemplos de asientos para el análisis de electrocardiograma

Giro de volante: El paso por cero del volante puede ser utilizado para la detección de fatiga o somnolencia. Los movimientos que realiza normalmente el conductor se ven alterados en condiciones de fatiga. Siendo el paso por cero del volante, mayor cuando el conductor no está fatigado, mientras que disminuye cuando existe fatiga. Con somnolencia se detectan movimientos lentos en el volante, de gran amplitud y desviaciones estándares. *(Faber, J, "Detection of different levels of vigilance EEG" 2004)*
Posición de vehículo, cambio de carril: El indicador de desviación de carril se encuentra muy relacionado con el cierre de los ojos del conductor. La desviación de carril instantánea y la desviación estándar de la posición del vehículo son datos útiles para parametrizar el nivel de fatiga o somnolencia. *(T. Ersal, H. J. Fuller, O. Tsimhoni, J. L. Stein & H. K. Fathy, "IEEE Transactions on Intelligent Transportation Systems", 2010)*.

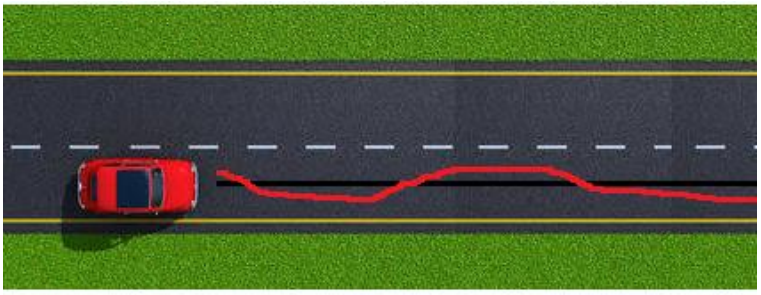

Fig. 5 Parametrización del manejo – Maniobra en estado de somnolencia

Dinámica, velocidad y entropía: Se utiliza la posición del volante, del pedal acelerador, los extremos del carril y la curvatura de la carretera para predecir el estado del conductor. Las señales originales son procesadas mediante algoritmos los cuales calculan parámetros medios, desviaciones y entropía. *(K. Torkkola, N. Massey & C. Wood, "Driver inattention detection through intelligent analysis of readilyavailable sensors", 2004)*.
Duración de cierre de ojos, posición de la cabeza, características de parpadeo: Estos patrones son analizados mediante técnicas de procesamiento de imagen, no intrusivas y monitorizan la somnolencia del conductor mediante el análisis de las imágenes capturadas por las cámaras instaladas en el vehículo. La somnolencia se puede parametrizar de forma efectiva mediante la apariencia de la cara, actividad de la cabeza y los ojos. La duración del cierre de los ojos, la frecuencia de parpadeo, y la mirada son utilizados para medir la fatiga y somnolencia. *(L. Bretzner & M. Krantz, "Vehicular Electronics and Safety", IEEE International Conference 2005)*.

1.5. Objetivos del proyecto

El objetivo de este proyecto es mitigar los errores humanos que se pueden generar en la conducción de vehículos, ya sea, en automóviles, vehículos de transporte de cargas o bien trasporte público (colectivos, trenes). A fin de disminuir el riesgo de sufrir accidentes de tránsito, los cuales, en su mayoría se producen por descuidos, errores del conductor, somnolencia o uso de teléfono móvil. Dicho objetivo se pretende lograr mediante el uso de herramientas de procesamiento de imágenes en tiempo real para monitorear las acciones del conductor, alertando en caso de detectar distracciones o somnolencia. Dicho sistema contara con una cámara encargada de detectar la atención del conductor y la apertura de los ojos, las imágenes tomadas por la cámara son procesadas por una placa computadora de bajo costo y alto poder de procesamiento. El software de análisis procesa cada una de las imágenes enviadas por la cámara y realiza la detección de parpadeo, la cual alerta al conductor si el mismo aparenta estar en estado de somnolencia, o apartase la vista del camino. La forma de alarmar al conductor en situación de somnolencia se realiza a través de una alerta por voz, a modo de evitar reacciones bruscas. La placa computadora enviara una señal de audio por voz, cuando el conductor apartara la vista por un determinado tiempo y el sistema detecte que el vehículo se encuentre en movimiento. La velocidad del vehículo, medida desde un módulo GPS, determinará el nivel de

sensibilidad que tendrá el sistema. Es decir, a mayor velocidad del vehículo, al detectarse uno de los eventos, el conductor será alertado en forma más rápida.

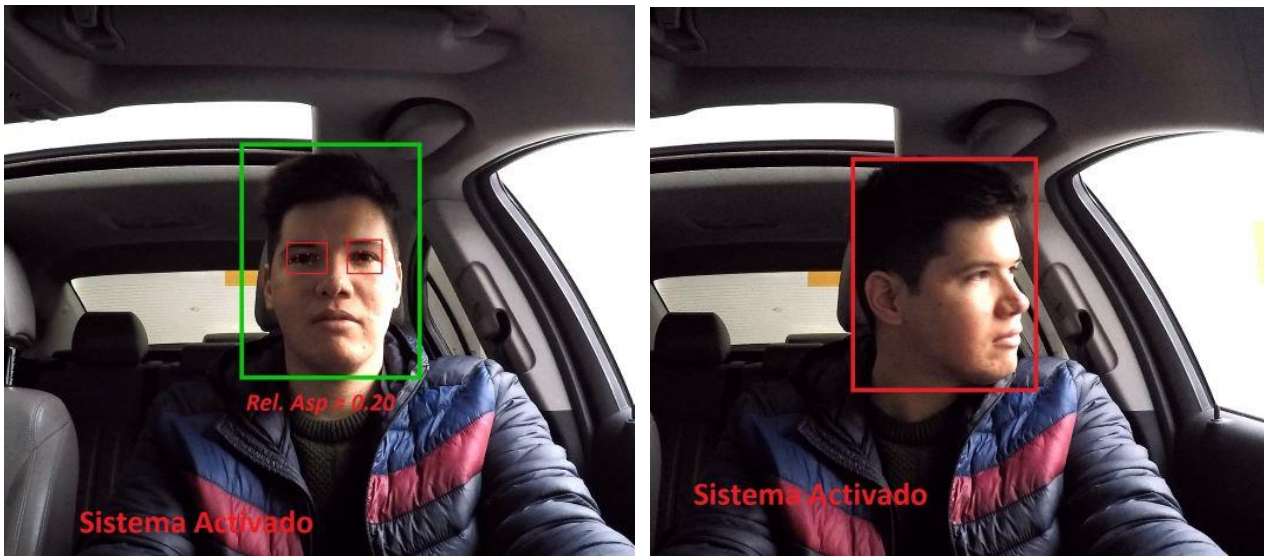

Fig. 6 Fotograma. Activación y desactivación del sistema según la velocidad del vehículo

El dispositivo que se proyecta no toma el control del vehículo, sino que funciona como alerta al conductor. A su vez gracias a la portabilidad del sistema el mismo puede funcionar en vehículos de cualquier marca, modelo y gama, posibilitando un mercado bastante amplio, automovilistas en general, empresas de viajes de larga distancia, empresas de transporte de carga.

2. Desarrollo

2.1. Software y herramientas utilizadas

2.1.1. OpenCV

La compañía Intel desarrollo OpenCV, dicha herramienta es una biblioteca libre de visión por computador, apareció su primera versión alfa en el mes de enero de 1999. Busca como objetivo simular la capacidad visual humana mediante sistemas de adquisición y dispositivos de cómputo.
Desde el comienzo de su aparición se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos.
La librería está escrita en los lenguajes C y C++ y es compatible con Linux, Windows y Mac OS X. Cuenta con un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes.
Se puede destacar de esta librería su eficiencia en cuanto a gasto de recursos computacionales y con un enfoque hacia las aplicaciones de tiempo real.
OpenCV contiene aproximadamente 500 funciones que abarcan muchas áreas de visión por computador, incluyendo inspección de productos de fábricas, escaneo médico, seguridad, interfaces de usuario, calibración de cámaras, robótica, y muchas más.
Principalmente cuenta con herramientas para la detección de rostros y ojos, motivo por el cual se llega a la elección de dicha librería. También tiene una completa librería de uso general de aprendizaje automático (MLL o Machine Learning Library), la cual es muy útil para cualquier problema de aprendizaje automático, herramienta muy útil en la actualidad.

2.1.2. Lenguaje Python

Python es un lenguaje de programación interpretado, interactivo, orientado a objetos y de alto nivel. Fue creado por Guido van Rossum durante 1985-1990. El código fuente de Python también está disponible bajo la Licencia Pública General de GNU.
Python tiene como cualidad la de ser un lenguaje fácil de leer. Su formato está visualmente despejado, y a menudo usa palabras clave en inglés donde otros idiomas usan la puntuación.
A diferencia de muchos otros lenguajes, no usa corchetes para delimitar bloques, y punto y coma después de las declaraciones son opcionales. Tiene menos excepciones sintácticas y casos especiales que C o Pascal.
Desde 2003, Python se ha clasificado constantemente entre los diez idiomas de programación más populares en el Índice de comunidad de programación TIOBE, donde, a partir de enero de 2018, es el cuarto idioma más popular (por detrás de Java, C y C++).
Por este motivo y su gran facilidad se decide realizar el proyecto en dicho lenguaje.
Python se ha utilizado en proyectos de inteligencia artificial. El proyecto de computadora de placa única Raspberry Pi ha adoptado Python como su lenguaje de programación de usuario principal, otro motivo principal que llevo a la elección de dicho lenguaje.

2.1.3. Herramienta PyQt

PyQt5 es un módulo que se puede usar para crear interfaces gráficas de usuario (GUI). Es un conjunto completo de enlaces de Python para Qt v5. Tiene más de 620 clases y 6000 funciones y métodos. Se implementa con más de 35 módulos de extensión y permite que Python se utilice como un lenguaje de desarrollo de aplicaciones con interfaz de usuario alternativo para C ++ en todas las plataformas compatibles, incluidas iOS y Android.

2.1.4. Detección de rostros

Cuando se quiere detectar un objeto específico dentro de una imagen tomada como información de entrada, la cual debe ser analizar, existen distintos tipos de modelos o técnicas de procesamiento de imágenes para poder llevar a cabo esta tarea.
En términos generales, se puede clasificar en dos familias de técnicas de reconocimiento facial: técnicas basadas en la apariencia y técnicas basadas en modelos. En el siguiente esquema se visualiza una clasificación más detallada de los distintos tipos de detección facial que hay en la actualidad.

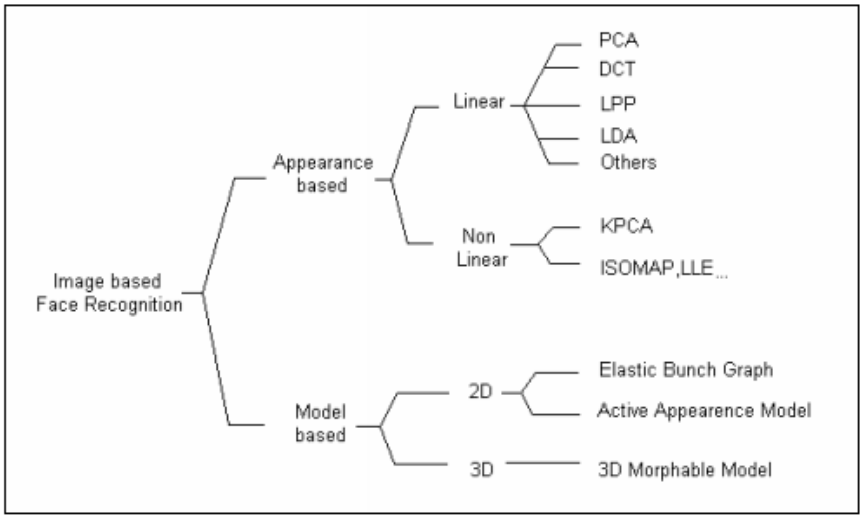

Fig. 7 Reconocimiento facial. Metodologías utilizadas

Puede decirse como principal cualidad de las técnicas basadas en apariencia, que analizan la textura de la imagen a partir de la cual se aplicarán diferentes técnicas estadísticas y se extraerá la información. En cambio, la gran diferencia con las técnicas basadas en modelos, es que estas extraen las características tanto de la forma del rostro como de la textura.
Estas técnicas basadas en apariencia transforman el problema de reconocimiento facial en un problema de análisis de espacio donde se pueden aplicar diferentes técnicas estadísticas. Se concluye entonces que, para este tipo de métodos o técnica, se destaca su aplicabilidad en imágenes de baja resolución o mala calidad, su baja complejidad, y su rapidez de ejecución ya que se pueden utilizar en sistemas en tiempo real. Sin embargo, también cuentan con algunos inconvenientes. Uno de los principales es que para conseguir buenos resultados se requiere un conjunto de muestras considerable para la fase de entrenamiento. Además, este tipo de métodos cuenta con dificultades a los cambios en la iluminación, la pose o la expresión de la cara. Esto tienen un gran impacto en los resultados finales.
Las técnicas basadas en modelos, buscan obtener cualidades o características biométricas de las imágenes para llevar a cabo el reconocimiento. Se tienen en cuenta aspectos como la distancia entre los ojos, el grosor de la nariz, el tamaño de la boca, etc. Como diferencia de estos sistemas con el anterior, estos requieren un conocimiento previo de las imágenes, son mucho más lentos y complejos. Sin embargo, cuando se producen cambios de orientación o expresión de la cara estos sistemas son más robustos. Por lo tanto, se ven menos afectados por cambios en la iluminación o las sombras.
Dependiendo del método empleado, estos inconvenientes tendrán un impacto mayor o menor. En este proyecto, se utilizó el método basado en apariencia debido a las limitaciones del hardware.

2.1.5. Detector Haar

El algoritmo de Viola-Jones es un método de detección de objetos que sobresale por su bajo coste computacional, lo que facilita su utilización en sistemas que requieren un procesamiento en tiempo real. La principal motivación para su desarrollo fue la problemática de la detección de caras, donde sigue siendo ampliamente utilizado en la actualidad. Aun así, puede ser utilizado para otras clases de objetos que, al igual que las caras, estén caracterizados por patrones típicos de iluminación.
El algoritmo se basa en una serie de clasificadores débiles denominados Haar-like features que se pueden calcular eficientemente a partir de una imagen integral.
Las Haar-like features son los elementos básicos con los que se realiza la detección. Estos clasificadores son características o cualidades muy simples que se identifican en las imágenes y que consisten en la diferencia de intensidades luminosas entre regiones rectangulares adyacentes. Las características quedan por tanto definidas por unos rectángulos y su posición relativa a la ventana de búsqueda y adquieren un valor numérico resultado de la comparación que evalúan.
En el desarrollo presentado por Viola-Jones existen tres tipos de características, estas se visualizan en la siguiente figura.

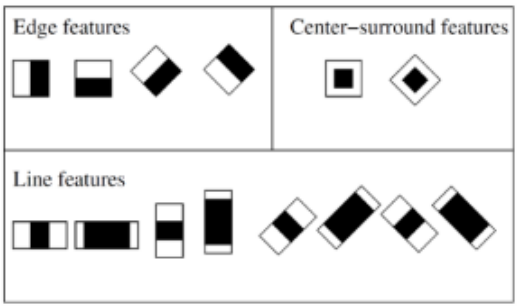

Fig. 8 Detector Haar. Ejemplo de características Haar

Este algoritmo considera regiones rectangulares como las que se describieron anteriormente en una determinada ventana de detección, suma las intensidades de los píxeles en cada región y calcula la diferencia entre estas sumas. En la siguiente figura se aprecian algunos ejemplos de características Haar comunes de un detector de caras.

Fig. 9 Detector Haar. Ejemplo de características Haar habituales en rostros

Este sistema utiliza el aprendizaje automático para el entrenamiento de la función de cascada, donde se realiza esto a partir de muchas imágenes positivas y negativas. Una vez finalizada la cascada, se puede utilizar esta función para detectar el objeto para la cual fue entrenada en otras imágenes.

Para hacer posible esto, se aplica todas y cada una de las características Haar en las imágenes que se adjuntan al sistema como imágenes de entrenamiento. Mediante el aprendizaje automático se establece para cada característica, el mejor umbral que clasificará las caras en positivas y negativas. Obviamente, habrá errores o clasificaciones erróneas. Se seleccionan las características con un índice de error mínimo, o sea con mayor precisión de clasificación de las imágenes en las que constituyen un rostro y las que no. El sistema continúa analizando hasta que se logre la precisión buscada o la tasa de error o se encuentra el número requerido de características.
El clasificador final es una suma ponderada de estos clasificadores débiles. Se llama débil porque solo no puede clasificar la imagen, pero junto con otros, forma un clasificador fuerte. Por ejemplo, para un clasificador con alrededor de 6000 características, con tan solo 200 de ellas se puede proporcionar una detección con el 95% de precisión.
En una imagen a procesar, la mayor parte de la misma en muchos casos es una región sin rostro. Si se lograr verificar si una ventada de procesamiento es o no una región de una cara, se puede desechar esta zona en cuestión de una sola vez y no lo volver a procesarla. Para así, concentrarse en las regiones donde puede haber un rostro y de esta forma, se dedica más tiempo a verificar posibles regiones faciales y a no perder tiempo de procesamiento en regiones falsas.
Para esto, se introdujo el concepto de cascada de clasificadores. Básicamente, en vez de aplicar todas las 6000 características en una ventana, las características se agrupan en diferentes etapas de clasificadores y se aplican una a una. Si una ventana falla la primera etapa, se descarta del procesamiento y no se considera las características restantes. La ventana que pasa todas las etapas es una región de la cara.

2.1.6. Dlib

Dlib es un moderno kit de herramientas de C++, que contiene algoritmos de aprendizaje automático y herramientas para crear software complejo para resolver problemas del mundo real. Tiene un uso amplio en la industria como en el educativo, con una gran variedad de otras áreas, como la robótica, los dispositivos integrados, los teléfonos móviles y los grandes entornos informáticos de alto rendimiento. Esta herramienta cuenta con licencia de código abierto que posibilita su utilización en cualquier aplicación o proyecto, sin costo alguno.
Dlib cuenta con un kit llamado SVM lineal basado en un histograma de Gradiente Orientado (HOG) para crear un detector de objetos, con herramientas de aprendizaje profundo más potentes (pero más lentas). Por este motivo fundamental se ha decidido su utilización en este proyecto.
El desafío más importante al construir un clasificador es que el mismo funcione de manera correcta ante variaciones en la iluminación, postura y oclusiones en la imagen. Para ello, se forma lo que comúnmente es llamado como vector de características. Donde el mismo, captura información que es útil para la clasificación, pero es invariable a pequeños cambios en la iluminación y las oclusiones.
La tecnología HOG es ampliamente utilizada para la detección de objetos, ya que es invariante a los cambios de forma e iluminación, por lo tanto, es una buena opción para llevar a cabo este proyecto.

La idea de HOG es que, en lugar de utilizar cada dirección de gradiente individual de cada píxel individual de una imagen, agrupamos los píxeles en celdas pequeñas. Para cada celda, calculamos todas las direcciones de gradiente y las agrupamos en varias ubicaciones de orientación. Sumamos la magnitud del gradiente en cada muestra. Por lo tanto, gradientes más fuertes contribuyen a aumentar el peso de sus contenedores, y se reducen los efectos de pequeñas orientaciones aleatorias debidas al ruido. Este

histograma nos da una idea de la orientación dominante de esa celda. Hacer esto para todas las celdas nos da una representación de la estructura de la imagen.

2.2. Detección de ojos y parpadeo

Para el caso de la detección de los ojos, se puede también utilizar los clasificadores Haar o la herramienta Dlib. Si la detección se hace con cualquiera de estas dos metodologías, se detectan tanto ojos abiertos como ojos cerrados, teniendo una mayor efectividad a la hora de encontrar ojos abiertos que cerrados, pero pese a ello, no se puede establecer con solo esto si los ojos se encuentran cerrados o no y por lo tanto determinar el estado de somnolencia del conductor.

Cualquier ojo tiene dos características principales que lo definen, el color y la forma. El color de los ojos es diferente en cada persona, pero todos ellos tienen una característica común que es el color blanco del globo ocular. En cuanto a la forma, las personas pueden tener diferentes tamaños de ojos, pero todos ellos tienen una forma elíptica muy próxima a un círculo, en función de cada persona los ojos serán más o menos circulares. Son estas dos cualidades de los ojos las que se utilizan para realizar en análisis y la detección de los parpadeos.

En cuanto al color, un ojo abierto contiene una mayor densidad de píxeles blancos que un ojo cerrado, esto es por el color blanco del globo ocular como antes se mencionó. Por lo tanto, es posible establecer un valor umbral de píxeles blancos, si el valor procesado de la imagen se encuentra por encima del umbral entonces el ojo se establece como abierto y por debajo el ojo permanece cerrado. Primeramente, en este proyecto se utilizó este método, pero no se lograron buenos resultados y por lo tanto fue descartado. El principal problema que tenía este método se debía a la obtención del valor umbral de la imagen en escala de grises.

Para el método que se basa en la forma de los ojos, se puede realizar esta medida mediante funciones de OpenCV utilizando la función houghcircles. Lo que hace esta función básicamente es encontrar en la imagen formas que se asemejan a círculos. Esto implica que la aplicación detectará a parte de los círculos que haya en la imagen, posibles falsos positivos que tengan una región de su forma similar a un círculo. Por este motivo este método también tiene sus desventajas.

Para este proyecto se utilizó el método basado en la forma del ojo para determinar el parpadeo, para ello se usó la herramienta Dlib que devuelve datos que hacen posible el análisis de la apertura de los ojos, esto se explicara más adelante con detalles.

2.3. Diagrama en bloques del hardware

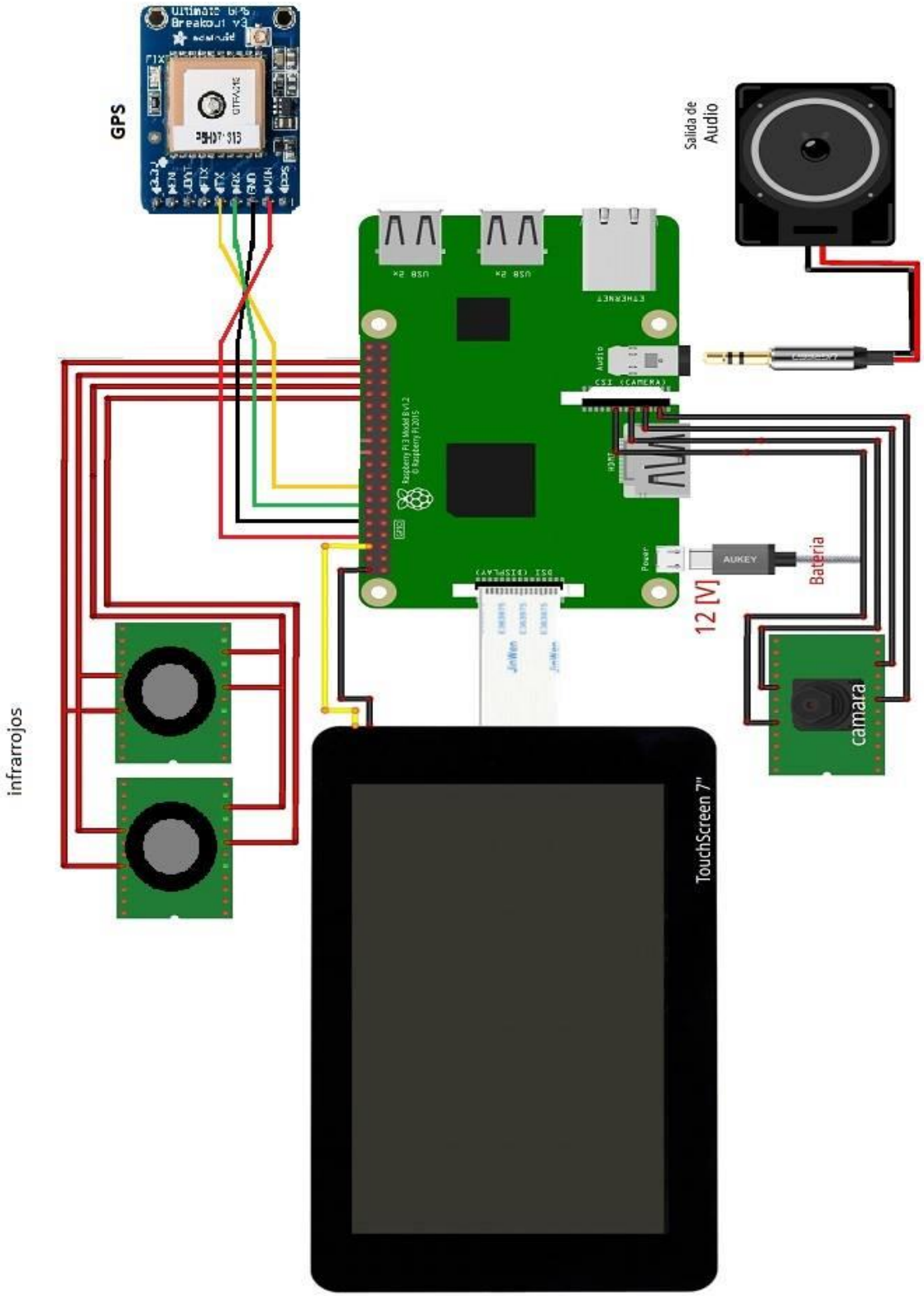

Fig. 10 Diagrama en bloques. Disposición y componentes que integran el sistema

2.4. Descripción general

Si se observa en el diagrama mostrado anteriormente, el sistema cuenta con una placa de desarrollo que tiene como funcionalidad ser el cerebro de todo el equipo. El procesamiento y análisis de las imágenes, la interfaz gráfica y el manejo de los periféricos de salida son controlados por dicha CPU.
El sistema captura la imagen mediante la cámara, en casos de visión nocturna o utilización de lentes de sol del conductor se ilumina con luz infrarroja. Esta captura es procesada y analizada mediante el software de detección, si detecta un estado de somnolencia de la persona al volante entonces activa una alarma sonora y visual.
El equipo cuenta con distintos tipos de mensajes de alarmas, esto va depender si se detecta somnolencia o si el conductor se encuentra distraído mirando hacia los costados del camino.
El sistema tiene como fin, ser un equipo de seguridad para la conducción en rutas. Por lo tanto, la detección de somnolencia y distracción del conductor solo estará en funcionamiento cuando el vehículo transite a una velocidad mayor a los 40 Km/h. Con este fin es que se utiliza un módulo GPS en el sistema.
Debido a que no es lo mismo cerrar los ojos o distraerse un segundo cuando se viaja a 40 Km/h que a 140 Km/h, el tiempo de tolerancia antes de activarse la alarma correspondiente va a ser directamente proporcional a la velocidad con la que circula el vehículo. Este análisis también es posible realizarse gracias a los datos obtenidos desde el módulo GPS.
Por último, el aparato cuanta con una pantalla de 7” en la cual se puede visualizar todas las alarmas activadas en el viaje.

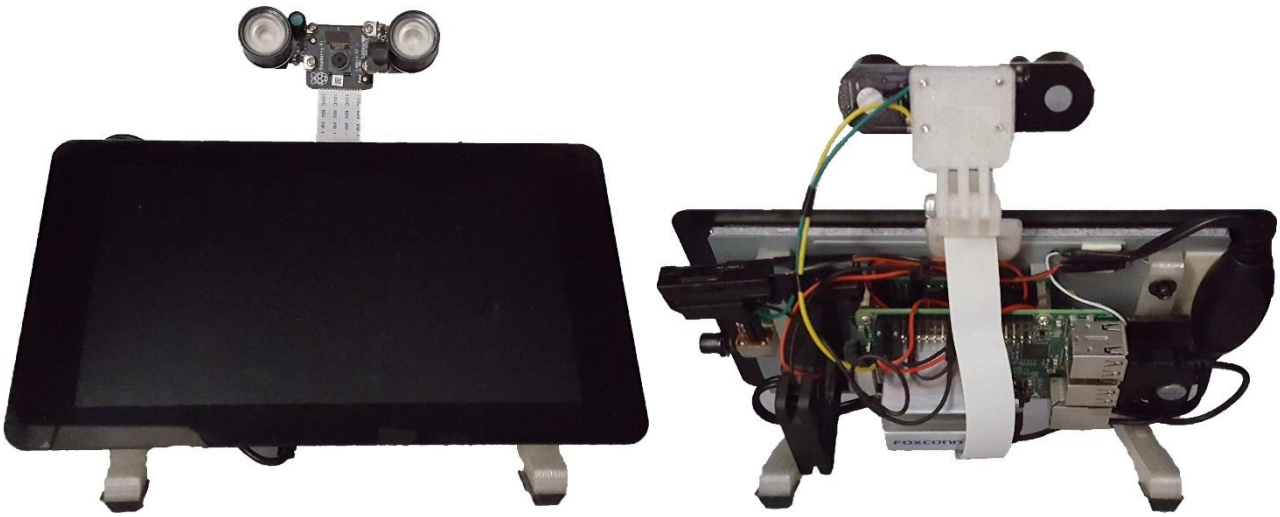

Fig. 11 Hardware. Vista frontal y posterior del sistema

2.4.1. Placa CPU

La unidad que desempeña de CPU en el sistema, es una placa de desarrollo Raspberry Pi 3. Se hizo un análisis en relación a precio y prestaciones de las distintas placas que hay en el mercado y la misma es la más conveniente para los recursos requeridos. Teniendo la Raspberry más allá de sus limitaciones un hardware capaz de realizar las tareas de procesamiento necesarias a una performance más que aceptable y a un bajo costo económico.
También un motivo de elección de esta placa, fue la gran cantidad de documentación disponible en infinidad de tutoriales en internet.

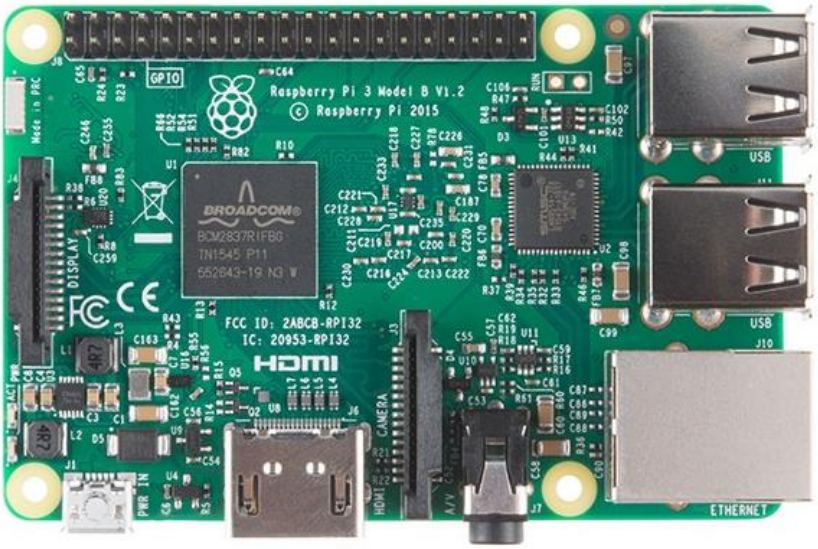

Fig. 12 CPU del sistema. Placa Raspberry Pi 3

La Raspberry Pi 3 ensambla en su circuito un chipset Broadcom BCM2387 con cuatro núcleos ARM Cortex-A53 a 1.2 GHz. Dicho procesador es capaz de mover con soltura videojuegos y aplicaciones, además de disponer de una gran potencia de procesado para otros tipos de tareas. La GPU encargada de los gráficos es la Broadcom VideoCore IV, una solución Dual Core compatible con Open GL ES 2.0 y OpenGV que permite llegar a resoluciones Full HD con soltura.
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. Otra razón fundamental por la cual se eligió utilizar esta placa, es el soporte que tiene con muchas de las herramientas que son necesarias para este proyecto y la existencia de documentación sobro el uso de las mismas.

2.4.2. Cámara

La opción elegida para realizar la captura de la imagen a procesar fue en este caso la cámara diseñada específicamente para la placa de desarrollo Raspberry Pi 3.

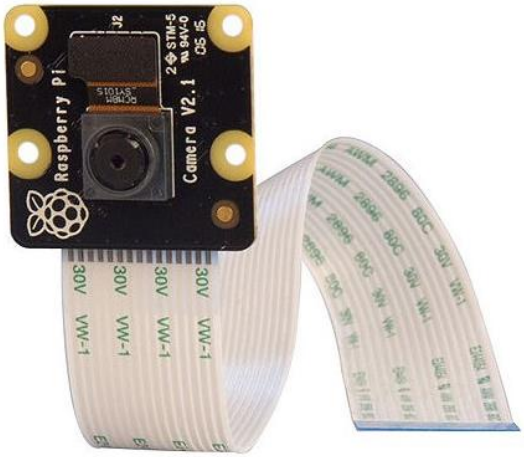

Fig. 13 Cámara del sistema. Módulo de Raspberry Pi V2.1

Por cuestiones de precio y calidad, esta fue la mejor solución para efectuar esta tarea. La cámara Raspberry Pi se conecta de forma directa al conector CSI en la Raspberry Pi. La

misma es capaz de entregar una imagen con una resolución de 8MP, o una grabación de video HD de 1080p a 30 fps. Este módulo tiene un sensor Sony IMX219, que como se dijo anteriormente es de 8MP en un módulo de enfoque fijo.
Esta plaquita se conecta por medio de un cable de cinta de 15 pines, a la interfaz serie de la cámara MIPI (CSI) de 15 pines, que fue diseñada especialmente para interactuar con cámaras.
Otra ventaja además del precio y la calidad, es el tamaño del módulo. El mismo es pequeño, alrededor de 25 mm x 20 mm x 9 mm, y pesa poco más de 3 g, lo que lo hace perfecto para aplicaciones móviles u otras donde el tamaño y el peso son importantes.
En cuanto a las imágenes fijas que esta cámara puede entregar, la misma es capaz de imágenes estáticas de 3280 × 2464 píxeles, y también admite 1080p a 30fps, 720p a 60fps y 640x480p grabación de video de 60/90.

2.4.3. Infrarrojo

Para este apartado se seleccionó también al igual que el caso anterior, un módulo ya diseñado específicamente para la placa de desarrollo Raspberry Pi 3. El infrarrojo tiene como funcionalidad en este sistema el permitir que el mismo pueda seguir con una visualización clara del conductor en casos de oscuridad o una luminosidad mediocre del ambiente.

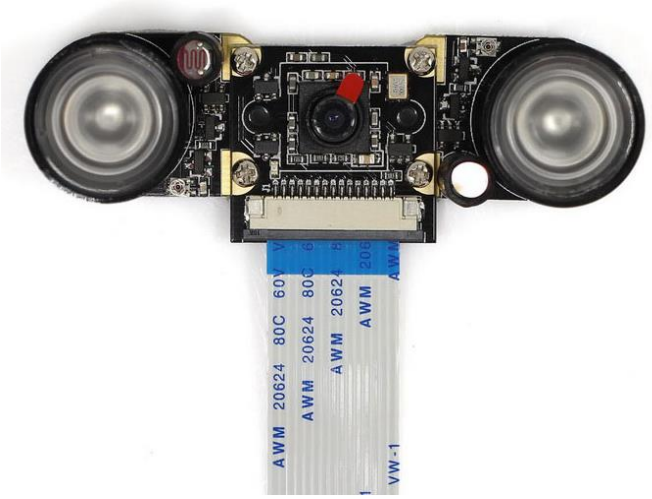

Fig. 14 Infrarrojo. Módulo de Raspberry Pi 3 utilizado en el sistema

Las características de este módulo de led infrarrojo son:

- LED infrarrojo 850nm
- Potencia máxima de 1W
- Fotorresistor, con el cual se detecta cuando activar el modulo
- Resistencia ajustable, para controlar el umbral de luz ambiental con el cual se debe activar los leds.

2.4.4. Salida de audio

Los modelos de la Raspberry Pi B +, Pi 2 y Pi 3 traen incorporado un conector Jack de 3,5mm de 4 polos que además incluye la señal de video compuesta.
Este nuevo conector es un zócalo de 4 polos que transmite señales de audio y video como se comentó anteriormente, el mismo es similar a los enchufes que se encuentran en otros dispositivos multimedia, como iPod, reproductores de MP3 y teléfonos inteligentes.
Este tipo de conector se lo conoce comúnmente como "TRRS". Donde los cuatro conductores del mismo llevan video, audio izquierdo, audio derecho y tierra.

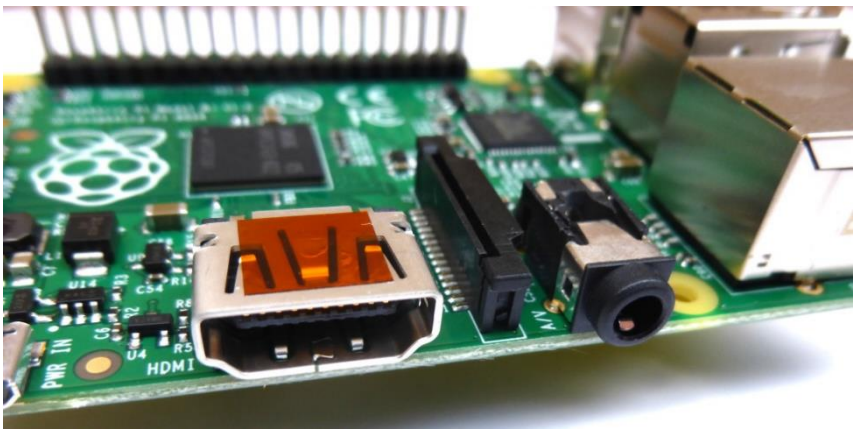

Fig. 15 Placa Raspberry. Salida de audio con conector Jack 3.5mm

2.4.5. Modulo GPS

El módulo GPS también llamado Sistema de Posicionamiento Global se usa para detectar la Latitud y la Longitud de cualquier ubicación en la Tierra, con la hora UTC exacta (Tiempo Universal Coordinado). Este dispositivo recibe las coordenadas del satélite para cada segundo, con hora y fecha.
El módulo GPS envía los datos relacionados con la posición de seguimiento en tiempo real en formato NMEA. El formato NMEA está integrado por caracteres, comenzando con $ GPGGA, luego contiene las coordenadas, el tiempo y otra información útil.
Podemos extraer las coordenadas de la cadena $ GPGGA contando las comas en la cadena. El ejemplo siguiente muestra los caracteres enviados por el módulo:

```
$GPGGA,104534.000,7791.0381,N,06727.4434,E,1,08,0.9,510.4,M,43.9, M,,* 47
```

La cadena de caracteres está formada por $ GPGGA, HHMMSS.SSS, latitud, N, longitud, E, FQ, NOS, HDP, altitud, M, altura, M ,, datos de suma de comprobación.

Fig. 16 GPS del equipo. Módulo de GPS para Raspberry

2.4.6. Pantalla

La pantalla tiene una resolución de 800 x 480. Se conecta a través de una placa adaptadora que maneja la potencia y la conversión de señales, requiere dos conexiones a la placa principal, energía del puerto GPIO del Pi y un cable plano que se conecta al puerto DSI presente en todos los Raspberry Pi. Los controladores de pantalla táctil permiten hasta 10 puntos táctiles capacitivos.

Fig. 17 Interfaz del sistema. Pantalla y driver utilizados en el proyecto

2.4.7. Sistema Operativo

El sistema operativo elegido para el desarrollo está basado en Debian, una de las distribuciones GNU/Linux más importantes. Se caracteriza por un proceso de desarrollo muy prolongado y con altos estándares de calidad para ofrecer un producto excelente, no en vano se considera el sistema operativo más estable del mundo, aunque ello sea a costa de incluir versiones más anticuadas de los paquetes que muchas otras distribuciones. Se considera además un sistema operativo universal ya que puede funcionar en casi todas las arquitecturas. Cuenta con el kernel Linux 4.9 LTS, y un conjunto de paquetes y herramientas de programación y desarrollo.

Fig. 18 Sistema operativo. Distribución GNU/Linux basado en Debian

Luego de instalar el sistema operativo, es necesario cargar las librerías y herramientas necesarias para desarrollar el software de visión artificial dentro de la placa.
El código se desarrolla en Python3, antes debe ser descargado e instalado.

```
$ sudo apt-get install python3-dev
```

La biblioteca OpenCV viene con un submódulo llamado Highgui que se utiliza para mostrar imágenes en pantalla y compilar interfaces básicas. Para compilar este módulo es necesario instalar la biblioteca de desarrollo GTK.
Para instalar OpenCV, se descargar el mismo de su repositorio oficial y luego se compila con CMake. Finalizada la compilación se procede a la instalación.

```
$  cd ~/opencv-3.1.0/
$  mkdir build
$  cd build
$  cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \
    -D BUILD_EXAMPLES=ON
$  make
$  sudo make install
```

Para el desarrollo de la interfaz gráfica se debe instalar PyQT5, esta es una herramienta de la librería grafica de Qt, para el desarrollo de interfaces en lenguaje Python.
Se deben descargar además de las herramientas, las distintas librerías utilizadas para el desarrollo de la interfaz gráfica y sistema de audio.
La cámara luego de ser instalada debe habilitarse desde la configuración.

```
$  sudo raspi-config
```

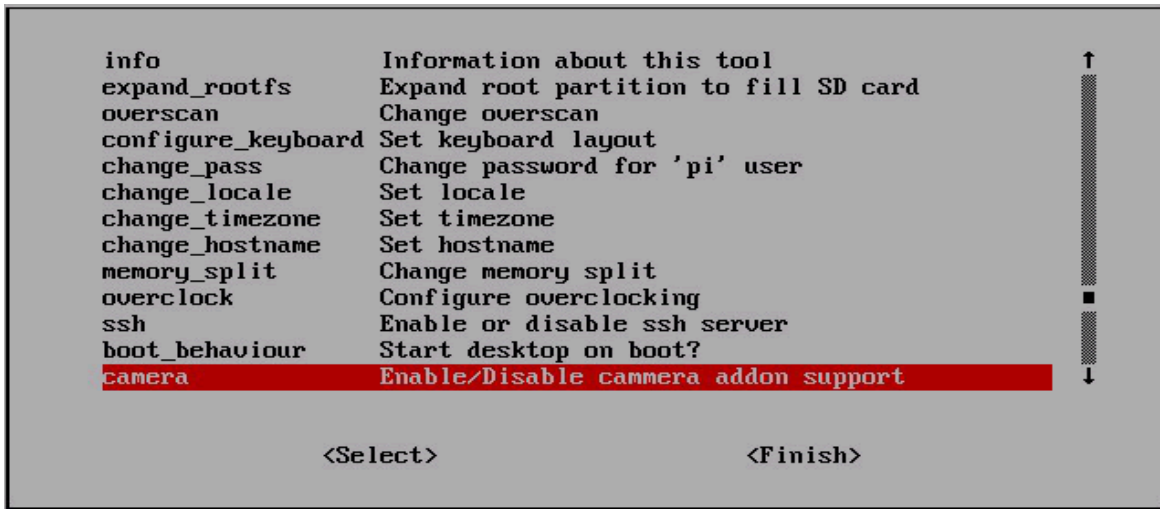

Fig. 19 Configuraciones del sistema. Habilitación de la cámara

Luego se instala y habilita el sistema de audio por salida jack 3.5, dentro de las configuraciones en opciones avanzadas.

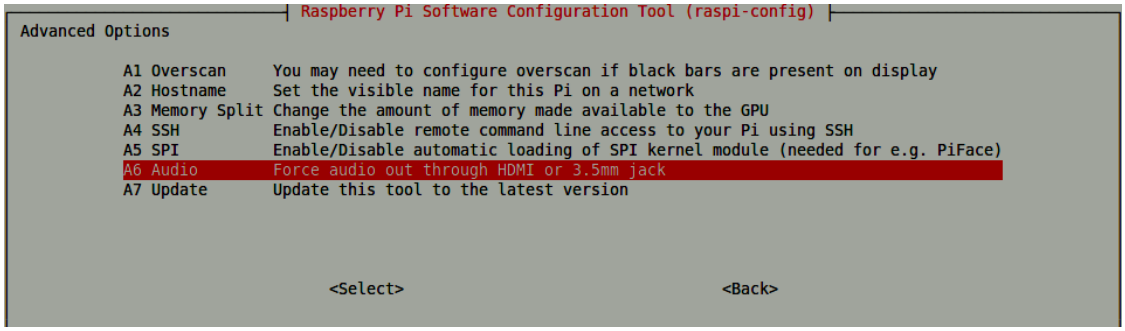

Fig. 20 Configuraciones del sistema. Habilitación sistema de audio

2.5. Software

2.5.1. Diagrama de flujo

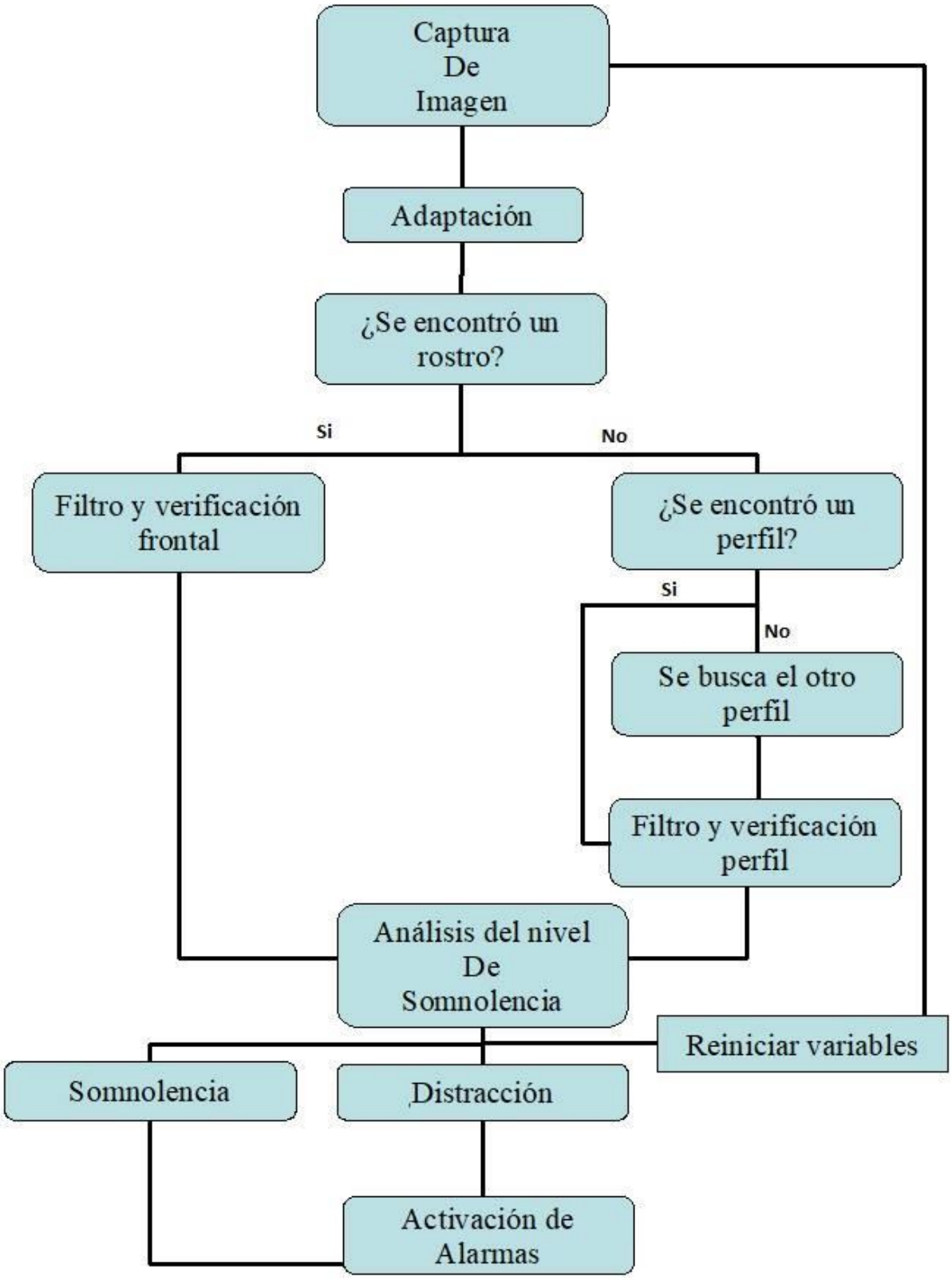

Fig. 21 Diagrama de flujo. Lógica y análisis del software

2.5.2. Adquisición de imagen

Es importante antes de comenzar con todo el procesamiento y análisis de las imágenes, cargar las librerías necesarias para las distintas acciones que se deben realizar a lo largo del software.

```
import numpy as np
import cv2
from imutils import face_utils
from imutils.video import VideoStream
import imutils
```

Si se habla de la visión artificial, las herramientas para lograrlo vienen incluidas en la librería de OpenCV, denominada en el código como cv2.
Numpy, es un kit que permite operaciones con matrices. Por lo tanto, al representar una imagen como tal, es posible efectuar distintas operaciones con ella.
En el caso de Imutils, la misma es una serie de funciones de conveniencia para realizar operaciones básicas de procesamiento de imágenes como traducción, rotación, cambio de tamaño y visualización de imágenes más fácil con OpenCV y Python. Además, la biblioteca también se ocupa de la manipulación de diferentes tipos de cámaras. Entonces, lo que se tiene que hacer es inicializar la transmisión de video con la cámara y el tamaño de fotograma que queremos usar.
Una vez incluido todas las bibliotecas necesarias, lo primero que debe hacer el software es la captura de la imagen del conductor para determinar el estado de somnolencia del mismo. Para llevar a cabo esta tarea, se utiliza las siguientes líneas de código.

```
vs = VideoStream(usePiCamera=True).start()
    time.sleep(1.0)
    while (True):
            frame = vs.read()
            frame = imutils.resize(frame, width=1000)
```

Se puede apreciar que lo primero es inicializar la cámara a utilizar, para luego leer sus datos y darle la resolución apropiada.
Una vez inicializada la cámara, el sistema entra en un bucle infinito donde va estar constantemente haciendo captura de imágenes. Pero como se comentó anteriormente, el procesamiento solo se efectúa si el vehículo circula a una velocidad mayor a los 40 Km/h. Por lo tanto, solo se activarán las alarmas cuando se cumpla esta condición de velocidad.

2.5.3. Carga de clasificadores

Una vez capturada la imagen del conductor, el siguiente paso es analizar la misma para poder detectar la posición de la cara con exactitud. Se vio anteriormente que para realizar esta operación existen dos herramientas disponibles en el sistema, ellas son clasificadores Haar y Dlib. La herramienta Dlib es muy precisa, pero requiere mayor procesamiento, solicitando un tiempo necesario más elevado lo que hace dificultoso su utilización en tiempo real. Para solucionar esto, se decidió utilizar el método Haar como primera medida para detectar donde se encuentra localizado el rostro, una vez obtenido esto se procede a utilizar la biblioteca Dlib para el análisis de somnolencia.
Simplemente se trata de crear un objeto al cual se le asigna un archivo, este puede ser dado mediante su ruta de ubicación o simplemente especificando su nombre en el caso de que el mismo se encuentre en la carpeta del proyecto.

Este archivo asignado, es el clasificador en cuestión. Si se requiere detectar una cierta cantidad de objetos diferentes, para la detección de cada uno de ellos se necesitará un clasificador distinto.
En el siguiente código se puede apreciar la carga de los mismos.

```
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
detector_perfil = cv2.CascadeClassifier('haarcascade_profileface.xml')
```

En este proyecto se utilizan dos clasificadores Haar, uno para poder detectar el rostro frontal el cual va ser el más importante o con mayor relevancia y otro para la detección del perfil para determinar si el conductor se encuentra distraído.

2.5.4. Detección rostro frontal

En esta etapa solo queda detectar el rostro frontal del conductor utilizando la función detectMultiScale, la cual es un atributo del objeto creado a la hora de cargar el clasificador.
Para utilizar esta función, se debe pasar como argumento la imagen que se quiere procesar en escala de grises. Esto se logra mediante el comando cvtColor perteneciente a la librería de OpenCV.
El código utilizado para dicha tarea es el siguiente:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
rects = detector.detectMultiScale(gray, scaleFactor=1.1,
                                  minNeighbors=5, minSize=(100, 100),
                                  flags=cv2.CASCADE_SCALE_IMAGE)
```

Se puede apreciar que, para la detección la función solicita varios argumentos además de la imagen. Los valores de estos argumentos son muy importantes ya que los mismos establecen las características de búsqueda.
Estos parámetros son los siguientes:

- Imagen: la primera entrada y la más importante es la imagen a procesar. La cual debe estar en escala de grises sí o sí.
- scaleFactor: Esta función compensa una percepción falsa de tamaño que ocurre cuando una cara parece ser más grande que la otra simplemente porque está más cerca de la cámara.
- MinNeighbors: este es un algoritmo de detección que usa una ventana móvil para detectar objetos, lo hace al definir cuántos objetos se encuentran cerca del actual antes de que pueda declarar el rostro encontrado.
- minSize: Es el tamaño mínimo de detección. Todos los objetos que se detecten con un tamaño inferior a este parámetro serán ignorados.
- maxSize: En este caso se establece el tamaño máximo que debe tener el objeto que se quiere detectar.
- flags: Son indicadores en el método de detección para que los desarrolladores tengan más control sobre la potencia de procesamiento frente a la precisión. Por ejemplo, SCALE_IMAGE para cada factor de escala utilizado, la función reducirá la escala de la imagen en lugar de "acercar" las coordenadas de la función. FIND_BIGGEST_OBJECT, si está configurado, la función encuentra el objeto más grande en la imagen. Es decir, la secuencia de salida contendrá uno o cero elementos.

Una vez ejecutada esta función, se puede leer el tamaño de la variable para conocer si se han encontrado objetos en la imagen procesada. Si el tamaña es mayor a cero entonces la búsqueda fue exitosa.

Los objetos detectados se devuelven como una lista de rectángulos. Donde cada uno contiene 4 parámetros (x, y, h, w), los dos primeros son las coordenadas de dicho rectángulo en la imagen y los dos últimos son el alto y ancho del mismo.

2.5.5. Filtro y verificación frontal

Es un muy importante establecer los parámetros de la función detectMultiScale mencionada anteriormente para realizar un primer filtro y lograr una mayor precisión a la hora de la búsqueda. De esta forma se está eliminando posibles detecciones falsas filtrándolas por su tamaño.
Otra manera de evitar los falsos positivos, es filtrar los objetos detectados según su ubicación dentro de la imagen. Para esto, se debe analizar cada rectángulo obtenido en la detección y comparar sus coordenadas para ver si las mismas están dentro de la zona de aceptación.
Mediante un “for” se recorre las coordenadas de cada objeto y se las va comparando con la utilización de un “if” con constantes que representan el rango o valores de aceptación dentro del cual pueden estar los rostros.

```
if len(rects) > 0:
   for (x, y, w, h) in rects:
        if x > faceXmin and x < faceXmax and y > faceYmin and y < faceYmax:
```

Si se cumple con estas condiciones entonces el rostro detectado recién es considerado como verdadero y se procede a su análisis.

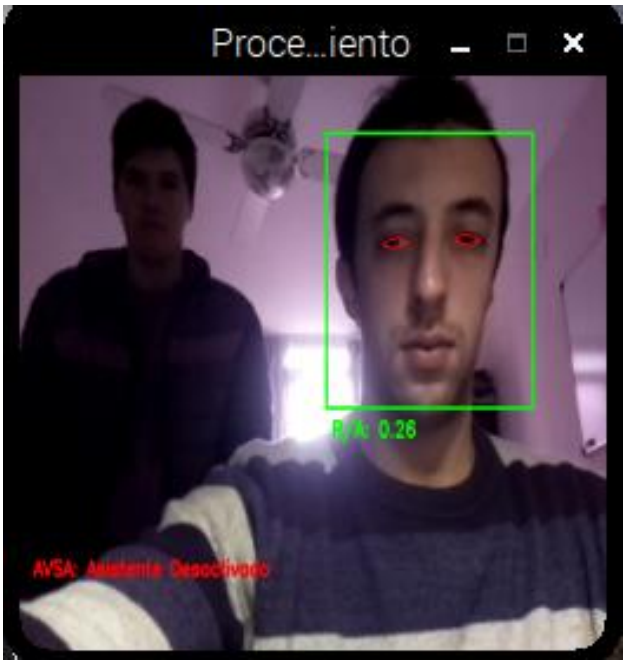

Fig. 22 Filtro frontal. Ejemplo de filtrado de rostro frontal en el sistema

En la imagen se aprecia cómo se filtra el rostro que se encuentra a un extremo de la imagen, y se detecta sin modificaciones el más próximo al centro de la imagen.

2.5.6. Detección perfil

Cuando el conductor se encuentre con la mirada hacia el costado del camilo, el sistema dejara de detectar el rostro frontal. Cuando esto sucede, el software procede a detectar el perfil del conductor.
Esta tarea se realiza de igual manera que el caso frontal, utilizando la función detectMultiScale. Pero se puede apreciar en el código, que en esta ocasión el objeto de la clase cascadeHaar no es el mismo que el anterior sino el cual contiene como archivo el detector de perfiles.

```
perfil = detector_perfil.detectMultiScale(gray, scaleFactor=1.1,
                                          minNeighbors=5, minSize=(100, 100),
                                          flags=cv2.CASCADE_SCALE_IMAGE)
```

De igual forma, es necesario establecer los parámetros de búsqueda de la función para una correcta detección de los perfiles.
Para este clasificador se ha encontrado inconvenientes a la hora de la detección, donde solo funcionaba o era más eficiente para un solo perfil de la persona. Esto se pudo mejorar de forma significativa con la siguiente línea de código:

```
gray = cv2.flip(gray,1)
```

Lo que hace la misma es invertir la imagen capturada. Por lo tanto, si a la primera no se encuentra un perfil la imagen es invertida y se procede a la búsqueda nuevamente.

2.5.7. Filtro y verificación perfil

Este paso se hace de manera similar al filtrado y verificación del rostro frontal. El cambio va estar dado por el área de aceptación, la cual es un tamaño mayor que en el caso anterior debido a la posible mayor movilidad del conductor a la hora de mirar hacia el costado.

```
if len(perfil) > 0:
   for (x, y, w, h) in perfil:
        if x > faceXmin and x < faceXmax and y > faceYmin and y < faceYmax:
```

2.5.8. Análisis del nivel de somnolencia

Una vez detectado el rostro frontal del conductor mediante el clasificador Haar, se crea un objeto rectángulo perteneciente a la clase Dlib con las coordenadas obtenidas en la detección. Este rectángulo es pasado como parámetro junto con la imagen capturada en escala de grises a la función predictor. Dado este recuadro delimitador de la cara, podemos obtener 68 puntos destacados utilizados para localizar los ojos, las cejas, la nariz, la boca y la mandíbula.

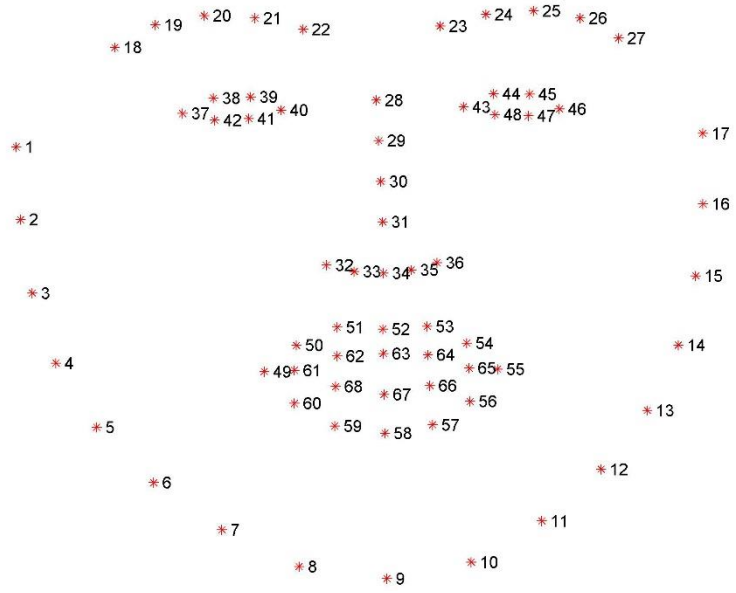

Fig. 23 Herramienta Dlib. Distribución de puntos faciales

El código implementado para la realización de estas operaciones es el siguiente:

```
rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
puntos_faciales = predictor(gray, rect)
puntos_faciales = face_utils.shape_to_np(puntos_faciales)
```

Es necesario para seguir con el procesamiento de estos datos, convertir la lista de puntos faciales a una matriz tipo Numpy, esto se realiza por medio de la función sahpe_to_np.
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.
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.

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

De la herramienta Dlib se puede extraer estructuras faciales específicas conociendo los índices de las partes faciales particulares. En este caso solo nos interesan dos conjuntos de estructuras faciales: los ojos.
Cada ojo está representado por 6 coordenadas, comenzando en la esquina izquierda del ojo y luego en sentido horario alrededor del resto de la región. Hay una relación entre el ancho y la altura de estas coordenadas.

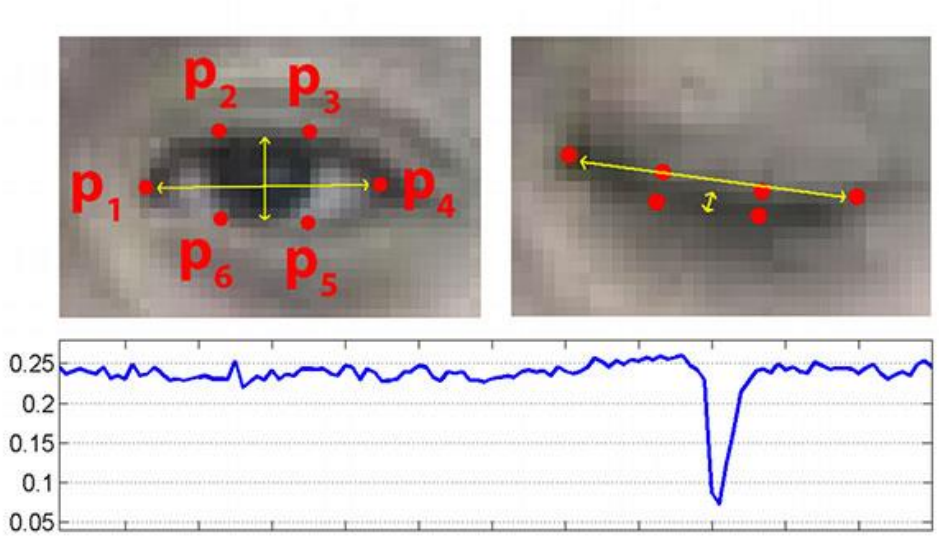

Fig. 24 Calculo de somnolencia. Variación en la relación de aspecto del ojo

En la esquina superior izquierda se observa un ojo que está completamente abierto y los puntos faciales del ojo trazados. Luego, en la parte superior derecha, un ojo que está cerrado. La parte inferior traza la relación de aspecto del ojo a lo largo del tiempo. Como se puede ver, la relación de aspecto del ojo es constante lo que indica que el ojo está abierto, luego cae rápidamente a cerca de cero, luego aumenta de nuevo, lo que indica que se ha producido un parpadeo.
Los puntos faciales devueltos por la herramienta Dlib siguen una lista indexable, por lo tanto, se busca determinar los valores de índice de división de matriz inicial y final para extraer las coordinadas para el ojo izquierdo y derecho de la siguiente forma:

```
# se obtienen los puntos faciales del ojo izquierdo y el derecho
respectivamente
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

Una vez obtenidos los índices, se extrae las coordenadas para el ojo izquierdo y derecho.

```
# se extrae las coordenadas del ojo izquierdo y derecho
Ojo_Izq = puntos_faciales[lStart:lEnd]
Ojo_Der = puntos_faciales[rStart:rEnd]
```

Con esta información ya se está en condiciones de calcular la relación de aspecto de cada uno de los ojos.

```
# se calcula la relación de aspecto para ambos ojos
RA_Ojo_Izq = rel_aspecto_ojo(Ojo_Izq)
RA_Ojo_Der = rel_aspecto_ojo(Ojo_Der)
```

Mediante la función rel_aspecto_ojo se realiza el cálculo del mismo. La misma acepta un solo parámetro, las coordenadas de los puntos de referencia faciales para un ojo dado.

```
def rel_aspecto_ojo(eye):
                # puntos de referencia verticales del ojo en coordenadas
                A = dist.euclidean(eye[1], eye[5])
                B = dist.euclidean(eye[2], eye[4])
                # distancia euclidiana de los puntos horizontales en
coordenadas
                C = dist.euclidean(eye[0], eye[3])
                # calculo de la relacion de aspecto del ojo
                RAo = (A + B) / (2.0 * C)
                return RAo
```

Lo primero que se calcula es 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.
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.

```
# Relacion de aspecto promedio
RA = (RA_Ojo_Izq + RA_Ojo_Der) / 2.0
```

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.

```
if RA < UMBRAL_RA:
        COUNTER += 1
        # compara la cantidad de frames consecutivos en los que se
        # mantuvieron los ojos cerrados
        if COUNTER >= ui.NIVEL_SOM:
                if not ALAR_SOM_MED:
                    ALAR_SOM_MED=True
                    ui.somnolencia.setPixmap(QtGui.QPixmap("somnolenciaMin.png"))
        if COUNTER >= ui.FRAMES_CONSEC:
                # si la alarma no esta encendida, la encendemos
```

Luego existe en este código otra constante importante de destacar, FRAMES_CONSEC. Este valor se establece en 30, para indicar que deben ocurrir 30 cuadros sucesivos con una relación de aspecto de ojo inferior a UMBRAL_RA para considerar que el conductor se encuentra en un estado de somnolencia.
Es importante destacar que este valor de cuadros que deben darse, va depender no solo de la velocidad de procesamiento de cada sistema sino también de la velocidad a la que vaya el vehículo. Por ejemplo, si el conductor cierra los ojos a 180 Km/h el sistema activara la alarma más rápido que si el auto iría a 40 Km/h.
COUNTER es un contador que lleva la cuenta del número total de fotogramas sucesivos que tienen una relación de aspecto de ojo inferior a UMBRAL_RA. Si este valor supera a FRAMES_CONCEC el sistema procede a la activación de la alarma correspondiente.

2.5.9. Activación de alarma

Las alarmas de somnolencia y de atención al camino están formadas por un tono y un código de voz de tal forma que el conductor pueda discriminar entre una y otra. Cuando se detecta somnolencia se ejecuta un tono seguido de un código por voz anunciando “Alerta de somnolencia”. La alarma se repetirá tantas veces como sea necesario hasta que el conductor salga del estado de somnolencia. Cuando se active la alarma de desatención, esta sonara con un tono y un código anunciando “Alerta, preste atención al camino”. Esta última sonara hasta que el conductor retome su vista al camino.

```
if not ALARM_ON:
                        ALARM_ON = True
                        sonido_alarma('tone.wav')
                        time.sleep(1.0)
                        sonido_alarma('alerta.wav')
                        ui.label_img.setPixmap(QtGui.QPixmap("atencion2.png"))
                        if not ALAR_SOM_MAX:
                                ALAR_SOM_MAX=True
                                ui.somnolencia.setPixmap(
                                        QtGui.QPixmap("somnolenciaMax.png"))
                        ui.label_img.setScaledContents(True)
                        cont_bal=30
```

Se utilizo para la reproducción de los archivos de audio la librería de Python llamada “Pygame”, la cual mediante la carga del archivo de audio en formato “wav” puede reproducirse mediante un simple comando.

```
# definicion de alarma sonora / archivo de audio alarm.wav
def sonido_alarma(archivo):
                pygame.mixer.init()
                pygame.mixer.music.load(archivo)
                pygame.mixer.music.play()
                return 0
```

```
def sonido_alarma_off():
                pygame.mixer.quit()
```

Las alarmas han sido grabadas con una voz de mujer y combinándolas con tonos de baja frecuencia esperando encontrar mejores resultados a la hora de quitar el estado de somnolencia del conductor.
Un estudio realizado por la *Universidad de Dundee, Escocia* y el *Servicio de Bomberos y Rescate de Derbyshire de Inglaterra* demostró que las personas reaccionan mejor a las advertencias de incendio grabadas con una voz de mujer, que a las alarmas ruidosas de los detectores de humo. Las alarmas que acompañaban a la voz femenina eran de baja frecuencia.

2.5.10. Activación por Velocidad

Para evitar posibles errores de detección de rostros o movimientos de la cabeza cuando el vehículo se encuentra detenido y a su vez lograr una relación de sensibilidad del sistema respecto a la velocidad actual del vehículo, es necesario saber la velocidad del

vehículo desde su marcha. Para lograr esto existen diferentes alternativas, obtener la velocidad del vehículo directamente de su instrumental o bien medirse a través de un elemento externo. El propósito de este proyecto es lograr un equipo independiente y adaptable, es por eso que se utiliza un GPS externo.

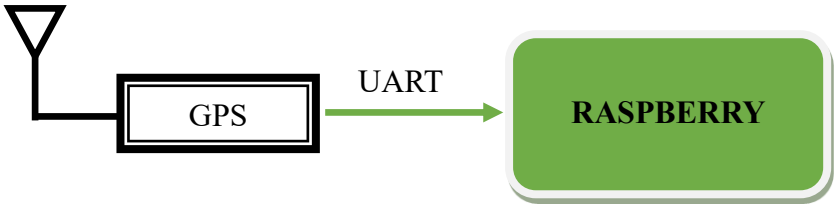

Fig. 25 Medición de velocidad. Conexión entre GPS y Raspberry

El GPS se conecta a la placa central a través de una comunicación serie por la UART.

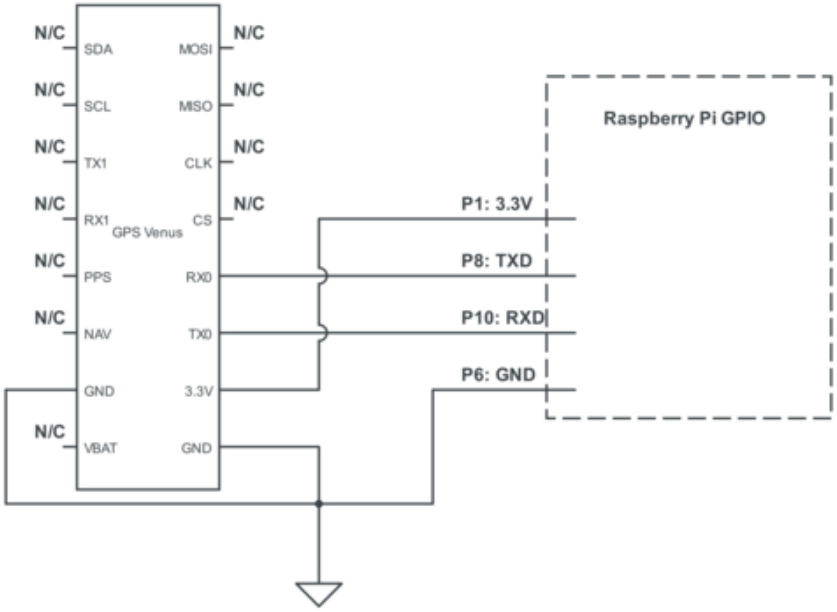

Fig. 26 Modulo GPS. Conexión con placa Raspberry Pi

La configuración del puerto serie para recibir los datos necesarios se realiza desde el código en lenguaje Python.

```
serie = serial.Serial('/dev/ttyAMA0', baudrate=9600,interCharTimeout=None}
t = threading.Thread(target=receiving,args=(ser,)).start()
```

Estas líneas inicializan el puerto, luego mediante una función leemos el puerto hasta encontrar el final de línea y lo almacenamos en un buffer.

```
def receiving(serie):
  global continue_reading,last_received
  buffer = ''
  while continue_reading:
    buffer += serie.read(serie.inWaiting())
    if '\n' in buffer:
      lines = buffer.split('\n')
      last_received = lines[-2]
      buffer = lines[-1]
```

```
      parse_nmea(last_received.strip())
 serie.close() = threading.Thread(target=receiving,args=(ser,)).start()
```

El GPS reporta la ubicación calculada por medio de mensajes NMEA, este es un protocolo estándar de comunicación que utiliza cadenas simples de texto que resultan muy fáciles de procesar. La cadena de caracteres está formada por:
$ GPGGA, HHMMSS.SSS, latitud, N, longitud, E, FQ, NOS, HPD, altitud, M, altura, M ,, datos de suma de comprobación.
La función extraer_nmea es la encargada de extraer todos los datos de la cadena recibida del GPS.

```
def extraer_nmea(line):
  global last_coord,last_time,spd,altitude,sats
  gpsdata = line.split(',')
  if gpsdata[0] == '$GPGGA':
    # tiempo
    curr_time = parse_time(gpsdata[1])
    # latitud
    lat = float(gpsdata[2][:2])+(float(gpsdata[2][2:])/60)
    if gpsdata[3] == 'S':
      lat = lat*-1
    # longitud
    lon = float(gpsdata[4][:3])+(float(gpsdata[4][3:])/60)
    if gpsdata[5] == 'W':
      lon = lon*-1
```

Con la información de ubicación, lo único que debe hacerse es calcular la distancia entre los dos puntos obtenidos por el GPS y dividirlo entre el tiempo que ha transcurrido en obtener esas dos posiciones. Para calcular esta distancia se tiene en cuenta que las coordenadas que nos brinda el GPS están en grados y minutos por lo que debemos aplicar la fórmula de Haversine, la cual facilita el cálculo de la distancia de círculo máximo entre dos puntos de un globo sabiendo su longitud y su latitud.

$$\mathrm{a} = \sin^2(\frac{\Delta\varphi}{2}) + \cos(\varphi 1)\,.\cos(\varphi 2).\sin^2(\frac{\Delta\lambda}{2})$$

$$c = 2.a.\tan 2(\sqrt{a}, \sqrt{1-a})$$

$$d = R.c$$

Donde: φ es la latitud, λ es longitud y R es el radio de la tierra.
Para realizar este cálculo dentro del software, hay que utilizar las funciones incluidas en la biblioteca “math” de Python.

```
def distance(coordenada1,coordenada2):
  R = 6371
  distLat = math.radians(coordenada2[0]- coordenada1[0])
  distLon = math.radians(coordenada2[1]- coordenada1[1])
  lat1 = math.radians(coordenada1[0])
```

```
lat2 = math.radians(coordenada2[0])
a = math.sin(dLat/2)*math.sin(distLat/2)+math.sin(distLon/2)*
    math.sin(distLon/2)*math.cos(lat1)*math.cos(lat2)
c = 2 * math.atan2(math.sqrt(a),math.sqrt(1-a))
return R*c
```

2.5.11. Métodos de iluminación del rostro

El objetivo es mantener iluminada la cara del conductor y las pupilas, para evitar errores en el procesamiento de la imagen. Cuando el nivel de luz externa disminuye, se utiliza una fuente de luz infrarroja de baja potencia, la cual pasa inadvertida para el conductor.

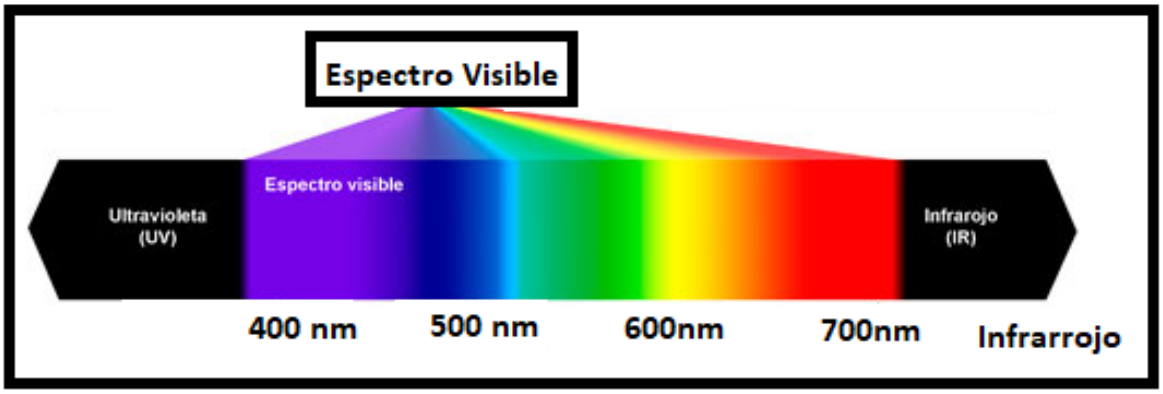

Fig. 27 Métodos de iluminación. Espectro de la luz

La luz visible se encuentra en el intervalo de los 400nm a los 750nm, luego cercano a los 750 nm en adelante se encuentran el infrarrojo. Este a su vez, se divide en tres regiones espectrales que aproximadamente se encuentran en los siguientes intervalos:

- *Infrarrojo cercano:* 700nm - 1500nm. La luz producida en esta franja no es térmica.
- *Infrarrojo medio:* 1500nm - 15000nm.
- Infrarrojo lejano: 15000nm - 1mm.

En este proyecto es de utilidad únicamente la luz perteneciente al infrarrojo cercano, precisamente en las longitudes de onda próximas a los 900nm.
Varios estudios advierten que una emisión prolongada de un valor determinado de potencia en el espectro infrarrojo puede afectar a la retina y a la córnea, produciendo photoretinitis y hasta una posible ceguera, ya que pueden recalentar el tejido celular del ojo. Es por ello que debe hacerse un cuidadoso diseño en este tipo de iluminación ya que al estar radiando continuamente sobre el ojo se podría producir lesiones, en el caso de potencias elevadas.
Una publicación realizada por *American Conference of Governmental Industrial Hygienists* en su artículo "*The Threshold Limit Values and Biological Exposure Indices*" establece que el límite de radiación en una longitud de onda comprendida entre los 700nm a los 3000nm generados por diodos de emisión "led" hacia la córnea y el cristalino del ojo no debe exceder los 10 mW/cm2 en exposiciones mayores a 1000 segundos.
Experimentalmente para un espectro infrarrojo cercano entre los 770 nm y los 950 nm el límite de radiación que produce lesiones en la retina es aproximadamente 20W/cm2 para una exposición de 100 segundos y 18W/cm2 para exposiciones mayores a 1000 segundos.
Como se ha mencionado antes en el informe se ha utilizado un módulo de dos leds en el espectro de infrarrojo cercano con una longitud de onda de 840nm. Cuando el habitáculo del vehículo este iluminado por luz natural, el infrarrojo no se activa. A medida que la luz

externa disminuye el infrarrojo eleva su potencia. Mediante una resistencia de regulación se ha controlado la potencia de los leds para no generar daños a los ojos del conductor.

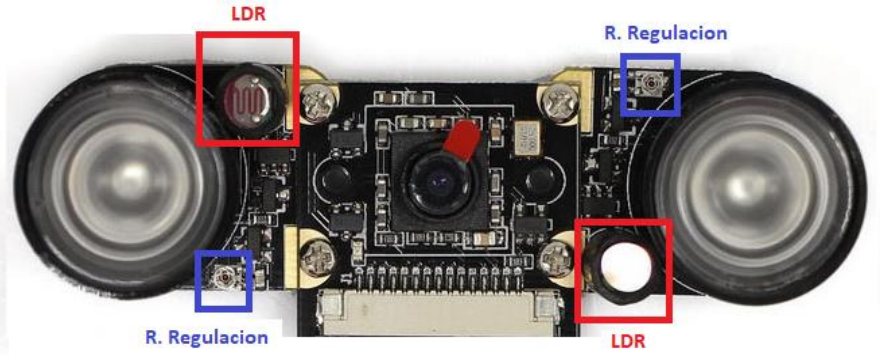

Fig. 28 Modulo Infrarrojo. Disposición en LDR y pote de regulación en la placa

El LDR es el encargado de sensar la luz externa que influye sobre la cara del conductor, de este modo aumentar o disminuir la iluminancia de los infrarrojos.

2.5.12. Interfaz grafica

Para realizar esta etapa, se utilizó la herramienta ya mencionada PyQt 5. El diseño de toda la interfaz gráfica fue realizado primeramente en Qt Creator, para luego convertir este proyecto a un código en lenguaje pythom para ejecutar este mismo desde el software que realiza el procesamiento de imágenes.

Fig. 29 Interfaz gráfica. Diseño en herramienta Qt Creator

Una vez finalizada la creación del proyecto en Qt, con todos los archivos del mismo dentro de la carpeta Scripts de Python se utiliza ambas herramientas marcadas en la siguiente imagen para convertir el proyecto.ui al código en Python y los recursos necesarios también al mismo lenguaje.

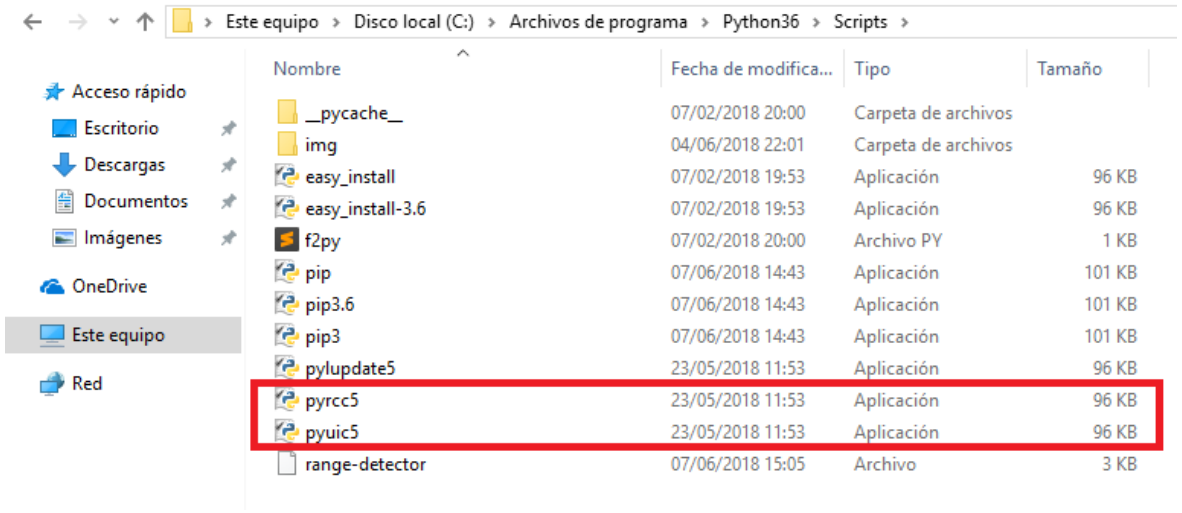

Fig. 30 Diseño de interfaz. Herramientas para conversión de código C++ a Python

Con los archivos necesarios ya generados solo queda incorporar dicho código en el software que realiza el procesamiento. Es importante incorporar en el código todas las librerías de PyQt 5 necesarias para llevar a cabo la creación de los objetos incluidos en la interfaz.

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import pyqtSlot
```

En la primera línea son herramientas necesarias para la creación de los objetos incluidos en la interfaz. En la segunda, se incluye lo requerido para realizar las señales que conectan a la interfaz con el código Python para poder modificar los objetos o el estado de los mismos.

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(800, 480)
```

En este código se puede apreciar la creación de la venta principal en lenguaje Python y la especificación del tamaño de la misma.
Para la imagen de fondo de la interfaz, se creó un objeto Label para usar de contenedor de la misma. El código es el siguiente:

```
self.label.setPalette(palette)
self.label.setMidLineWidth(0)
self.label.setText("")
self.label.setPixmap(QtGui.QPixmap(":/instrument-cluster.png"))
self.label.setScaledContents(True)
self.label.setObjectName("label")
```

La aplicación cuenta demás de alarmas sonoras con señalizaciones visuales, estas se realizan creación un objeto Label como el caso anterior donde se va modificando la imagen del mismo según lo que se quiera indicar. El código para hacer esta tarea es idéntico al ejemplo anterior.

La presentación final de la aplicación es la siguiente:

Fig. 31 Interfaz gráfica. Visualización al iniciar el sistema

Como primer indicador tenemos el objeto central de la interfaz, el cual nos da la información del estado en el que se encuentra el equipo. Vemos en la imagen anterior que cuando el vehículo circula a una velocidad menor a los 40 Km/h el sistema se presenta en un estado desactivado el cual se indica en tal objeto.

Fig. 32 Componentes de la interfaz. Indicador de estado en Activo

Este indicador de estado puede presentar 4 condiciones, activado, desactivado, somnolencia o prestar atención al camino. Siendo las dos últimas alarmas visuales que se presentan en pantalla por un tiempo corto luego de ser activadas. Estos dos casos se pueden ver en las siguientes 2 imágenes.

Fig. 33 Componentes de la interfaz. Indicador de estado en atención al camino

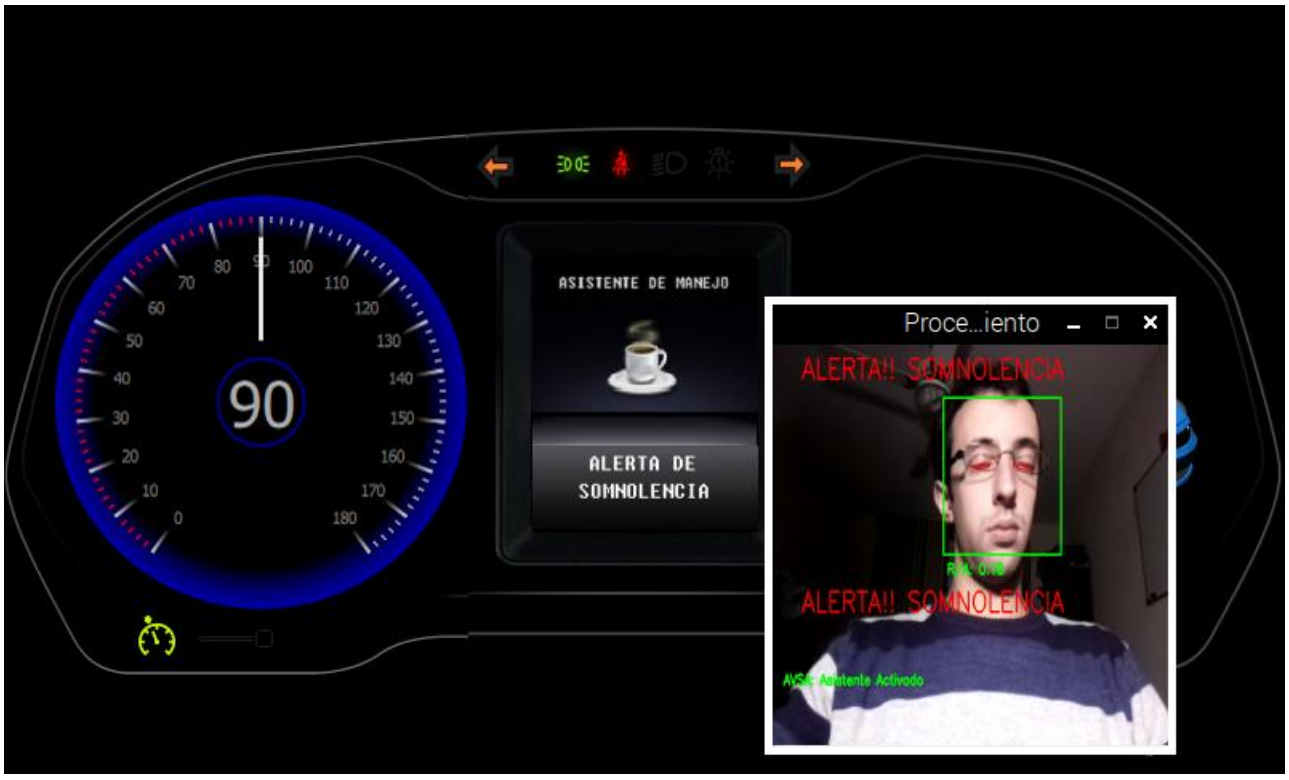

Fig. 34 Componentes de la interfaz. Alerta de somnolencia activada

En cuanto al estado de somnolencia, la aplicación también cuenta con otro indicador para mostrar 2 estados posibles del mismo. Cuando la barra de estado se muestra por la mitad, significa que el conductor se encuentra ya presentando signos importantes de fatiga. En otros términos, los parpadeos de la persona son de un tiempo significativo, pero aun sin pasar el umbral crítico del sistema. En cambio, cuando la barra se muestra completa, el conductor ya presenta un estado crítico y el sistema no desactiva este indicador hasta que el vehículo se detenga.

Fig. 35 Indicador de somnolencia. Estado intermedio activado

Fig. 36 Indicador de somnolencia. Estado maximo activado

Cada vez que se active una de las dos alarmas posibles, también se cuenta con un indicador intermitente como señalización. Estas balizas se pueden apreciar en la siguiente ilustración.

Fig. 37 Componentes en la interfaz. Indicador intermitente

Debido a que la velocidad del vehículo es un dato muy importante para el sistema, y la misma es obtenida mediante el módulo GPS existe una imposibilidad del probar el sistema fuera del automóvil. Por ello, se realizó mediante la aplicación grafica la posibilidad de simular el mismo para poder realizar las pruebas del equipo.
En la siguiente captura de la pantalla del sistema se puede apreciar los componentes necesarios para esta simulación.

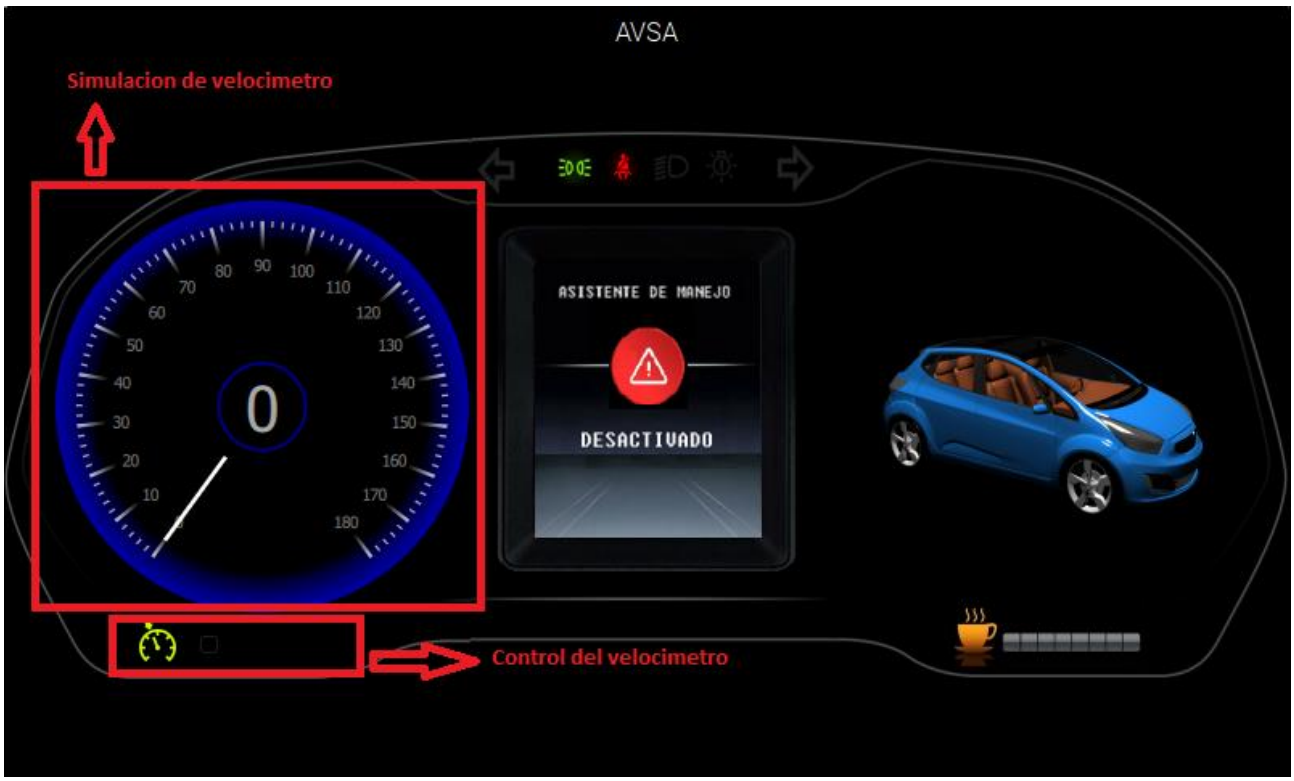

Fig. 38 Componentes de la interfaz. Velocímetro y control manual del mismo

Se cuenta con un control manual para modificar la velocidad y un componente visual para poder ver el valor seleccionado. Con dicha simulación se puede modificar la velocidad desde los 0 Km/h hasta los 180 Km/h.

3. Resultados

3.1. Resultados en la detección de los ojos y distracción

Se realizaron pruebas de laboratorio con iluminación controlada y en condiciones de iluminación natural en el habitáculo de un automóvil. En ambos casos se ha intentado simular las situaciones de desatención y de somnolencia. Realizar pruebas en condiciones reales de conducción resultan complicadas debido a que los eventos de somnolencia no siempre ocurren en las conducciones diarias a fin de evaluar el sistema. Los fabricantes de automóviles disponen de recursos para evaluar sus productos en ambientes parecidos a los reales, con vehículos especializados, donde los conductores son físicamente fatigados antes del inicio del experimento o simulación, consiguiendo que aumente la probabilidad de que se presenten estos eventos. La recolección de estos datos no es viable porque no podemos acumular suficientes ejemplos de conducción somnolienta para evaluar completamente el sistema. Además, sería irresponsable realizar este experimento sin tomar el tipo de medidas preventivas. Por estas razones se evaluó el sistema utilizando un conjunto de datos creado a partir de eventos de somnolencia controlados.
El dispositivo dentro del vehículo ha sido instalado sobre el tablero de tal forma que no obstaculice la visión del camino.

Fig. 39 Prueba del sistema. Ubicación del equipo en el automóvil

Para las pruebas de laboratorio se ha montado una carcasa diseñada en impresora 3D con la inclinación similar a la postura dentro del habitáculo.

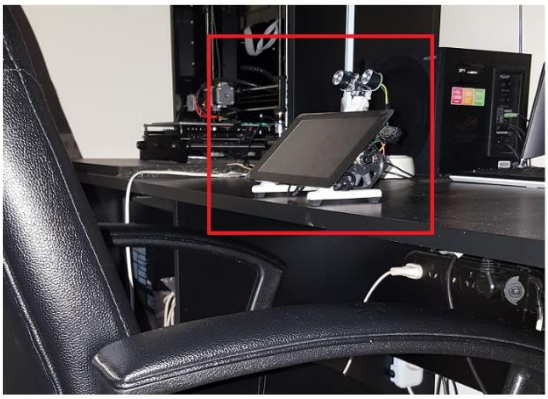

Fig. 40 Prueba del sistema. Disposición del equipo en el laboratorio

Las pruebas de somnolencia y distracción se han realizado sobre dos conductores tanto en laboratorio como en el habitáculo de un automóvil. Para la evaluación de la efectividad del sistema se han tomado diferentes muestras, considerando como muestras los frames tomados por la cámara en diferentes situaciones y movimientos. Se ha intentado simular las actitudes de los conductores durante una conducción normal y algunas de las actitudes físicas que genera la somnolencia.

Porcentaje de detección de somnolencia en laboratorio cada 50 frames.

Conductor	Numero de Aciertos	% de Deteccion
Conductor 1	45	**90%**
Conductor 2	43	**86%**

Tabla 2 Pruebas del sistema. Detección de somnolencia en laboratorio

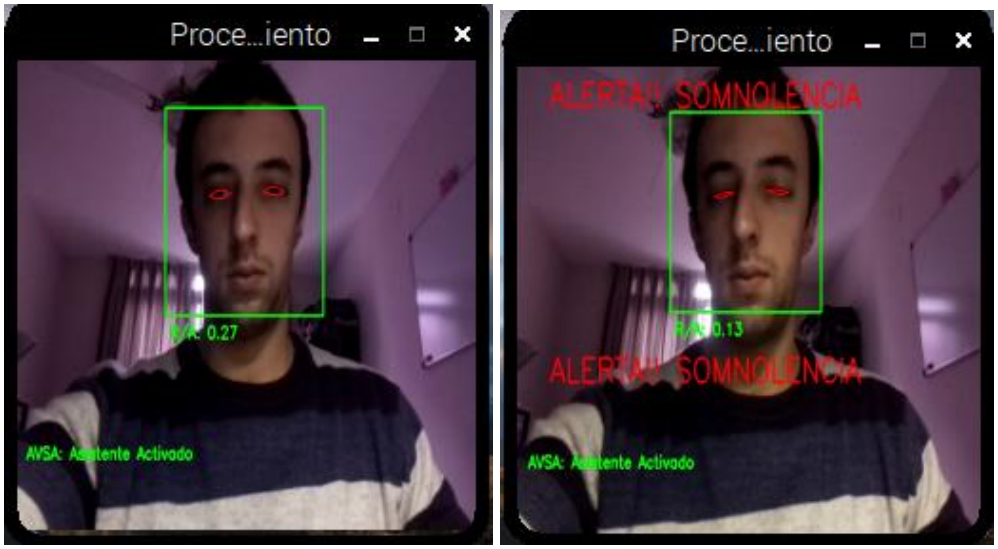

Fig. 41 Resultado final. Detección de somnolencia

Porcentaje de detección de distracción en laboratorio cada 50 frames.

Conductor	Numero de Aciertos	% de Deteccion
Conductor 1	42	**84%**
Conductor 2	45	**90%**

Tabla 3 Pruebas del sistema. Detección de distracción en el laboratorio

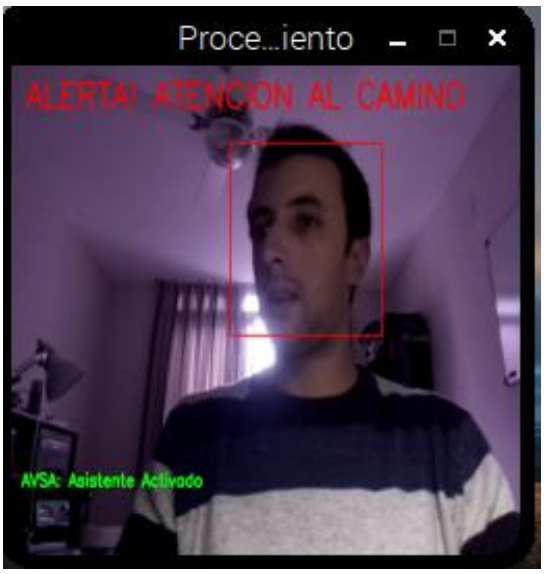

Fig. 42 Resultado final. Detección de distracción

Porcentaje de detección de somnolencia con lentes recetados en laboratorio cada 50 frames.

Conductor	Numero de Aciertos	% de Deteccion
Conductor 1	45	**90%**
Conductor 2	-	**-**

Tabla 4 Pruebas del sistema. Detección de somnolencia con lentes recetados

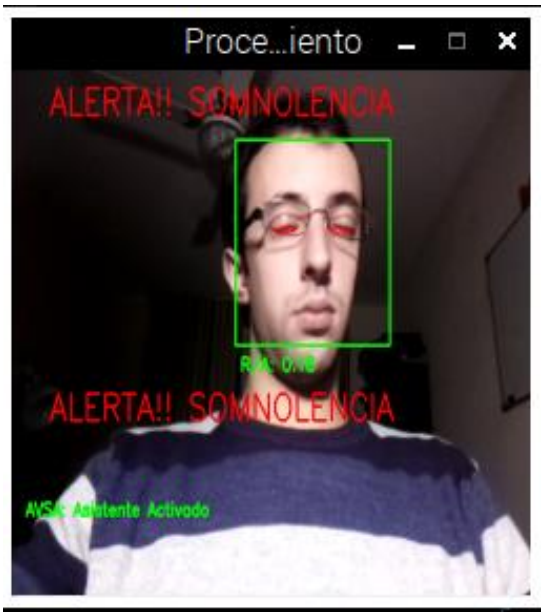

Fig. 43 Resultado final. Detección de somnolencia utilizando lentes recetados

Porcentaje de detección de somnolencia con lentes oscuros en laboratorio cada 50 frames.

	Conductor	Numero de Aciertos	% de Deteccion
Lentes económicos	Conductor 1	30	**60%**
	Conductor 2	26	**52%**
Lentes con filtro UV antireflex	Conductor 1	0	0%
	Conductor 2	0	0%

Tabla 5 Pruebas del sistema. Detección de somnolencia con lentes oscuros

Fig. 44 Resultado final. Sistema en funcionamiento con lentes oscuros

En las próximas imágenes se muestran los frames con los lentes de sol con filtros UV antirreflejo, en el cual el sistema es incapaz de localizar con efectividad los ojos.

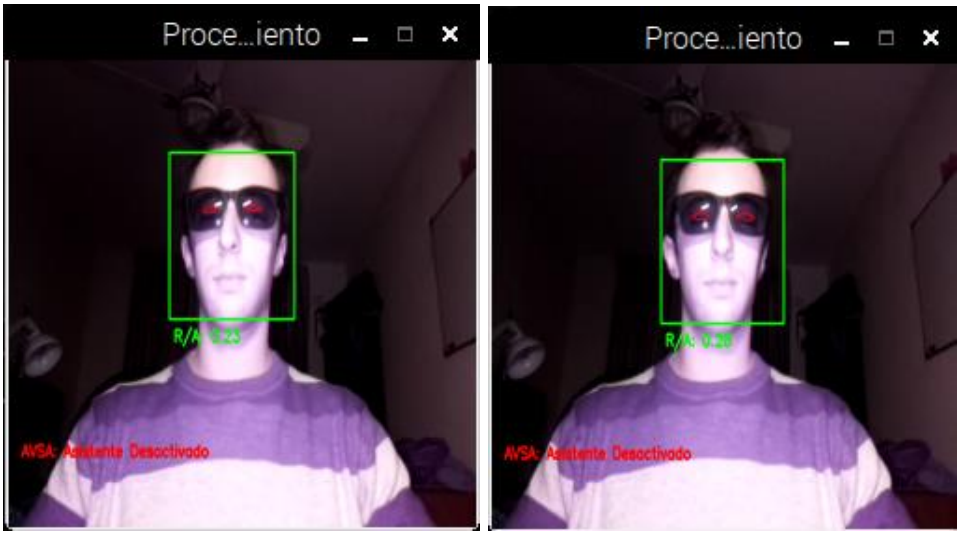

Fig. 45 Resultado final. Problemática de los lentes oscuros en el sistema

Porcentaje de detección de distracción en laboratorio cada 50 frames con lentes de sol.

Conductor	Numero de Aciertos	% de Deteccion
Conductor 1	45	**90%**
Conductor 2	-	**-**

Tabla 6 Pruebas del sistema. Detección de distracción con lentes de sol

Fig. 46 Resultado final. Detección de distracción con lentes oscuros

Porcentaje de detección de somnolencia en automóvil.

Conductor	Numero de Aciertos	% de Deteccion
Conductor 1	39	**80%**
Conductor 2	37	**75%**

Tabla 7 Pruebas del sistema. Detección de somnolencia en el vehículo

Fig. 47 Resultado final. Sistema funcionando en el vehículo

Porcentaje de detección de distracción en automóvil.

Conductor	Numero de Aciertos	% de Deteccion
Conductor 1	42	**84%**
Conductor 2	42	**84%**

Tabla 8 Pruebas del sistema. Detección de distracción en el automóvil

Fig. 48 Resultado final. Detección de distracción en el vehículo

El sistema de detección de somnolencia tiene una eficiencia promedio en la detección de 90% para los ensayos realizados en laboratorio con iluminación controlada y una eficiencia de 80% en el habitáculo del vehículo con influencia de luz externa. No se han encontrado variaciones con el uso de lentes recetados antirreflejo, obteniendo el mismo resultado. El sistema de detección de distracción tiene una eficiencia promedio de 85% en ambos casos. Durante el uso de lentes de sol no se puede garantizar el funcionamiento de la detección de somnolencia, ya que en algunos casos según el tipo de lente y las variaciones en la iluminación externa se producen errores en la localización de los parpados. El sistema de detección de distracción no sufre inconvenientes en tales casos.

A pesar de las grandes ventajas que presenta la iluminación infrarroja tales como resaltar las pupilas para detectar el estado de los ojos y de iluminar la cara en conducciones nocturnas, presenta ciertas desventajas que hace que no sea perfectamente eficiente para trabajar en ambientes sometidos a cambios de iluminación ambientales. Es decir, la iluminación infrarroja funciona correctamente bajo condiciones estables de iluminación, esto tiene como consecuencia que el sistema sea altamente inestable por depender de factores que no se pueden controlar.

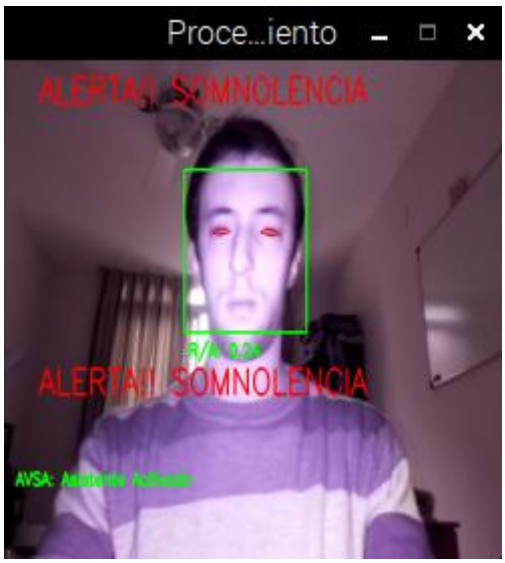

Fig. 49 Infrarrojo. Exceso de luz infrarroja

En la imagen se aprecia un exceso de iluminación infrarroja debido a las variaciones de iluminación ambiental, provocando un falso positivo acusando somnolencia con los ojos abiertos.

3.2. Detección de ojos con uso de lentes de sol

Las características de los ojos que se utilizan para realizar en análisis y la detección de los parpados son el color y la forma, la mayoría de los ojos se pueden aproximar a una forma elíptica y todos ellos independientemente del color de ojo tienen el globo ocular blanco. Los puntos oculares devueltos por la herramienta Dlib dependen del análisis de estas dos cualidades que se ven afectadas con el uso de lentes oscuros. Por lo tanto, debemos buscar la transparencia del cristal de los lentes respecto a alguna longitud de onda que podamos procesarla. Se han ensayado dos longitudes de onda dentro del infrarrojo cercano (850nm y 940nm), logrando resultados positivos para un grupo específico de lentes de sol.

Fig. 50 Lentes de sol. Detección de los ojos a través de gafas de sol

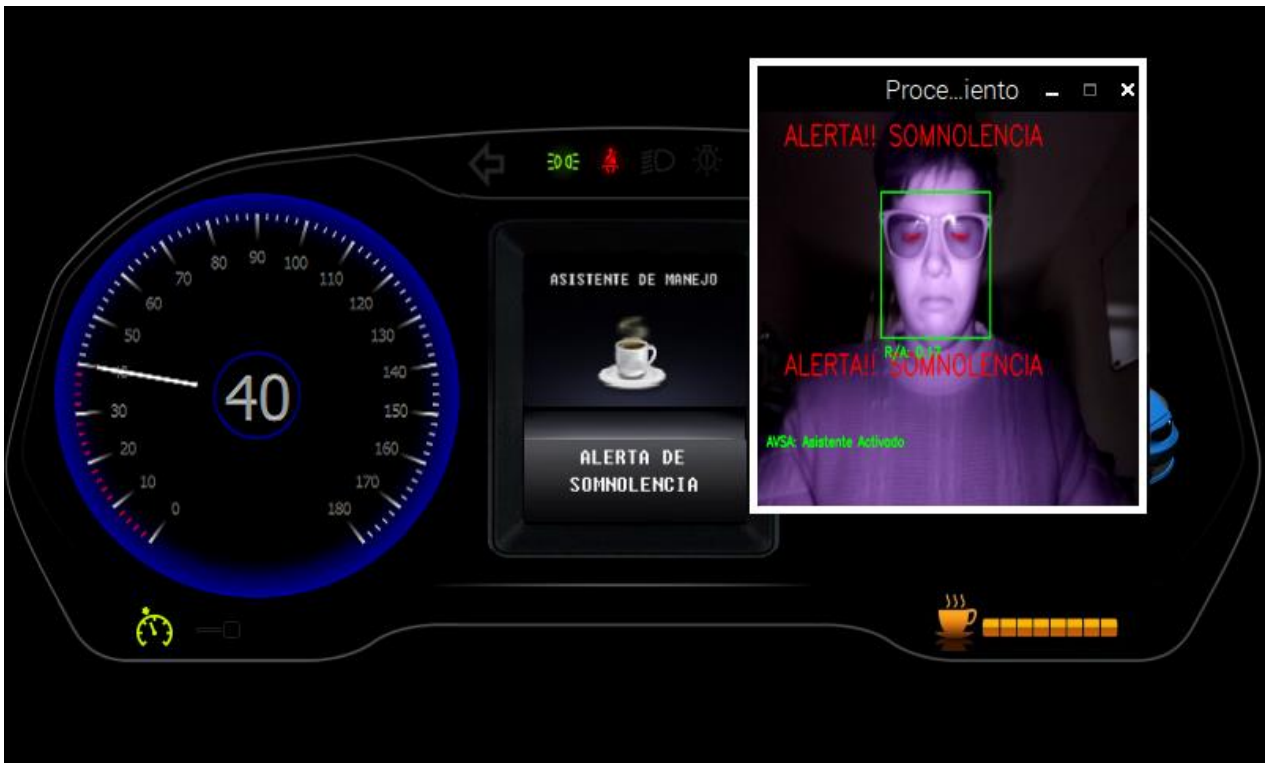

Fig. 51 Lentes de sol. Detección de somnolencia con lentes de sol

La funcionalidad del sistema de detección de atención al camino no se vio afectada con el uso de lentes ya que realiza un proceso de análisis diferente al de los ojos.

Fig. 52 Lentes de sol. Alerta de distracción activada

3.3. Métodos fallidos para el sistema de detección de ojos

La primera estrategia que se utilizó para detectar el parpadeo en el conductor fue determinar el estado del ojo según si se detectaba o no el iris del mismo. Para lograr esto se usó el método de clasificadores Haar.

Se determina primeramente la posición del rostro mediante el clasificador, una vez hecho esto, dentro de la cara se procede a ubicar los ojos utilizando el mismo método Haar. Cabe destacar que existe un clasificador para cada ojo, la carga de los mismo se puede apreciar en el siguiente código.

```
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade_left = cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')
eye_cascade_right = cv2.CascadeClassifier('haarcascade_righteye_2splits.xml')
```

Una vez detectado el rectángulo donde se encuentra el rostro, se fragmenta el dos partes las cuales son sometidas a la búsqueda de cada ojo. Dicho en otras palabras, en la parte derecha de la cara se busca el ojo derecho y con la parte izquierda el ojo izquierdo.

```
eyes_right = eye_cascade_right.detectMultiScale(roi_gray_right,
                                                1.3, 5, 0,(60,60),(100,100))
eyes_left = eye_cascade_left.detectMultiScale(roi_gray_left,
                                                1.3, 5, 0,(60,60),(100,100))
```

Una vez obtenidos ambos ojos, en el recuadro en el que se encuentran cada uno de ellos se procede a realizar el procesamiento para ubicar el iris. Lo primero que se hace es eliminar la porción de la imagen que no es útil, es este caso la eliminación de los parpados. Para ello se aplica una máscara a la imagen del ojo para solo seleccionar el iris y la pupila del mismo.

Fig. 53 Método fallido. Imagen utilizada para filtrar información innecesaria de los ojos

Para detectar la pupila se utiliza una función que detecta objetos en la imagen que tengan forma circular, pero para ello la imagen que se desea procesar debe estar en blanco y negro. La herramienta que logra esto es la que se puede apreciar en el código.

```
t, eye_left_binary = cv2.threshold (roi_eye_left_gray, 0, 255,
                                    cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

Esta función threshold, necesita un valor umbral para binarizar la imagen. Todos los valores que se encuentran por encima del mismo son convertidos al blanco y los que están por debajo se transforman en negro. Por eso, al utilizar la máscara vista anteriormente se elimina la información que no es útil y por lo tanto al convertir en blanco y negro se logra una diferenciación muy clara de la pupila. Este resultado se puede apreciar en la siguiente ilustración.

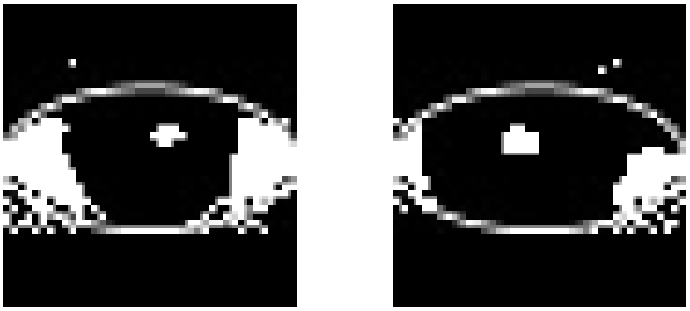

Fig. 54 Método fallido. Resultado de análisis de los ojos

Ahora solo queda aplicar HoughCircles, la misma devuelve las formas circulares que encuentra dentro de la imagen. Se utiliza de la siguiente manera.

```
circles_left = cv2.HoughCircles(eye_left_binary,cv2.HOUGH_GRADIENT,1,20,
                        param1=50,param2=20,minRadius=11,maxRadius=100)
if circles_left is not None:
        circles_left = np.round(circles_left[0, :]).astype("int")
        for (x, y, r) in circles_left:
                cv2.circle(roi_eye_left_color, (x, y), r, (0, 0, 255), 4)
                cv2.circle(roi_eye_left_color_2, (int(x/2.5),
                                    int(y/2.5)), int(r/2.5), (0,0,255), 4)
                cv2.rectangle(roi_eye_left_color, (x - 5, y - 5),
                                    (x + 5, y + 5), (0, 128, 255), -1)
```

Las primeras dos líneas corresponden a la detección de los circulo, todo lo demás es para filtrar el tamaño buscado y para dibujar en la imagen donde se ha detectado la pupila. Esto debe ser realizado en forma separada para cada ojo. El resultado es el siguiente.

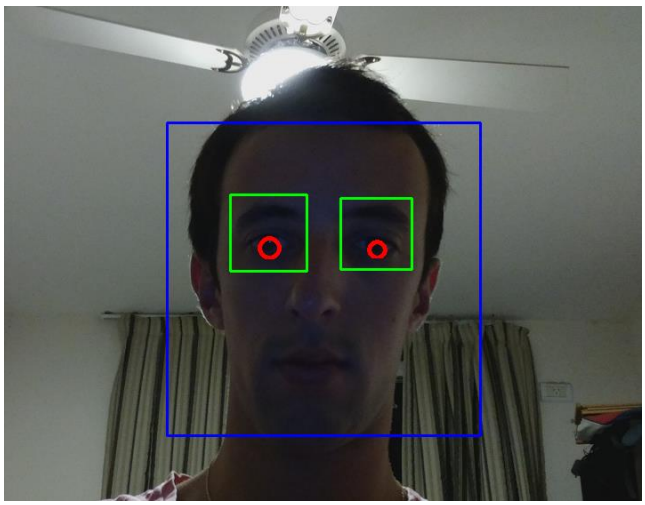

Fig. 55 Método fallido. Resultado final del sistema

Cuando se probó el sistema en otras condiciones lumínicas, el mismo presentaba muchas dificultades. En muchos casos se detectaba el iris en forma incorrecta y en muchos directamente no se detectaba, por ello este sistema ha sido descartado.

Otra alternativa para el sistema fue en vez de detectar la pupila ubicar los parpados y compararlos con una elipse. De esta forma, comparando el radio vertical de dicha elipse, se puede determinar qué tan cerrado están los ojos del conductor.

```
_, contornos , _ = cv2 . findContours ( eye_left_binary . copy (),
                                        cv2.RETR_EXTERNAL ,cv2.CHAIN_APPROX_SIMPLE )
cnt = contornos [ 0 ]
elipse = cv2 . fitEllipse ( cnt )
cv2.ellipse( roi_eye_left_color , elipse , (0,255,0) , 2 , cv2 . LINE_AA )
```

La ubicación de los ojos se realiza de igual forma que el caso anterior, pero antes de detectar la elipse es necesario aplicar la función findContours a la imagen de cada ojo. Una vez establecido todos los contornos de la imagen, se aplica la función fitEllipse para detectar la elipse del ojo y el comando ellipse para dibujar el mismo en la imagen.

Cada elipse que se detecta va tener como parámetros los radios vertical y horizontal, las coordenadas de ubicación y el ángulo de inclinación al que se encuentra dicha figura. Con estos datos se puede hacer tranquilamente un análisis de la apertura del ojo de la persona.

Este método tampoco tuvo buenos resultados a la hora de las pruebas, por el mismo motivo que el anterior debido a los cambios de luminosidad.

Por lo tanto, estos dos métodos fueron descartados y se utilizó la herramienta Dlib con la que se logró la mejor performance.

3.4. Limitaciones del Hardware

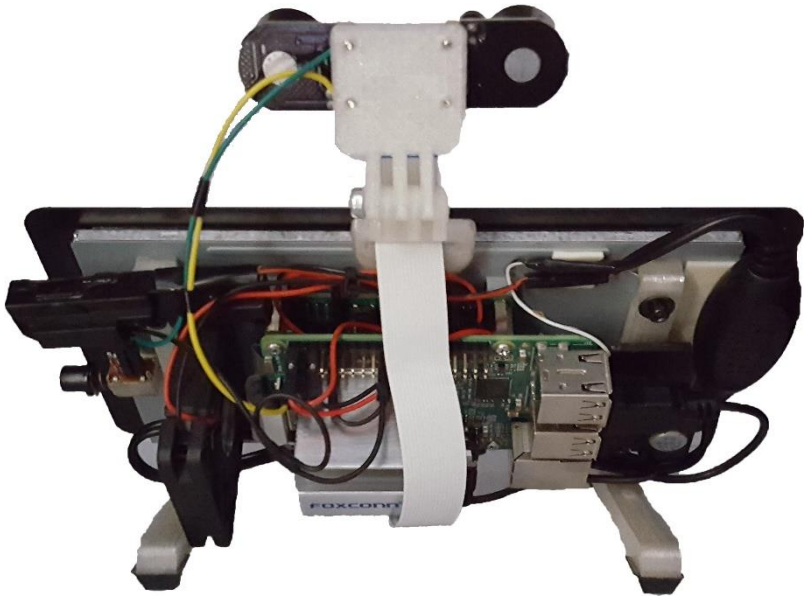

Fig. 56 Hardware. Vista posterior del equipo

La Raspberry Pi 3 es una placa muy versátil y rápida. Así mismo debido al complejo proceso de captura y análisis de los frames se ha notado rendimientos muy bajos, por los cuales se llega a la obligación de sacrificar precisión en el procesamiento de los frames. Además de esto se han realizado cambios a nivel procesador, tales como velocidad de procesador y voltajes, conocidos como Overclocking.
El Overclocking se realiza a partir de modificar el archivo “config.txt” dentro del directorio /boot.
Los valores predeterminados son los siguientes:

```
arm_freq = 1200
gpu_freq = 400
core_freq = 400
sdram_freq = 450
over_voltage_sdram = 0
```

Luego de realizar las modificaciones, el archivo “config.txt” queda de la siguiente manera:

```
arm_freq = 1350
gpu_freq = 500
core_freq = 400
sdram_freq = 500
over_voltage_sdram = 0
```

Realizar modificaciones en los parámetros del procesador exige un cuidadoso diseño respecto a la disipación de temperatura, ya que, de no disipar el calor, se estaría acortando la vida de la placa de desarrollo.

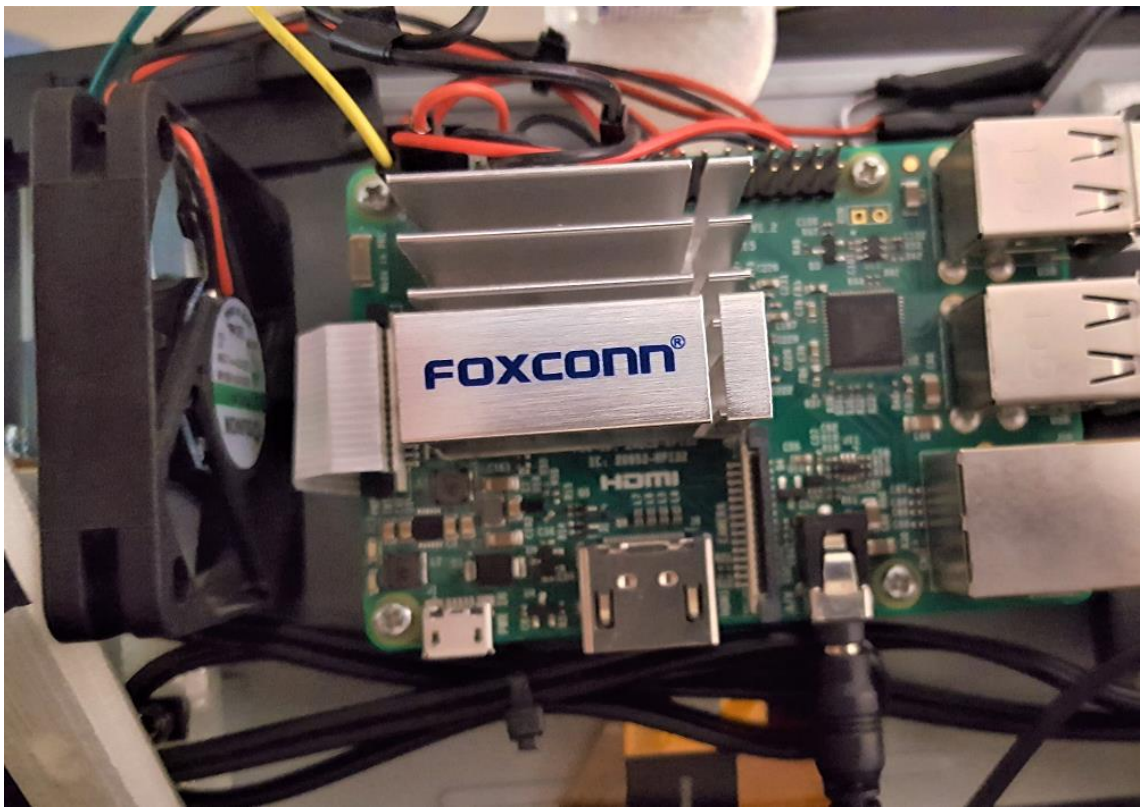

Fig. 57 Hardware. Disipador del equipo

Para asegurar la estabilidad en temperatura se ha montado un disipador de aluminio con aletas y un sistema de ventilación forzada.

El segundo paso para mejorar el rendimiento ha sido sacrificar precisión en la detección del rostro reemplazando los detectores HOG y vectores lineales SVM por Cascadas de Haar. Mientras que los detectores HOG y vectores lineales SVM tienden a ser significativamente más precisos que las cascadas Haar, requieren un nivel de procesamiento mucho más elevado lo que los hace notablemente más lentos.

Si bien con los pasos anteriores se logró agilizar el procesamiento, el sistema en general mostraba algo de lentitud, por lo que fue necesario disminuir el número de frames por segundos capturados por la cámara y su resolución.

La cámara instalada tiene una capacidad de 8 megapíxeles, siendo capaz de tomar imágenes estáticas de 3280 x 2464p, y también es capaz de tomar vídeo de 1080p a 30fps. Luego de las modificaciones inevitables, la cámara quedo configurada en 640x480 a 20fps.

4. Análisis de Costos

4.1. Plan de negocios

El modelo de negocios pensado para emprender el proyecto es el modelo CANVAS, el mismo está estructurado de la siguiente manera:

Propuesta de Valor:

- Sistema de detección de parpadeo anti somnolencia
- Sistema de detección de distracción
- Sistema de alerta por voz
- Portabilidad y adaptabilidad del sistema de seguridad
- Estatus de seguridad al contar con el sistema AVSA
- Diseño compacto y atractivo
- Facilidad de uso y configuración

Segmento de Mercado:

- Automovilistas en general
- Empresas de colectivos de transporte público y privado
- Trenes de transporte de pasajeros
- Empresas de transporte de cargas
- Fabricantes de automóviles
- Fabricantes de colectivos y camiones

Canales de Comunicaciones y distribución:

- Ventas directas al cliente a través de Internet
- Ventas a través de casas de repuestos
- Ventas en tiendas de audio/alarmas
- Ventas personalizadas a grandes empresas

Relación con los clientes:

- Realimentación constante en mejoras del sistema
- Soporte técnico y asistencia
- Capacitación y fidelización

Modelo de ingresos:

- Ingresos a través de transacciones de pagos puntuales de clientes
- Servicio técnico
- Desarrollo de software o hardware a pedido de clientes
- Ingresos recurrentes derivados de pagos periódicos realizados a cambio de un servicio postventa

Recursos clave:

- Software de calidad optima
- Profesionales al servicio del sistema
- Experiencia en aplicaciones de electrónica
- Apoyo económico gubernamental
- Oferta de mano de obra calificada en el sector

Actividades clave:

- Desarrollo del software
- Diseño del Hardware- Estrategias de Marketing
- Plazos cortos de tiempo de entrega
- Disponibilidad de servicio técnico
- Atención al cliente

Socios claves:

- Alianzas estratégicas con fabricantes de vehículos
- Proveedores de dispositivos electrónicos
- Revendedores e instaladores de Sistemas de seguridad habilitados por la marca
- Convenios con casas de ventas de repuestos

Estructura de costos:

- Recursos Humanos
- Marketing y ventas
- Proceso de diseño y ensamble
- Costos variables de materias primas
- Infraestructura para el desarrollo

4.2. Costos y Retorno de Inversión

La efectividad de la inversión, suponiendo la fabricación de 50 equipos, con un valor comercial establecido a **U$S 1.200** por unidad:

50 equipos vendidos/mes = **U$S 60.000** (Ingreso Mensual)

Costos por unidad de fabricación:

- Hardware: U$S 300
- Carcasa: U$S 20
- Software: U$S 100

Total: U$S 420

Para fabricar 50 equipos el costo de Insumos y software es U$S 21.000, si se considera además que las campañas de marketing, las instalaciones, infraestructura y equipos para el desarrollo son solventados a partir de la inversión inicial:

- Suponiendo una inversión inicial de **U$S 42.000**

El retorno de la inversión con 50 equipos vendidos en el mes:

- ROI=((Ingreso Mensual - Inversión) /inversión)*100%= **43 %**

4.3. Plan de Marketing

El plan de marketing pensado para el producto:
Mediante un análisis del mercado, se estudia las competencias y se encuesta al cliente para determinar lo que valora en el producto.
Se pretende hacer llegar a los potenciales clientes una virtualización del sistema mediante video simulación en 3D, donde se muestra las acciones del sistema. La difusión será por distintos medios, haciendo un fuerte uso de redes sociales. El objetivo es que el cliente conozca la existencia del producto, y sobre todo llegue a identificar el mismo a través de sus iniciales. Se estudiarán las críticas y las respuestas del público para hacer las modificaciones necesarias.
Se pretende que el cliente asocie el sistema AVSA a, tranquilidad, seguridad y estatus elevado. De tal forma que aquellos que viajen en servicios de transporte público o privado, encuentren como servicio de mayor calidad a aquellas empresas que cuenten con el sistema.
Se acordará con empresas de transporte para que prueben el sistema en forma gratuita, contribuyendo a la empresa con la colocación de las iniciales identificativas del sistema en sus unidades de transporte.
Se desarrollará una página web, donde se encuentre toda la información del equipo, con imágenes, videos y todo lo necesario para dar a conocer este producto de excelentes prestaciones. Además de poder realizar la compra, consultas, asesoramiento, solicitud de servicio técnico, etc.

5. Discusión y Conclusión

5.1. Conclusiones

El asistente de manejo AVSA esta implementado mediante dos detecciones fundamentales: la detección de cara y la detección de parpadeos. Los resultados obtenidos en la detección de los parpadeos para determinar la somnolencia son positivos, ya que se han obtenido pocos errores, los cuales, en la mayoría de los casos, ocurrieron por influencia de la iluminación. Generalmente no han ocurrido errores en la localización de los ojos, esto se debe a que se logró un software preciso para garantizar la obtención del estado de los parpados.

La detección de parpadeos ha requerido un complejo sistema de iluminación y análisis para la identificación correcta de los ojos ya que el conductor puede conducir con lentes recetados o bien lentes de sol. Otro aspecto destacable es la implementación de umbrales adaptativos para la estimación de los parámetros de parpadeo, es decir, los valores que deciden si el ojo está abierto o cerrado. Esto ha permitido que el asistente se adapte a diferentes formas y tamaños de ojos además de patrones de parpadeo personalizados. Para evaluar el detector de parpadeos se analizó cada frame procesado por el software dando como resultado una eficiencia del 90% en condiciones de iluminación controlada. Los resultados obtenidos con lentes recetados no han sufrido cambios significativos. En cuanto al uso de lentes de sol si bien se ha logrado la transparencia en algunos casos, no se puede asegurar el funcionamiento para la detección de somnolencia utilizando solo la información de los ojos, para tal caso sería útil medir otro parámetro para aumentar la fiabilidad.

El uso de SVM en conjunto con cascadas de Haar ha servido para construir un método eficiente para detectar el estado de somnolencia y distracción del conductor. Detectar si un ojo está abierto o cerrado en situaciones de iluminación variable resulta una tarea difícil, sobre todo con las limitaciones que pueda presentar el hardware en un dispositivo de tales características. La combinación de ambas técnicas ha logrado un método robusto para analizar el estado de los ojos a través del tiempo y con ello generar el índice de somnolencia en tiempo real.

En la detección de atención, se han obtenido buenos resultados, logrando identificar situaciones donde el conductor desvía su campo de visión del camino en un 85% de efectividad.

Se ha utilizado para el desarrollo del proyecto imágenes de videos de conducción reales. Generando aspectos importantes a considerar, tales como los cambios de iluminación, condiciones de iluminación nocturna y diurna.

Se puede afirmar que el método implementado en este proyecto ha permitido cumplir los objetivos principales dejando abiertas muchas opciones de mejora y ampliación de funcionalidades que invitan y motivan a seguir trabajando en su perfeccionamiento.

5.2. Mejoras en futuros desarrollos

Si bien los objetivos planteados en el proyecto fueron alcanzados satisfactoriamente, aún resta mucho trabajo por hacer hasta llegar a construir un sistema que sea capaz de monitorizar todas las actividades del conductor y que sea 100% robusto y fiable para ser instalado en un vehículo comercial.

Entre los principales trabajos que se pueden desarrollar para contribuir y mejorar el proyecto se encuentran:

Principalmente, como ya se ha dicho en el informe anteriormente, el sistema de detección de somnolencia no es fiable cuando se utilizan lentes oscuros, por tal motivo es necesaria la inclusión de otra variable que permita continuar el análisis de somnolencia aun cuando los ojos no son totalmente detectables. Otra de las posibles soluciones para esta situación seria trabajar con distintas longitudes de onda y detectores para encontrar la transparencia de los distintos tipos de lentes, teniendo siempre en cuenta que estas no sean perjudiciales a la salud de los ojos del conductor.

Otra de las mejoras sería el monitoreo de las variables del organismo del conductor, es decir de acciones preventivas para poder evitar que ocurra un accidente, como señales cardiacas. Esta perspectiva, se complementa con las mejoras mecánicas que han aplicado las distintas fábricas automotrices en sus automóviles.

Incluir la información externa, por ejemplo: la temperatura, la hora del día, las condiciones del clima, etc. Esto contribuye a mejorar el índice de somnolencia e interactuar de mejor manera con la información medida sobre el conductor.

Adicionalmente pueden aplicarse diversos algoritmos sobre los datos monitoreados con el fin de encontrar patrones y asociarlos a una situación particular, por ejemplo, un determinado movimiento del conductor.

La utilización de un acelerómetro en los ejes X, Y y Z, permitiría identificar cuando el auto realiza un movimiento inusual, como puede ser un lomo de burro, un movimiento de volante a izquierda o derecha.

6. Literatura Citada

Asociación Civil Luchemos Por La Vida. Estadísticas. sitio web, 2018
http: //www.luchemos.org.ar/es/estadisticas

NHTSA National highway traffic safety administration. Sitio web, 2018.
http: //www.nhtsa.dot.gov/

Viola P. and Jones M. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, Proceedings of the 2001 IEEE Computer Society Conference on, 2001.

Organización mundial de la salud. Informe sobre la situación mundial de la seguridad vial 2015. Sitio Web, *http://www.who.int/violence_injury_prevention/road_safety_status/*

RACE. Datos generales de siniestralidad. Sitio web, 2018.
http://www.race.es/portal/transform.jsp?seccion=/docs/20060110/0020.xml&xml=/docs/20060313/0001.xml&xsl=/contenido.xsl&menu=0&submenu=0&menu3=1

Lamble, D., Tatu, K., Laakso, M., Summala H., Cognitive load and detection thresholds in car following situations: safety implications for using mobile (cellular) telephones while driving. Publicado en 2010. Unidad de Investigación de Tráfico del Departamento de Psicología, Universidad de Helsinki, Helsinki, Finlandia. Sitio Web :
http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V5S3X6B5KY3&_user=10&_coverDate=11/30/1999&_rdoc=1&_fmt=high&_orig=search&_origin=search&_sort=d&_docanchor=&view=c&_searchStrId=1464995441&_rerunOrigin=scholar.google&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=c928a6e5b0c6b6cc89cc36445f41cbee&searchtype=a

News Center Ford Motor Company. Ford Develops Heart Rate Monitoring Seat, Adds New Element to Health and Wellness Research. Sitio web, 2018
http://corporate.ford.com/news-center/news/pressreleases/press-releases-detail/pr-ford-develops-heart-rate-34664

Toyota Motor Corporation, Hino Motors Ltd. (2009). TMC, Hino to Test Breathalyzer Ignition-interlock System. Publicado en 2011. Sitio web:
http://pressroom.toyota.com/article_display.cfm?article_id=10

Drowsiness detection with OpenCV by Adrian Rosebrock, 2018. Sitio Web:
https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

Stutts, J. C., Wilkins, J. W., Osberg, J. S., & Vaughn, B. V. (2003). Driver risk factors for sleep-related crashes. Accident Analysis and Prevention. Sitio Web:

https://uncch.pure.elsevier.com/en/publications/driver-risk-factors-for-sleep-related-crashes

K. Torkkola, N. Massey & C. Wood, “Driver inattention detection through intelligent analysis of readilyavailable sensors”, 2004

T. Ersal, H. J. Fuller, O. Tsimhoni, J. L. Stein & H. K. Fathy, “IEEE Transactions on Intelligent Transportation Systems”, 2010