UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE DETECCIÓN DEL ESTADO DE SOMNOLENCIA PARA CONDUCTORES MEDIANTE EL USO DEL SOFTWARE TEACHABLE MACHINE.

Trabajo de titulación previo a la obtención del

Título de Ingeniero en Electrónica

AUTORES: ABRAHAM ANTONIO MOSQUERA MAYORGA

IVÁN ANDRÉS VENTURA TORRES

TUTOR: ING. LIVINGTON MIRANDA, MGTR.

Guayaquil – Ecuador

2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Abraham Antonio Mosquera Mayorga con documento de identificación N° 0953962313 y Iván Andrés Ventura Torres con documento de identificación N° 0931335277, manifestamos que:

Somos las autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 05 de febrero del año 2024.

Atentamente,

Abraham Antonio Mosquera Mayorga
095396231-3

Iván Andrés Ventura Torres
093133527-7

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Abraham Antonio Mosquera Mayorga, con documento de identificación No. 0953962313 e Iván Andrés Ventura Torres con documento de identificación No. 0931335277 expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del artículo académico: "Diseño e implementación de un prototipo de detección del estado de somnolencia para conductores mediante el uso del software Teachable Machine", el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 06 de febrero del año 2024

Atentamente,

Abraham Antonio Mosquera Mayorga
0953962313

Iván Andrés Ventura Torres
0931335277

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Livington Miranda con documento de identificación N° 0930635172, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE DETECCIÓN DEL ESTADO DE SOMNOLENCIA PARA CONDUCTORES MEDIANTE EL USO DEL SOFTWARE TEACHABLE MACHINE, realizado por Abraham Antonio Mosquera Mayorga N° 0953962313 e Iván Andrés Ventura Torres con documento de identificación N° 0931335277 ,obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 5 de febrero del año 2024.

Atentamente,

Ing. Livington Miranda, Mgtr.
0930635172

DEDICATORIA

A Dios por darme la salud y fuerza en todo este periodo académico y a mis padres, Fanny y Washington, quienes me han inculcado valores y principios desde la niñez y en quienes puedo contar con todo el apoyo en cada situación que pueda atravesar en la vida.

Abraham Mosquera M.

A Dios por permitirme luchar día a día por mis sueños, a mis amados padres, por apoyarme y alentarme en mis proyectos, por darme todo e impulsarme a soñar en grande para seguir mis metas hasta alcanzarlas. Sin todo el amor y la fortaleza que me han dado no hubiera podido seguir adelante. Los amo infinitamente.

Iván Ventura T.

AGRADECIMIENTO

A Dios por brindarme el privilegio de una familia maravillosa.

A mis padres, por su entrega y constancia en el arduo trabajo de formarnos ¡Las seis flores de su jardín, no seríamos sin ustedes! A mis hermanas por siempre estar presentes con cariño y paciencia por su preocupación por mis sueños, por ser mi fuerza en los momentos débiles, sin todos ustedes no habría alcanzado esta meta.

Abraham Mosquera M.

A Dios por otorgarme la dicha de culminar mis estudios, a mis padres por siempre apoyarme y poder darles este orgullo, a la universidad por formarme como profesional y sobre todo a mí por nunca darme por vencido.

Iván Ventura T.

Resumen

Este proyecto de titulación responde a la problemática de los accidentes de tráfico, para ofrecer una solución dirigida a prevenir situaciones causadas por la somnolencia de los conductores. Se ha desarrollado un prototipo que se instala en la parte delantera del tablero, utilizando una cámara para reconocer imágenes y emitir una alarma al detectar signos de somnolencia.

La implementación se llevó a cabo utilizando el sitio web TEACHABLE MACHINE, específicamente en la sección de proyectos de imagen. A través de una webcam, se capturaron imágenes de poses asociadas a la somnolencia. Posteriormente, se procedió al entrenamiento de un modelo mediante la opción "Export Model" de TEACHABLE MACHINE, utilizando la configuración "Tensorflow" y seleccionando "Saved Model". El modelo entrenado se descargó en formato .h5, el cual se integró en una Raspberry Pi.

En la fase final del proyecto, se importó el archivo .h5 a la Raspberry Pi, y se eligió un puerto para conectar un altavoz. Este altavoz emite una alarma sonora al reconocer las poses características de la somnolencia, alertando así al conductor sobre su estado.

Palabras Clave: Entrenamiento en Teachable Machine, detección de somnolencia, captura de fotos por webcam, exportar modelo creado, sonido cuando hay somnolencia.

Abstract

This graduation project arises in response to the problem of traffic accidents, aiming to provide a solution targeted at preventing situations caused by driver drowsiness. A prototype has been developed, which is installed at the front of the dashboard, utilizing a camera to recognize images and trigger an alarm upon detecting signs of drowsiness.

The implementation was carried out using the TEACHABLE MACHINE website, specifically in the image projects section. Through a webcam, images of poses associated with drowsiness were captured. Subsequently, a model was trained using the "Export Model" option of TEACHABLE MACHINE, employing the "Tensorflow" configuration and selecting "Saved Model". The trained model was downloaded in .h5 format, which was then integrated into a Raspberry Pi.

In the final phase of the project, the .h5 file was imported into the Raspberry Pi, and a port was selected to connect a speaker. This speaker emits a sound alarm upon recognizing characteristic poses of drowsiness, thereby alerting the driver about their state. In summary, this work focuses not only on technical development using tools such as TEACHABLE MACHINE and Raspberry Pi but also on the application of this knowledge to enhance road safety, especially in situations involving driver fatigue.

Keywords: Teachable Machine training, drowsiness detection, webcam photo capture, export created model, sound when there is drowsiness.

ÍNDICE DE CONTENIDO

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

El presente trabajo de titulación busca diseñar e implementar un prototipo que ayude a mejorar la seguridad vial evitando accidentes por posibles conductores en estado somnolencia debido a la fatiga, para lo cual se usará técnicas derivadas de la inteligencia artificial. Se ha identificado un índice de riesgo vial que suele ser subestimado, y se refiere a la posibilidad de quedarse dormido al volante. Con datos estadisticos recientes de otros países revelan que la somnolencia está vinculada a aproximadamente un 20% a 30% de los accidentes de tráfico mortales.

Este proyecto emplea la plataforma TEACHABLE MACHINE, específicamente en la sección de proyectos de imagen, para capturar imágenes mediante una webcam que representan las posiciones asociadas a la somnolencia. Este proceso se inicia con la recopilación de datos a través de fotografías de las distintas poses que los conductores adoptan cuando experimentan somnolencia. Estas imágenes son utilizadas para entrenar un modelo de detección, gracias a la interfaz intuitiva y accesible de TEACHABLE MACHINE.

Además de la implementación técnica de la detección de somnolencia a través de TEACHABLE MACHINE, también se integra el sistema en el entorno de un vehículo, para alertar proactivamente al conductor ante situaciones peligrosas.

El documento se separa en las siguientes secciones la investigación comienza con la descripción de los objetivos, alcances, hipótesis y justificación del proyecto. El cuerpo del documento consta de cuatro capítulos: el primero aborda la somnolencia, la visión artificial y los microcomputadores; la segunda se describen las técnicas que se

utilizarán para el análisis y recolección de datos; el tercer capítulo presenta los detalles específicos de la propuesta, indicando las herramientas que se utilizarán y un estudio de opciones para encontrar la mejor opción en términos de tiempo y costo. Finalmente, el cuarto capítulo detalla la implementación, pruebas y análisis de los resultados de la propuesta.

I. PROBLEMA

Una investigación realizada por la Fundación AAA para la Seguridad del Tráfico en los Estados Unidos, descubrió que una cifra importante de los accidentes fue producida por la somnolencia y que esta cifra se triplicaba en las horas nocturnas. Monitorearon a más de 3.500 personas de seis lugares en los Estados Unidos durante varios meses; de 701 accidentes que los especialistas estudiaron, la somnolencia fue un factor entre el 8.8% a 9.5%. (AAA Foundation for Traffic Safety, 2016).

La tecnología de “Atención Asistente” asiste al conductor Mercedes-Benz ATTENTION ASSIST utiliza un algoritmo y sensores especializados para detectar la fatiga del conductor. Los ingenieros de Mercedes-Benz realizaron una investigación exhaustiva que encontró que los conductores cansados pueden cometer pequeños errores de dirección y cambiar ligeramente su comportamiento al volante. El sistema emitirá el mensaje audible y visible "Es hora de un descanso" después de analizar las respuestas y los parámetros. Atención Asistente evita accidentes causados por la fatiga del conductor. (Mercedes-Benz of Easton, 2024).

Además de Mercedez Benz, existen otras marcas que cuentan con sistemas similares entre ellas están: Volvo con la función Driver Alert Control (DAC) que tiene como objetivo ayudar al conductor a reconocer cuando comienza a manejar de manera errática, como si estuviera distraído o experimentando somnolencia.

Esta función se usa para carreteras principales y pretende identificar un progresivo deterioro en el modo de conducir. El tráfico urbano no está previsto para la función. El sistema comienza a funcionar cuando la velocidad supera los 65 km/h (40 mph) y permanece activo cuando la velocidad supera los 60 km/h (37 mph). (Volvo, 2023).

El sistema de detección de fatiga con mantenimiento de carril de Ford utiliza una cámara frontal para monitorear las líneas del carril y detectar si el vehículo tiende a desviarse. Este sistema ayuda al usuario en la dirección para mantener un curso regular. El detalle interesante es que el detector de fatiga complementa esta función al alertar al conductor si percibe comportamientos en el manejo que puedan indicar signos de cansancio. (Cultura geek, 2017)

En el Ecuador los carros de entrada como los city car no disponen de asistentes electrónicos de detección de somnolencia para el conductor por ello se pretende diseñar un prototipo que pueda alertar cuando exista la somnolencia.

II. JUSTIFICACIÓN

La Agencia Nacional de Tránsito del Ecuador en 2024 expuso que, en el año 2023 se registraron 20.994 siniestros de tránsito y un saldo de 2373 fallecidos; la conducción en estado de somnolencia fue el 1,13 % del total de siniestros, 238 siniestros y 25 fallecidos. Este proyecto pretende desarrollar un prototipo que detecte la somnolencia en conductores y así dejar las cifras de accidentes de tránsito por fatiga en el porcentaje más bajo posible.

Varias actividades de investigación en marcha han analizado el estado de sueño y fatiga de los conductores. Lovato (2020) propone un sistema de detección de micro sueño utilizando un módulo Raspberry Pi V3. Este módulo incluye un dispositivo de grabación visual que registrará imágenes del conductor y las compara en una base de datos. Utilizando la librería OpenCV y el lenguaje de programación Python, se procesarán los datos serán analizados por algoritmos diseñados para detectar el nivel de somnolencia y emitirán un aviso visual y un sonido continuo de alerta si es superior al 80%, empleando una maniobra simple y práctica, pero no tan exacta. Como se puede notar en la figura número 1.

Figura 1

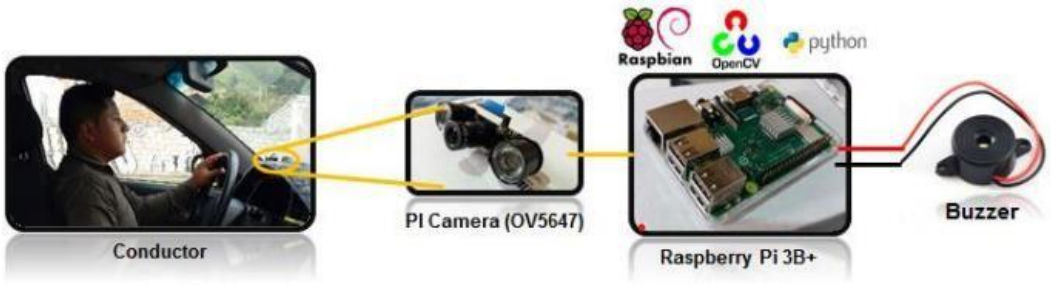
Implementación del prototipo Baiza Lovato.

Nota: Vista de implementación de prototipo (Lovato, 2020)

Alba Neppas (2020), implementó un sistema de visión artificial en el cual incorporó una pequeña computadora Raspberry Pi 3 B+. Este sistema analiza las ilustraciones capturadas mediante el uso una cámara web RGB, localiza los ojos y las expresiones del conductor, y determina el porcentaje de apertura de sus ojos (índice EAR). Si el valor es inferior al umbral generado dinámicamente al arrancar el sistema, el dispositivo genera un sonido de aviso mientras el conductor presente síntomas de somnolencia. Usando la matriz de incertidumbre para cuantificar su desempeño, Alba obtuvo el 88,25 % de precisión en un entorno controlado y 87 % de precisión en un entorno de conducción real. En la figura 2 se puede apreciar el orden de procesos del prototipo de Alba Neppas.

Figura 2

Orden de procesos para detección de somnolencia.



Nota: Arquitectura de prototipo (Alba Neppas, 2020)

A causa de lo que antes se ha mencionado hemos propuesto nuestro trabajo de tesis usando inteligencia artificial con una cámara integrada de mejores prestaciones, es así que se ha conseguido resultados con mayor porcentaje de aciertos. Además de contribuir en la seguridad del conductor, pasajeros y/o peatones.

III. OBJETIVOS

3.1 Objetivo general

Diseñar e implementar un prototipo de detección de somnolencia para conductores viales mediante la programación de un modelo de prevención en el software Teachable Machine, para la generación de alertas anticipadas sobre el estado de fatigas y prevención de accidentes en carretera.

3.2 Objetivo Específicos

- Capturar las posturas para el dataset del software Teachable Machine.
- Entrenar el modelo en base al dataset creado.
- Implementar el prototipo en una tarjeta Raspberry Pi IV, usando la cámara OV5647 para demostrar la viabilidad y funcionalidad del sistema de detección en tiempo real.

IV. MARCO TEÓRICO

4.1 Somnolencia

Las alteraciones producidas por la somnolencia son peligrosas al volante y pueden comprometer la seguridad vial. Conducir en un estado somnoliento o de fatiga puede tener consecuencias similares o incluso peores que conducir bajo los efectos del alcohol u otras sustancias, siempre es fundamental estar descansado y alerta mientras se conduce.

Algunas de las alteraciones más importantes que la somnolencia puede causar en la conducción son:

- Incremento del tiempo de reacción**:** La capacidad de reaccionar rápidamente ante situaciones inesperadas se ve significativamente disminuida cuando estás somnoliento, lo que puede resultar en colisiones y accidentes.

- Menor concentración y más distracciones: La falta de concentración debido a la somnolencia puede hacer que te distraigas con mayor facilidad, aumentando el riesgo de no notar señales de tráfico importantes, luces de freno de otros vehículos, cambios en el tráfico, etc.

- Alteraciones motoras: Los músculos relajados y la disminución de la coordinación motora pueden resultar en movimientos menos precisos y reacciones más lentas, lo que dificulta el control del vehículo.

- Alteración de las funciones sensoriales: La visión, uno de los sentidos más cruciales para la conducción, se ve gravemente afectada. La dificultad para enfocar

la vista y los problemas de visión pueden dificultar la identificación de obstáculos y señales.

- Aparición de micro sueños: Los micro sueños, aunque breves, pueden causar una desconexión momentánea de la realidad y dejar al conductor en un estado de inconsciencia temporal, lo que es especialmente peligroso al volante.

- Alteraciones en la percepción: La somnolencia puede distorsionar la percepción de señales, luces y sonidos, lo que aumenta la posibilidad de cometer errores al interpretar la información del entorno. (UNASEV, 2020)

4.2 Visión artificial

La visión artificial es un campo de la inteligencia artificial y el procesamiento de imágenes que buscan poder interpretar y comprender el mundo visual igual que los humanos. Esto se logra a través del análisis y procesamiento de imágenes digitales. (Krishna, 2020)

- Captura de Imágenes: Las imágenes se capturan a través de dispositivos con cámaras, como cámaras digitales, teléfonos móviles, cámaras de vigilancia, drones, etc. Estas imágenes se componen de píxeles que contienen información de color en las bandas RGB (rojo, verde y azul).

- Procesamiento de Imágenes: Una vez que se capturan las imágenes, se inicia el proceso de procesamiento. Esto implica usar algoritmos y técnicas para manipular y analizar imágenes para obtener información relevante.

- Reconocimiento**:** La visión artificial busca automatizar tareas que normalmente requerirían la interpretación humana, como el reconocimiento de objetos,

personas, patrones, gestos y más en imágenes y videos. Para esto, se aplican algoritmos de machine learning y métodos de procesamiento de imágenes para identificar patrones y particularidades.

- Aplicaciones: La visión artificial se utiliza en diversos ámbitos, como la medicina (diagnóstico médico por imágenes), la industria manufacturera (control de calidad automatizado), la robótica (navegación y manipulación de objetos), la seguridad (detección de intrusos), la automoción (conducción autónoma), entre otros.

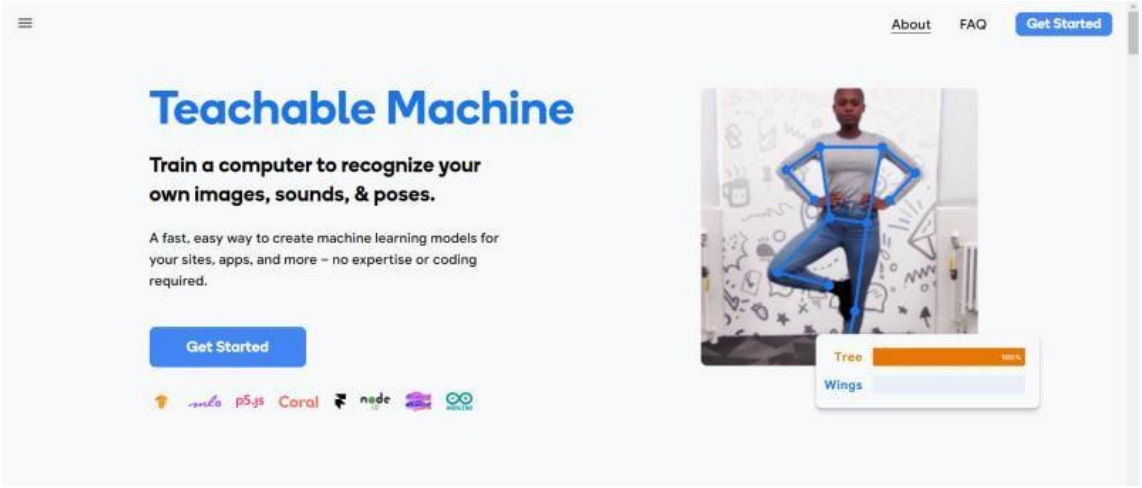
4.3 Software Teachable Machine

INTEF en el año 2018 expuso que, Teachable Machine es una herramienta en línea que permite crear modelos de aprendizaje automático de manera rápida y sencilla para tus proyectos, sin necesidad de programar. Puedes enseñar a una computadora a reconocer imágenes, sonidos y posturas, y luego exportar el modelo para usarlo en tus sitios web, aplicaciones y más. Es una herramienta de inteligencia artificial desarrollada por Google, diseñada para facilitar la creación de modelos de aprendizaje automático de forma rápida, fácil y accesible para todos. Con esta herramienta, puedes entrenar a una computadora para que reconozca imágenes, sonidos y posturas. Es una forma rápida y sencilla de crear modelos de aprendizaje automático para incorporarlas en sitios web o aplicaciones.

Como se visualiza la imagen correspondiente a la figura 3 de la página web principal de Teachable Machine.

Figura 3

Página web Teachable Machine.



Nota: Página de inicio de página web Teachable Machine (Google, 2023)

4.4 Componente de la arquitectura del sistema

4.4.1 Cámara OV5647

En la detección de signos de fatiga se requiere implementar una cámara que permita captar el rostro de una persona y realizar el procesamiento de imágenes y detección de signos de fatiga. La razón por la cual se eligió la cámara de visión nocturna Raspberry Pi es porque satisface los requisitos del sistema.

En la figura 4 se puede observar la cámara Rasberry Pi a utilizar.

Figura 4

Cámara Rasberry Pi infrarroja 5MP.

Nota: En la figura se muestra una cámara de 5 MP que será usada en el prototipo. (electronica, 2024)

4.4.2 Rasberry PI IV

Raspberry Pi es un pequeño ordenador de placa única, su tamaño es compacto, versátil y de bajo costo.

Tamaño y Componentes: La Raspberry Pi es una placa base pequeña que contiene componentes esenciales de un ordenador, como un procesador, memoria RAM, puertos USB, HDMI, conectividad Ethernet y, en algunos modelos, Wi-Fi y Bluetooth integrados. (MCI electronics, 2017)

En la figura 5 se puede apreciar el Raspberry Pi IV a usar.

Figura 5

Raspberry Pi IV.

Nota: tarjeta Raspberry Pi IV usada para la programación de la identificación de la somnolencia en conductores.

4.4.3 Alarma sonora

Una alarma es una señal o aviso audible que se emite debido a la existencia actual o próxima un evento negativo, se comunica mediante medios físicos como sirenas, campanas, megáfonos, radios, detonante, etc. (Unidad Nacional para la Gestión del Riesgo de Desastres, 2011). En la figura 6 podemos observar un tipo de alarma sonora.

Figura 6

Dispositivo de aviso acústico.

Nota: (Wikipedia, 2019)

4.4.4 Router

Un router es un dispositivo que proporciona servicios Wi-Fi y generalmente está conectado a un módem. Envía datos a dispositivos personales como ordenadores, teléfonos o tabletas mediante el uso de Internet. La red de área local (LAN) consiste en todos los dispositivos de la casa conectados a Internet. El router transmite los datos a los dispositivos después de que el módem los recopila de Internet. (Support google, 2024). En la figura 7 se logra observar el router que se usó en el prototipo.

Figura 7

Router tp-link.

Nota: (Sincables, 2024)

4.5 Librerías a usar

4.5.1 OpenCV

Open Source Computer Vision es de las más extensas bibliotecas de visión por computadora en términos de funciones poseídas, incluye más de 2500 algoritmos implementados. Para varios lenguajes de programación como Python, Java y C++ se pueden determinar varias interfaces (Marín, 2020).

Una representación visual se reduce esencialmente a una matriz convencional de Numpy que almacena puntos de datos en forma de píxeles. La calidad de la imagen mejora a medida que aumenta el número de píxeles, lo que se traduce en una mayor resolución. Se puede concebir que los píxeles son unidades de información dispuestas en una cuadrícula bidimensional, y la profundidad de un píxel se refiere a la información de color que contiene (Marín, 2020)

Para que un ordenador procese una imagen, esta debe transformarse a una forma binaria. La fórmula para calcular el número de colores o sombras en una imagen es:

Número de colores/sombras = 2^bpp (donde bpp representa bits por píxel)

En consecuencia, cuanto mayor sea la cantidad de bits por píxel, mayor será la variedad de colores posible en las imágenes. (Marín, 2020)

La siguiente tabla muestra de forma más evidente la conexión entre la cantidad de bits por píxel y la variedad de colores.

Tabla 1

Cantidad de bits por pixel.

BITS/PIXEL	POSSIBLE COLOURS
1	2^1=2
2	2262=4
3	2^3=8
4	2^4=16
8	2^8=256

16	2^16=65000

4.5.2 Librería VLC

El marco multimedia libVLC, también conocido como VLC SDK, se puede incorporar en una aplicación para aprovechar sus capacidades multimedia.

libVLC es una API multiplataforma de audio y video que ofrece una completa funcionalidad multimedia. Puede ser utilizada en dispositivos móviles, servidores y computadoras de escritorio para reproducir video, generar audio, así como codificar y transmitir contenido multimedia.

Dado que VLC se construye sobre libVLC, es posible disfrutar de las mismas características que ofrece el reproductor multimedia VLC al integrar esta biblioteca en una aplicación.

libVLC tiene la capacidad de realizar diversas funciones, que incluyen:

1. Reproducción de todos los tipos de archivos multimedia, códecs y protocolos de transmisión (RTSP, RTMP, Multicast, HLS/Dash/HDS, entre otros).
2. Funcionamiento en una amplia variedad de plataformas, desde computadoras de escritorio (Windows, Linux, Mac) hasta dispositivos móviles (Android, iOS), televisores (AppleTV, Android TV, Tizen) y consolas (PS4/5, Xbox).
3. Eficiente manejo de hardware y decodificación en todas las plataformas, incluso hasta resoluciones de 8K.
4. Navegación en red para sistemas de archivos remotos (SMB, FTP, SFTP, NFS...) y servidores (UPnP, DLNA).
5. Reproducción de Audio CD, DVD y Bluray con navegación por menú.
6. Compatibilidad con 8K y HDR, incluido el mapeo de tonos para transmisiones SDR.

7. Transferencia de audio a través de SPDIF y HDMI, incluso para códecs de audio HD como DD+, TrueHD o DTS-HD.

8. Admite filtros complejos de video y audio.

9. Compatibilidad con la reproducción de video 360/VR y audio 3D, incluido Ambisonics.

10. Capacidad para transmitir y enviar a renderizadores remotos, como Chromecast y UPnP. (VideoLAN Wiki, 2021)

4.5.3 Librería Numpy

La importación de Numpy como 'np' constituye una herramienta impresionantemente robusta y versátil de computación numérica para Python, capaz de ejecutar una variedad de operaciones matemáticas. Al optar por importar Numpy de esta manera, se desbloquea su potencial completo.

La ventaja de utilizar "Import Numpy as np" en lugar de las funciones estándar de Python radica en su capacidad para llevar a cabo operaciones extremadamente rápidas y eficientes gracias a bibliotecas precompiladas escritas en C. Esto permite realizar operaciones matemáticas comunes, como funciones de matriz, operaciones de álgebra lineal, generación de números aleatorios, polinomios y transformaciones de Fourier, de manera considerablemente más rápida que las funciones convencionales de Python.

Otro punto destacado de "Import numpy as np" es su habilidad para vectorizar el código, permitiendo la aplicación de una sola operación en varios elementos de una matriz simultáneamente. Esta capacidad simplifica significativamente el trabajo con conjuntos de datos extensos y elimina la necesidad de iterar manualmente sobre cada elemento de una matriz.

Además, "Import numpy as np" respalda la transmisión, un método que facilita la realización de operaciones aritméticas entre matrices de diversas formas, simplificando la comparación y manipulación de datos provenientes de distintos conjuntos o fuentes.

Por último, es importante destacar que "Import Numpy as np" presenta una excelente interoperabilidad con otros paquetes populares de Python, como Pandas y Matplotlib, convirtiéndolo en una herramienta invaluable para tareas de análisis de datos.

En resumen, la importación de Numpy como 'np' ofrece numerosos beneficios esenciales para tareas de computación matemática y científica, gracias a su velocidad, eficiencia y una amplia gama de capacidades. Al utilizar "Import Numpy as np", es posible llevar a cabo de manera rápida y efectiva operaciones de matriz, operaciones de álgebra lineal, generación de números aleatorios, funciones polinómicas, transformadas de Fourier, así como aprovechar la capacidad de transmisión de código vectorizado e interoperabilidad con otros paquetes, todo dentro de una biblioteca. (Uikey, 2023)

4.5.4 Librería Keras

Keras es una biblioteca Python sencilla y potente para el aprendizaje profundo. Según Brownlee (2023), Keras divide la tarea de guardar la estructura del modelo y guardar los pesos del modelo en preocupaciones separadas. Los parámetros de los modelos se almacenan en una estructura de formato HDF5. El formato de matriz en forma de cuadrícula es perfecto para almacenar matrices de números con múltiples dimensiones. La configuración del prototipo se puede describir y guardar empleando dos tipos de formatos distintos: JSON y YAML.

En esta publicación, verá tres ejemplos de cómo guardar y cargar su modelo en un archivo:

- Guardar modelo en JSON
- Guardar modelo en YAML
- Guardar modelo en HDF5

Los dos primeros ejemplos guardan la estructura del modelo y los parámetros por separado. Los pesos del modelo se guardan en un archivo de formato HDF5 en todos los casos. (Brownlee, 2023)

4.5.5 Librería Tensorflow

TensorFlow es un sistema de código abierto que permite la creación y el entrenamiento de modelos de aprendizaje automático; permite crear redes neuronales artificiales para que las máquinas las utilicen; se trata de un sistema de código abierto destinado al aprendizaje automático, es decir, la creación y gestión de modelos computacionales capaces de identificar relaciones entre datos y tomar medidas basadas en estas relaciones. (Crego, 2023)

V. MARCO METODOLÓGICO

En este proyecto se implementará un sistema de inteligencia artificial dónde se utilizará herramientas como el sitio web Teachable Machine que entrena modelos de aprendizaje basado en redes neuronales, este modelo ayudará a la detección de somnolencia en conductores por medio de fotografía de sus posturas. Python, el lenguaje de programación fue utilizado en la tarjeta Raspberry para su desarrollo, usando una cámara para detectar la postura previamente entrenada.

En la figura 8 se escoge la opción de proyecto de imagen para empezar a armar la base de datos para el proyecto. Esta opción nos permite grabar distintos escenarios con un dataset bastante amplio de imágenes.

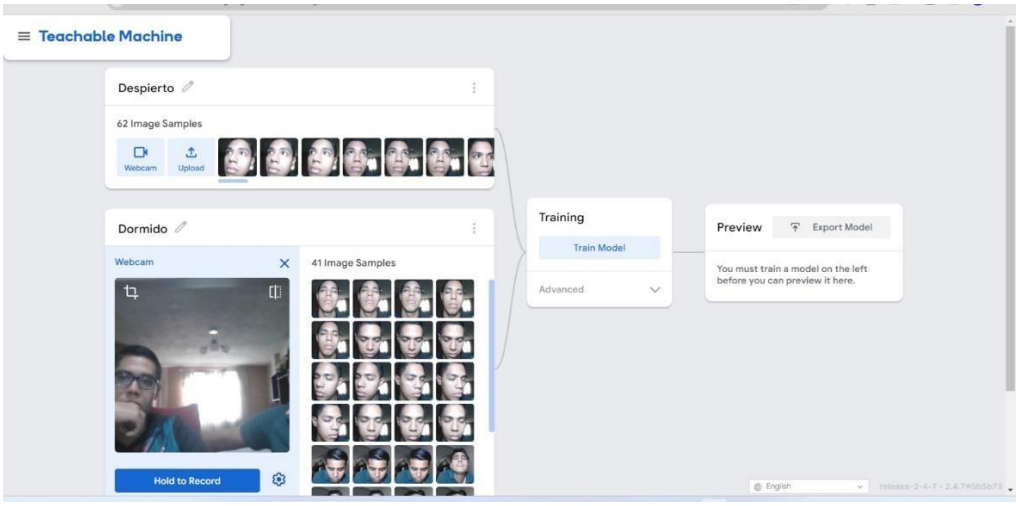
Figura 8

Proyecto de imagen Teachable Machine

Luego se realizaron dos modelados, el primero con fotos tomadas con la persona despierta y el segundo con fotos tomadas con la persona dormida, como se puede apreciar en la figura 9.

Figura 9

Capturas de base de datos usando Teachable Machine.



En el siguiente paso se realizó la vista previa con la base de datos guardada y se obtuvo el resultado de 96% de acierto con la persona despierta y 97% de coincidencia con la persona dormida, tal como se ven las figuras 10 y 11.

Figura 10

Entrenamiento persona despierta.

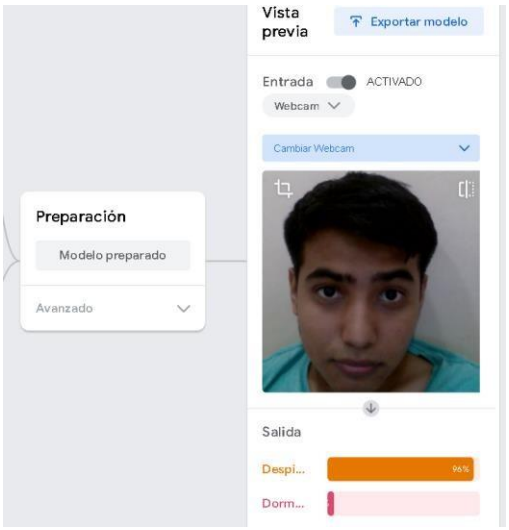
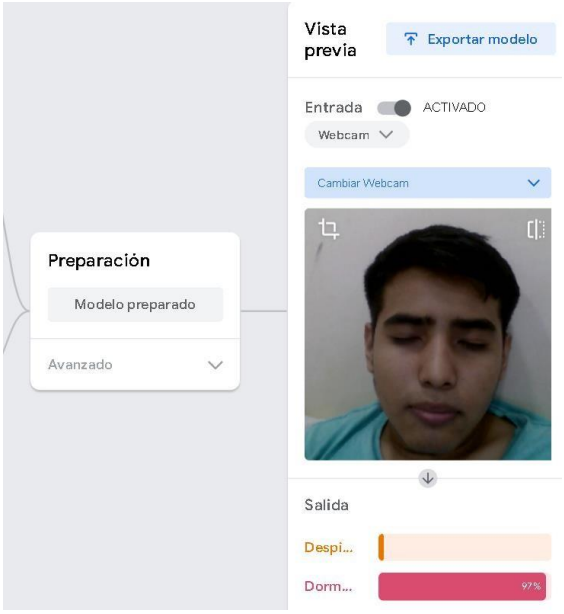


Figura 11

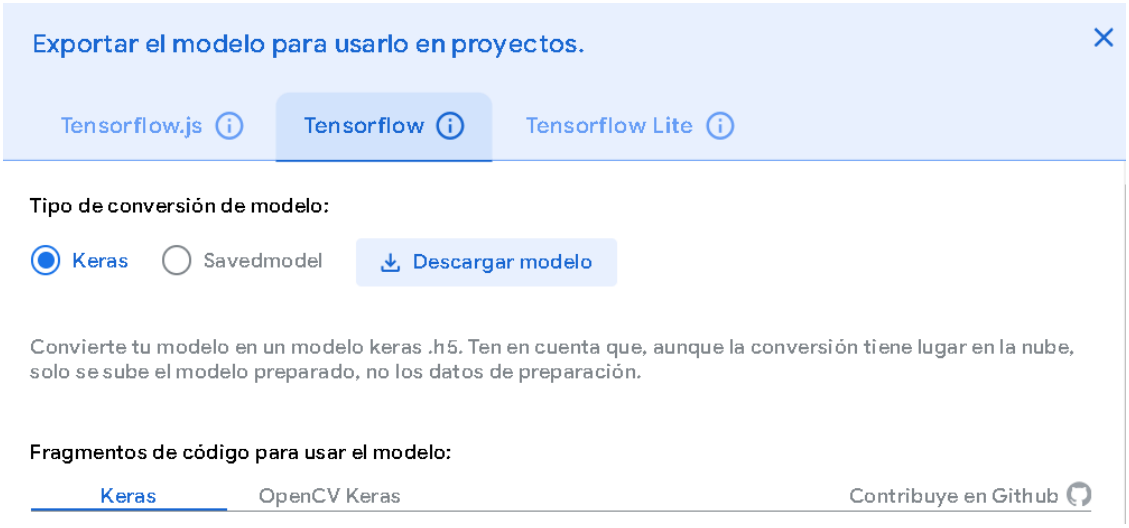
Entrenamiento persona dormida.



Seguidamente se exportó el modelo tensorflow para poder luego realizar la conversión en modelo Keras, como se puede apreciar en la figura 12.

Figura 12

Se exporta modelo Tensorflow.



Se utilizará la cámara Raspberry Pi infrarroja 5MP OV5647 que es de un tamaño muy pequeño y tiene las características para no perturbar la visibilidad del conductor.

Se realizará el enlace entre las posturas grabadas en Teachable Machine a la tarjeta Raspberry y emitirá dos alarmas; una será de tipo sonora que trabajará con un speaker mientras que la otra alarma será de tipo visual; reflejando dos imágenes de advertencia.

Según Cuervo (2023), una herramienta basada en la web llamada Teachable Machine permite crear rápidamente modelos de aprendizaje automático que son utilizados por docentes y estudiantes para explicar diversos temas de manera más interactiva. Además, funciona con Google Drive.

Este sitio web está dirigido a artistas, estudiantes, innovadores o creadores de todo tipo que tengan una idea que quieran explorar. De igual modo, no es necesario tener experiencia previa con el aprendizaje automático. Es posible crear instrucciones únicas que puedan usar la IA para reconocer cualquier sonido, movimiento o persona (Cuervo, 2023).

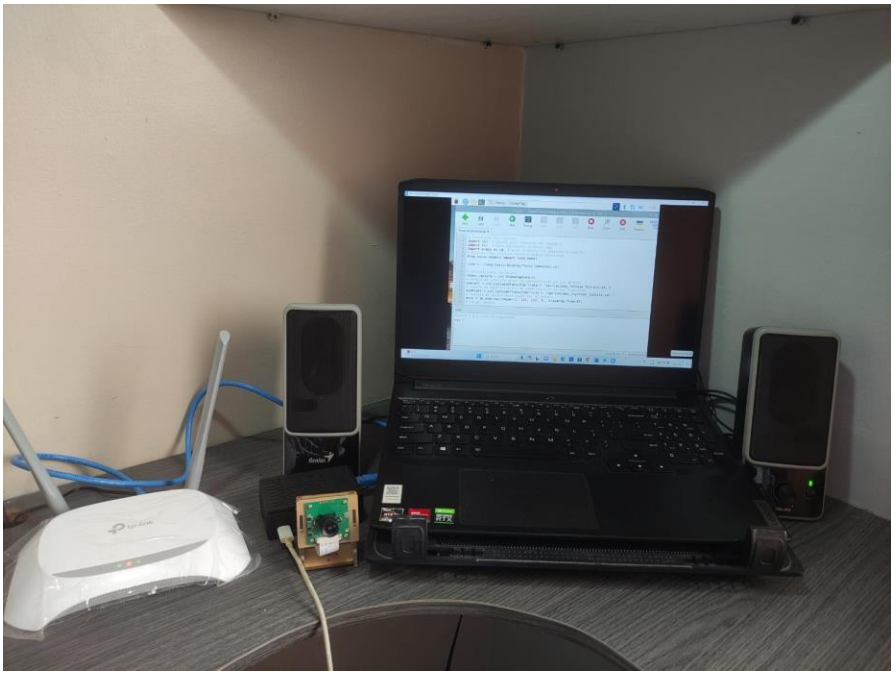
Ésta plataforma en línea recopila y agrupa los ejemplos que desea que aprenda el ordenador por clases o categorías, permitiendo preparar el modelo y probarlo para ver si puede clasificar correctamente los nuevos ejemplos. Estos proyectos también se pueden exportar para sitios web, aplicaciones y más (Cuervo, 2023).

La máquina aprendizaje utiliza archivos o captura ejemplos en vivo. Además, la organización afirma que "es una herramienta que respeta el modo de funcionamiento y puede usarse en el dispositivo, sin que ningún dato de la webcam o el micrófono salga del sitio web". Este prototipo será colocado en el tablero del auto en una posición en la cual la cámara detecte los movimientos del conductor (Cuervo, 2023).

Prototipo armado dónde se usará una laptop como pantalla de monitoreo a través de la aplicación RealVNC Viewer, bocinas para reproducción de alarma sonora y un router para tener comunicación con la tarjeta Raspberry, como se puede apreciar en la figura 13.

Figura 13

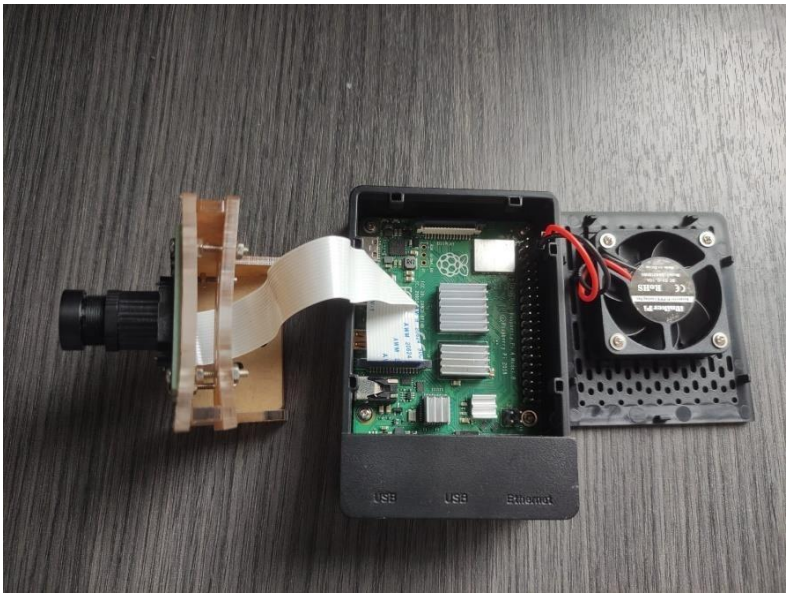
Prueba de prototipo para el sistema de detección de somnolencia.



En la figura 14 se realizó la conexión se realizó la conexión del ventilador de la tarjeta Raspberry y de la cámara conectada por medio de un cable tipo flex.

Figura 14

Conexión de flex de cámara y disipador de calor.



Se realiza la conexión entre el router y la tarjeta Raspberry para tener comunicación, por medio de un cable patch cord, de acuerdo a lo que se puede apreciar en la figura 15.

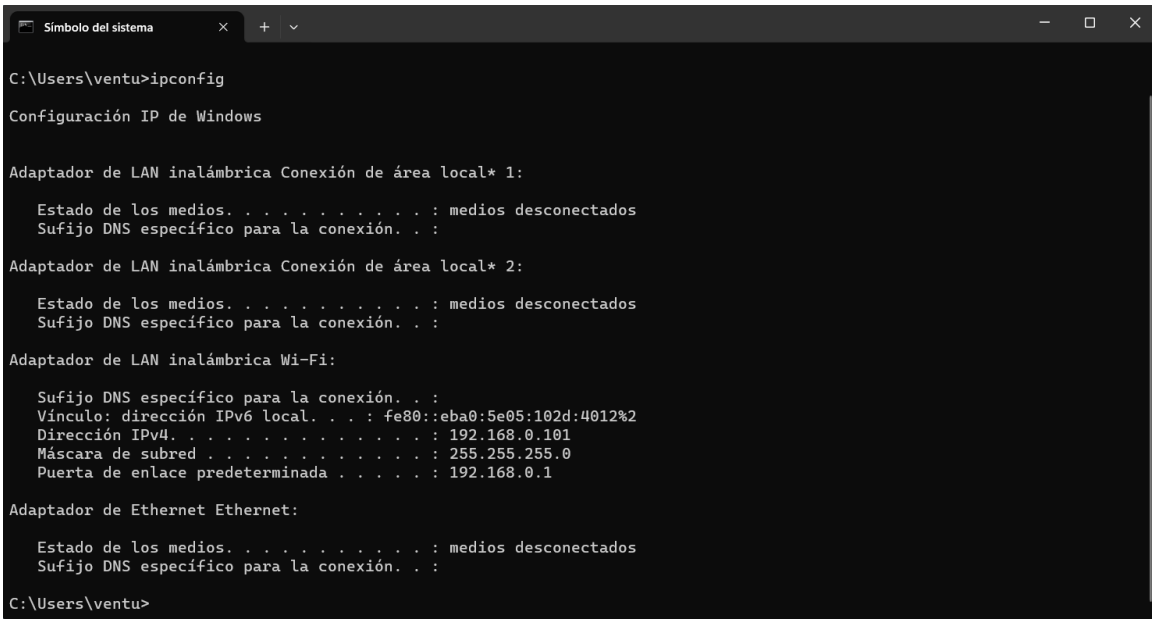
Figura 15

Comunicación de tarjeta Rasberry pi IV con router.

Se apertura la ventana cmd y revisamos la IP de nuestro equipo informático tal y como se puede apreciar en la figura 16.

Figura 16

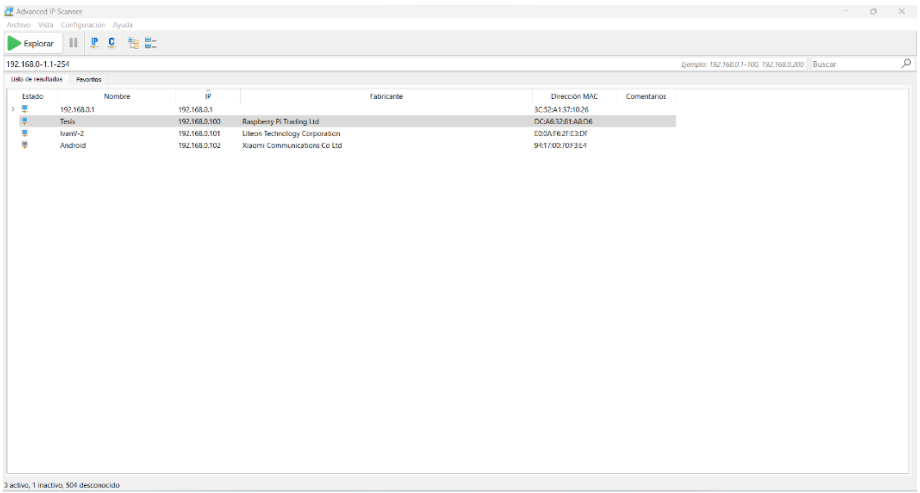
Identificar IP del ordenador que se está usando.



Luego utilizamos la aplicación Advanced IP scanner para revisar la dirección IP del router que tenemos conectado, como se puede apreciar en la figura 17.

Figura 17

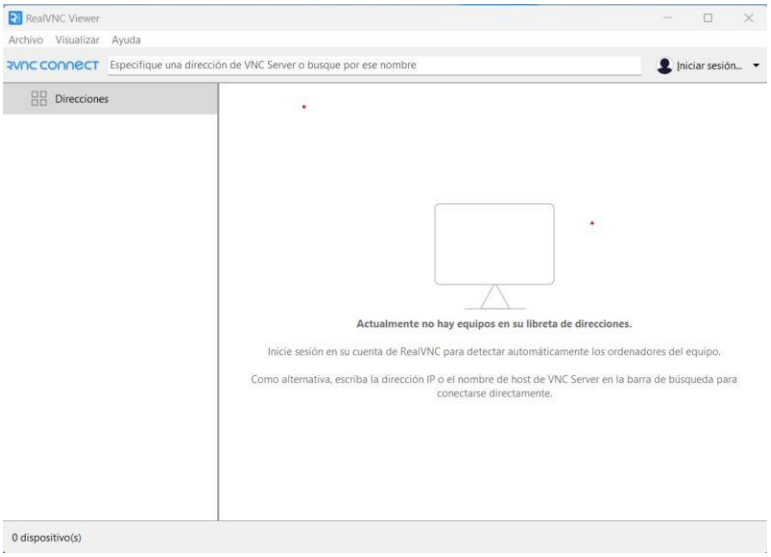
Identificar IP del router a usar.



Como paso siguiente ejecutamos la aplicación RealVNC Viewer y le damos click a la opción “Archivo”, como se visualiza el contenido en la figura 18.

Figura 18

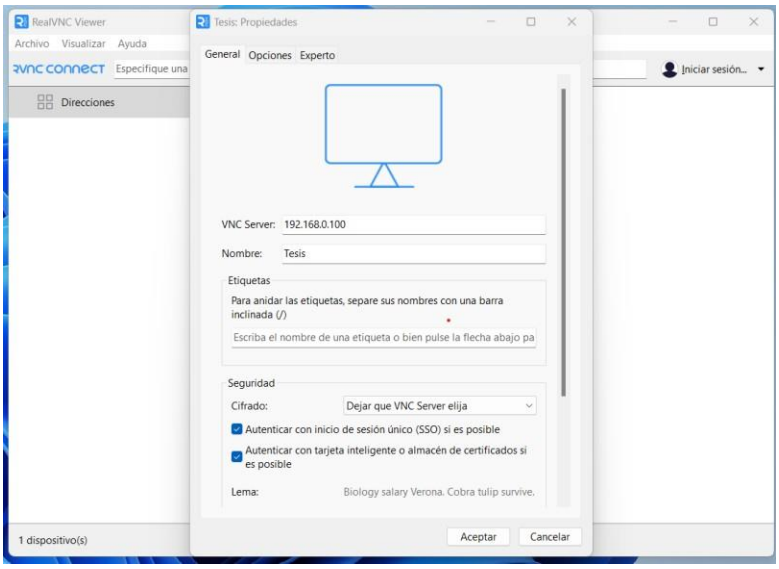
Ejecutar aplicación RealVNC Viewer.



A manera consecuente se despliega una pestaña, como se indica en la figura 19 y se llenan los campos correspondientes como lo son; dirección IP de la tarjeta Raspberry y se coloca un nombre distintivo.

Figura 19

Agregar nuevo dispositivo.



En la figura 20 se observa una ventana en la que solicita nombre de usuario y contraseña, para eso se realizó así para que solo puedan acceder a la tarjeta Raspberry y a su información todo el que cuente con la información solicitada en pantalla, para proteger la programación.

Figura 20

Iniciar sesión.

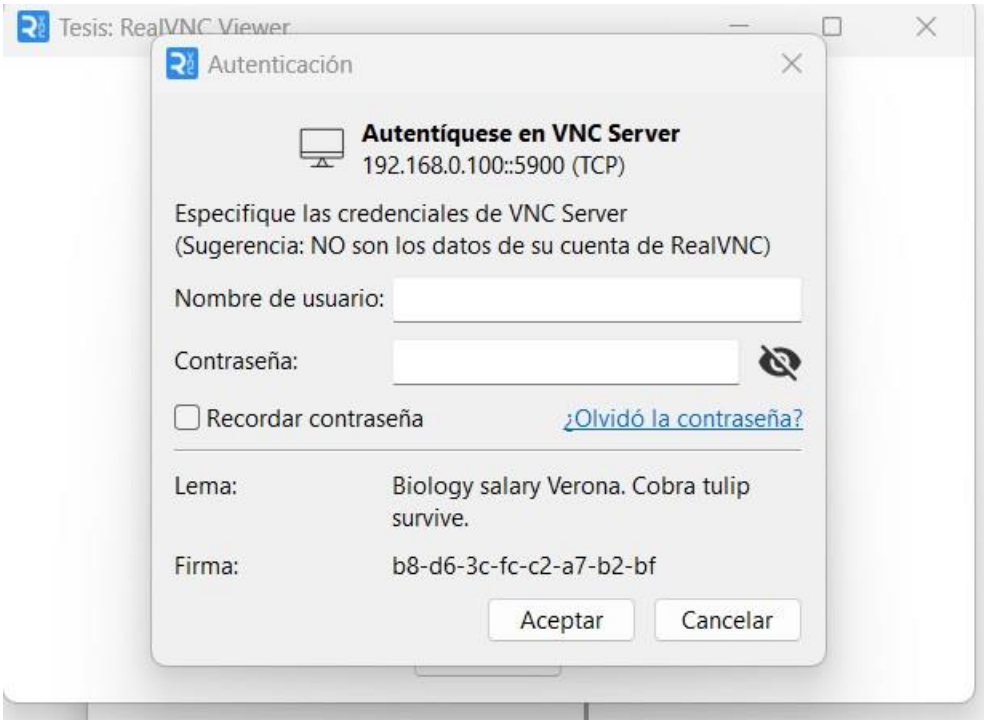
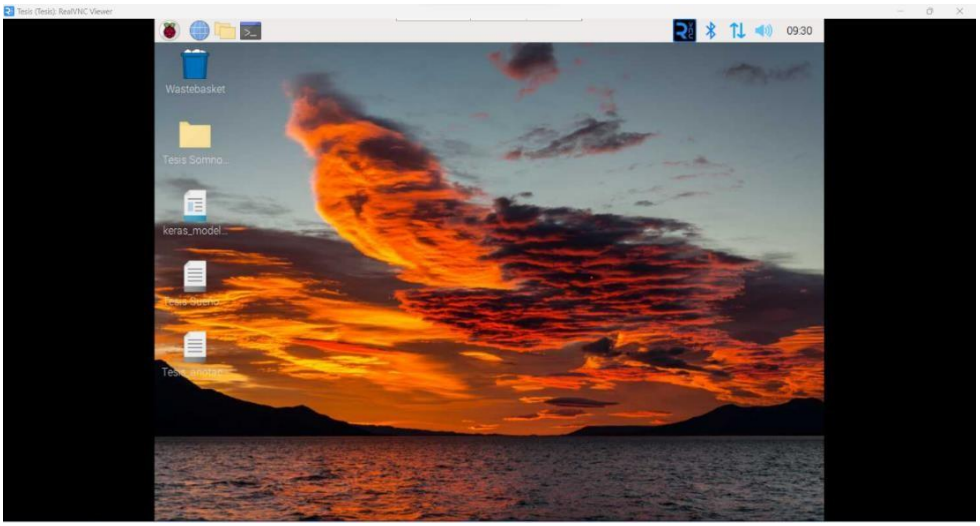


Figura 21

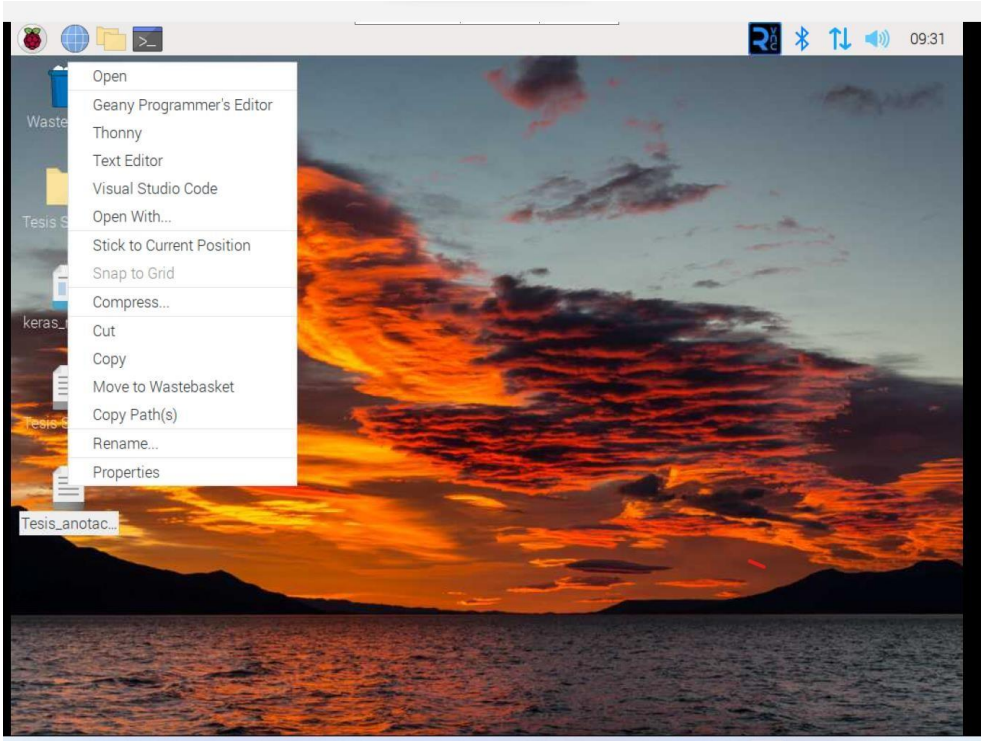
Se inicia en el sistema operativo Raspberry.



*En la figura 22 s*e abre el archivo de codificación de Python en el entorno Thonny.

Figura 22

Se abre archivo de codificación.

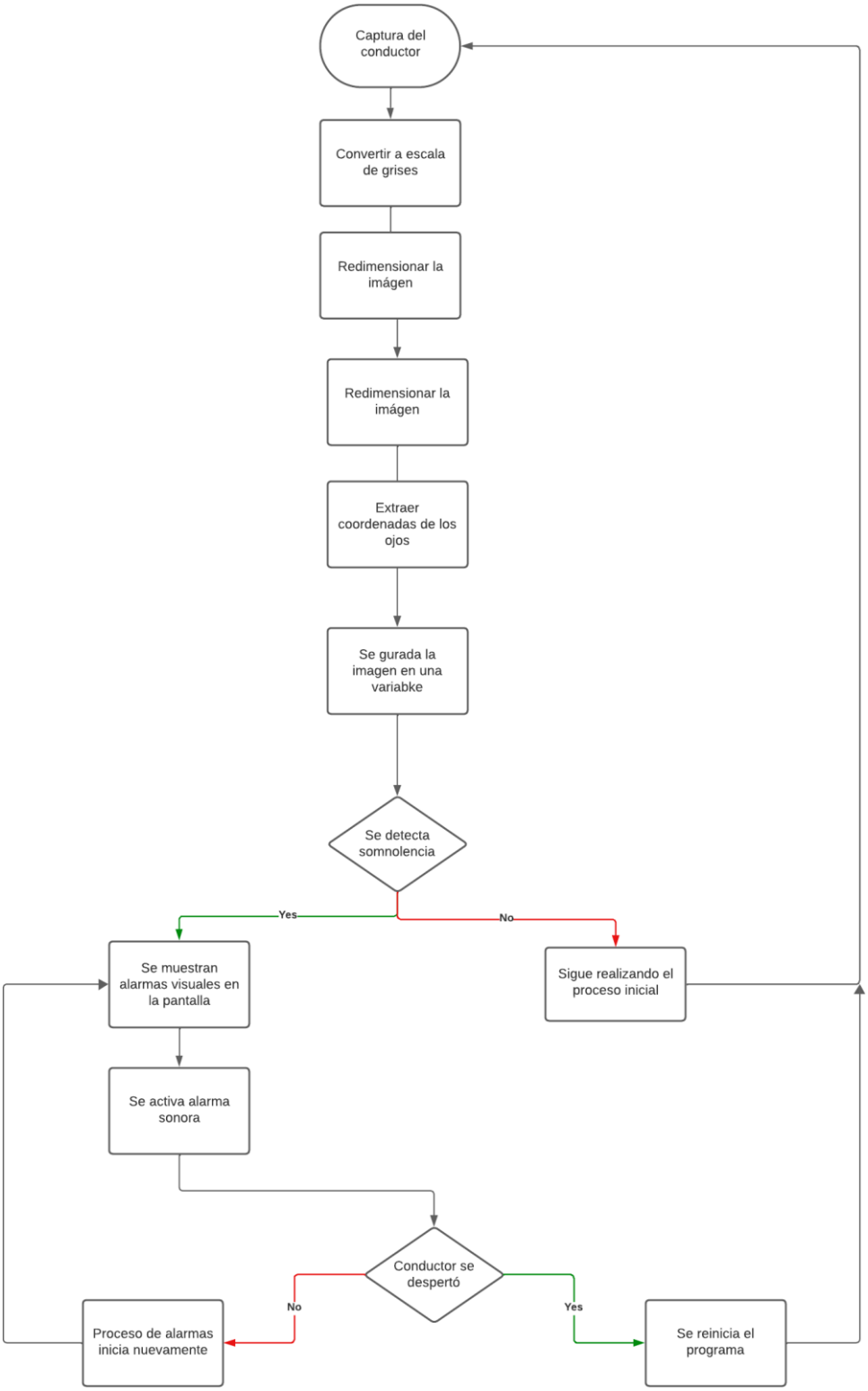


Representación gráfica del proceso en la figura 23 en el cual se realiza la codificación, empezando con la detección de las imágenes del conductor, luego realiza la escala de grises esto para agilizar el proceso de reconocimiento de imágenes; consecutivo a esto se redimensionan las imágenes, luego se extraen las coordenadas X y Y de los ojos y guarda las imágenes en una variable. El próximo paso es abrir un laso loop y realizar las interrogantes, ¿el conductor está en estado de somnolencia?, sí la respuesta es afirmativa entonces se activa la alarma sonora y se muestran en pantalla las 2 imágenes de advertencia preestablecidas. Sí la respuesta es no entonces vuelve al primer paso que es la detección de imágenes y continua con el proceso normal.

Si después de activarse las alarmas por primera vez, la cámara debe detectar que el conductor se despertó, si es así entonces vuelve al primer paso del proceso y continua con el ciclo, sino y vuelve a reincidir en la somnolencia el conductor, el código seguirá activando las alarmas hasta que el individuo este despierto por completo o salga de la vía para poder descansar y evitar un accidente de tránsito.

Figura 23

Diagrama de bloques del proceso de programación.

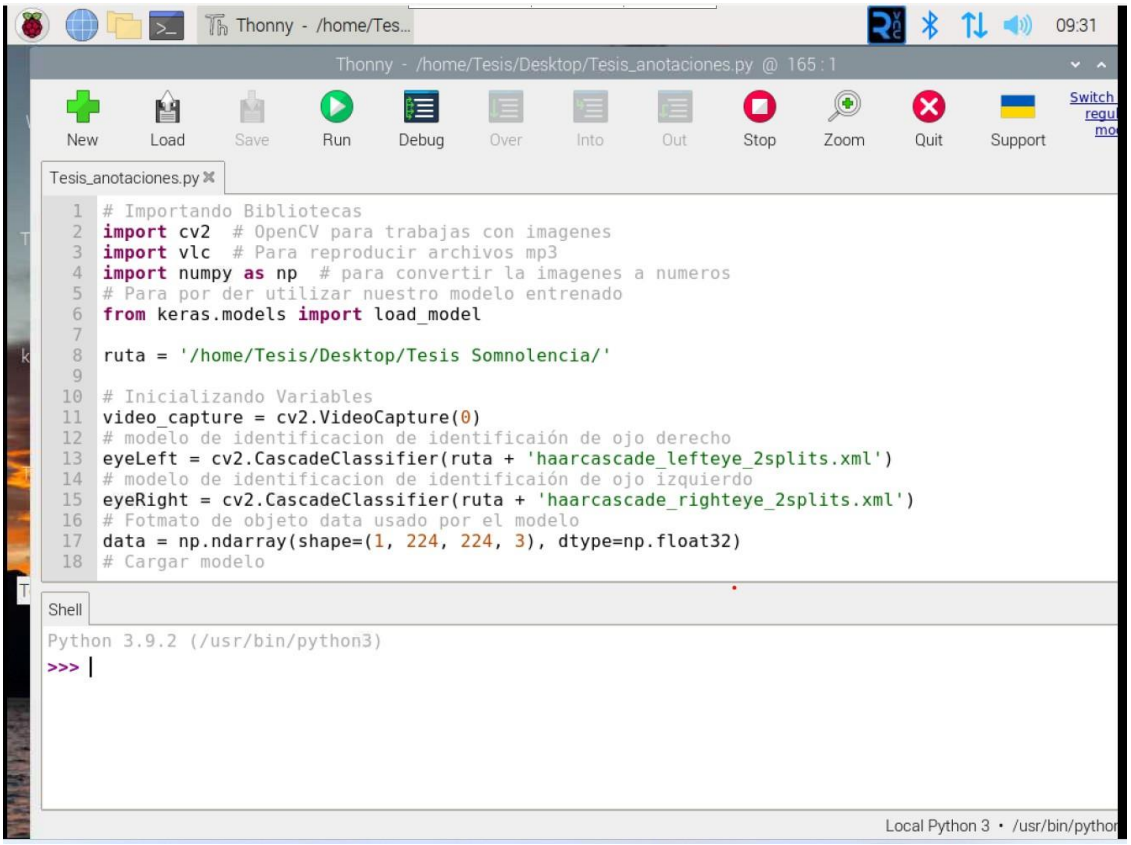


Para este proyecto se usaron 4 librerías; de las cuales son: cv2, vlc, Numpy, keras.

En la línea número 8 se colocó la ruta del archivo donde se guardó toda la codificación. A continuación, se inicializan las variables, se detalla el modelo de identificación para los dos ojos, izquierdo y derecho, como se puede apreciar en la figura 24.

Figura 24

Se importan las librerías a usar.

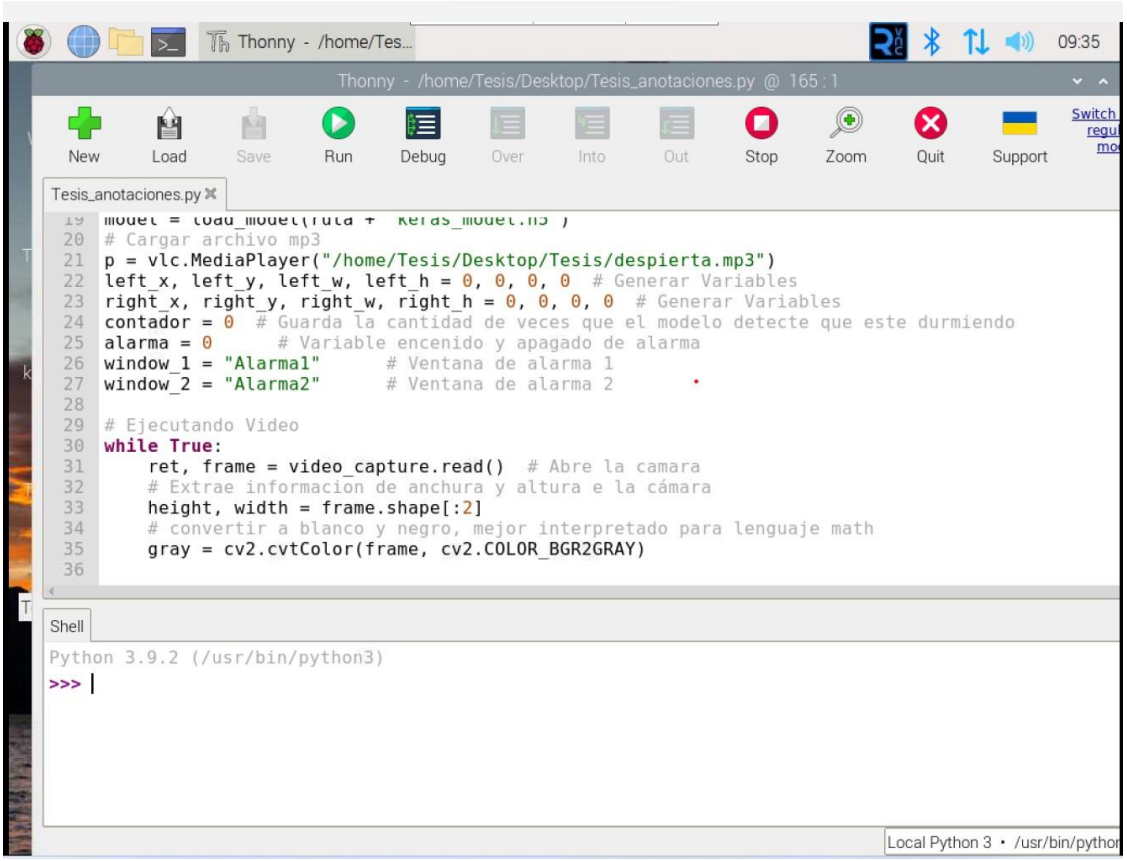


En las líneas 22 y 23 se generan las variables; luego de eso se guarda la cantidad de veces que el modelo detecte que el individuo esta durmiendo.

En la línea 25 se crea la variable de encendido y apagado de alarma. Se crean 2 ventanas que luego se visualizarán cada vez que la alarma se encienda, como se puede apreciar en la figura 23.

Figura 25

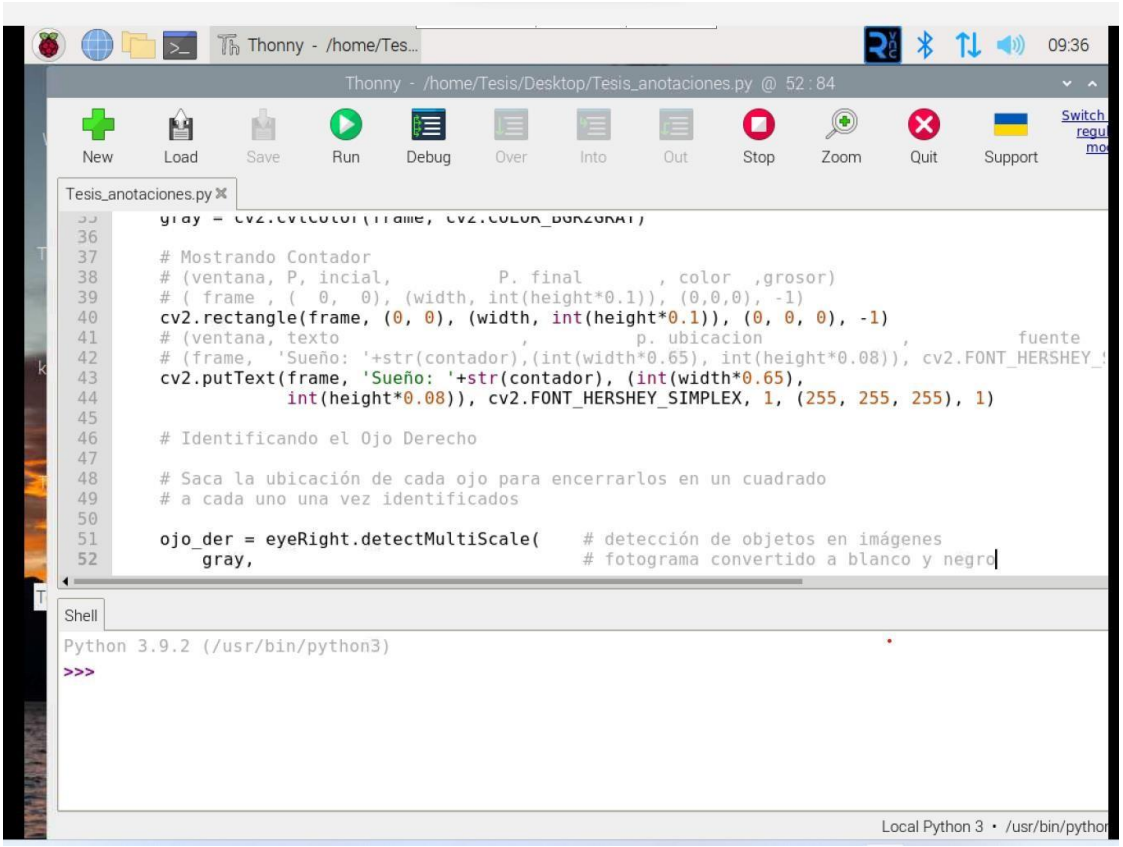
Se carga archivo de extensión .mp3.



En la línea 40 se crea un rectángulo y se colocan los parámetros de ancho y alto; en la línea 43 se coloca una etiqueta en el rectángulo “Sueño”; se convierte la variable contador a string, explica el contenido visual de la figura número 26.

Figura 26

Se exhibe el contador de somnolencia.



En la línea 51 de la figura 27 se observa cómo se realiza la detección de objetos en imágenes; luego se convierten los fotogramas a blanco y negro para que la cámara tenga una mayor velocidad al momento de detectar las imágenes.

Figura 27

Se realiza la detección de los ojos.

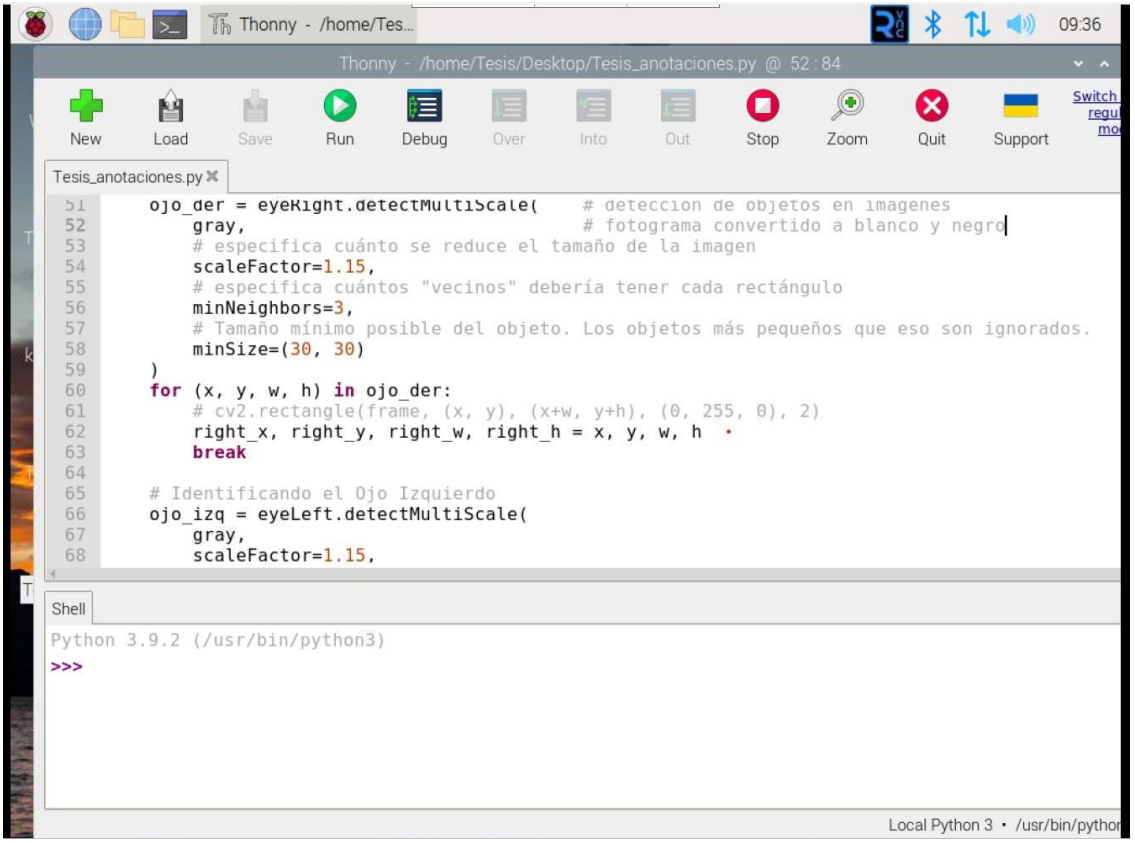
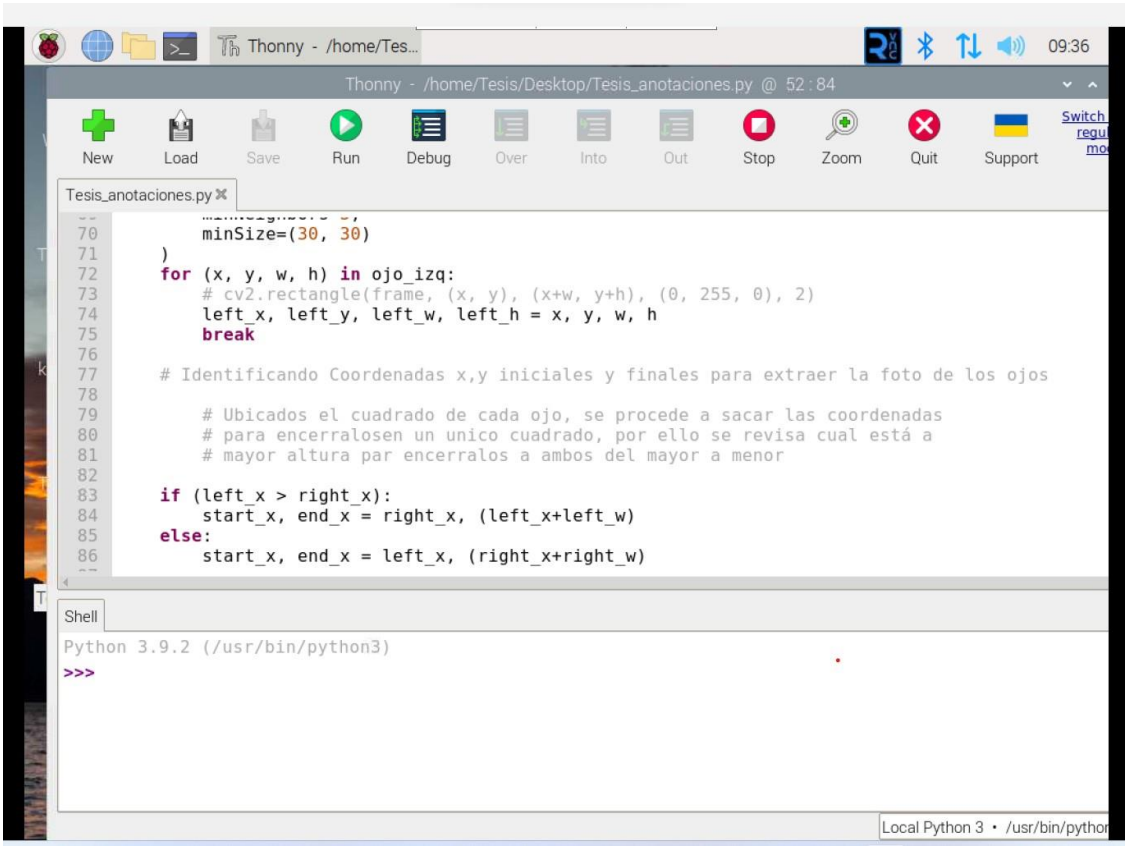


Figura 28

Identificando coordenadas X y Y.

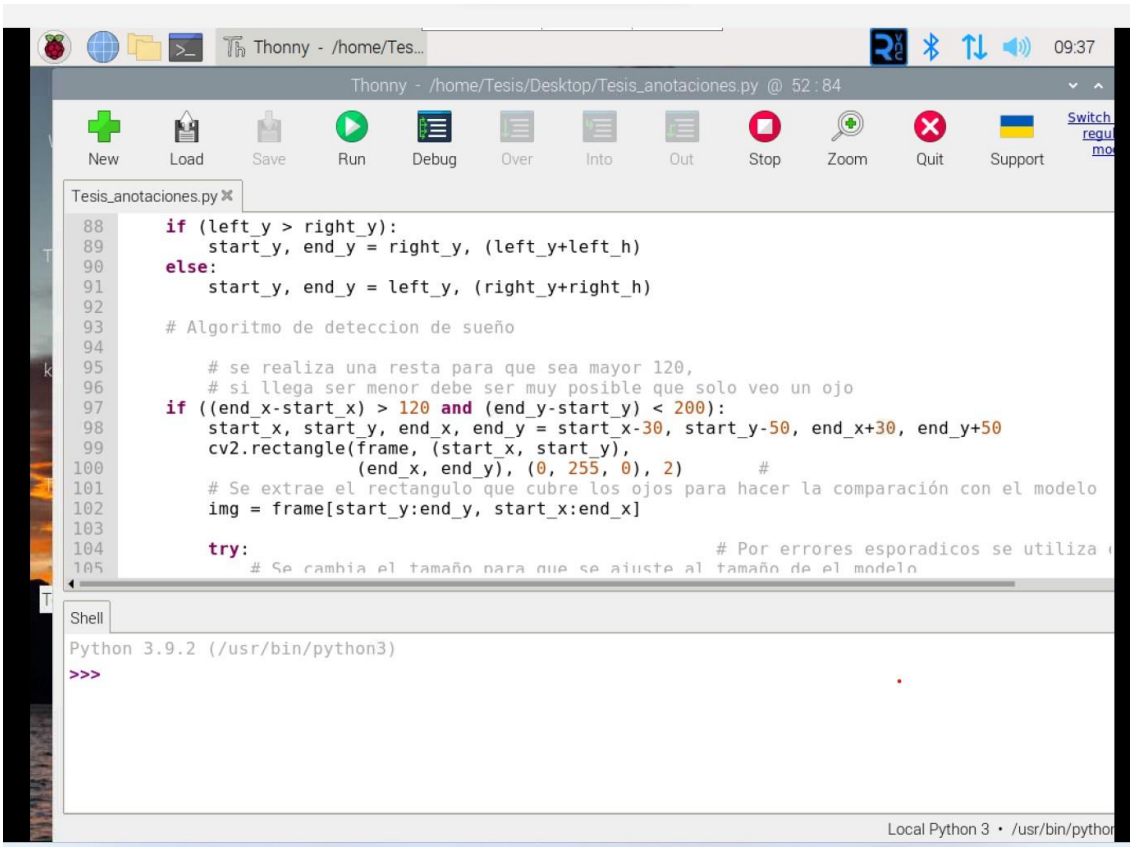


Nota: Se identifican las coordenadas X y Y iniciales y finales para extraer la foto de los ojos.

En la línea 83 se abre un bucle para ubicar un cuadrado en cada ojo, se procede a obtener las coordenadas para embarcarlos en un solo cuadro; por tal motivo es que se revisan los 2 ojos para saber cuál de los dos está a mayor altura con respecto al otro y de esta manera que el rectángulo general este alineado.

Figura 29

Se realiza la codificación del algoritmo de detección de sueño.



Nota: Se realiza una resta para que el tamaño del rectángulo principal sea mayor a 120; puesto que si llega a ser menor de este valor el algoritmo solo detectará un solo ojo.

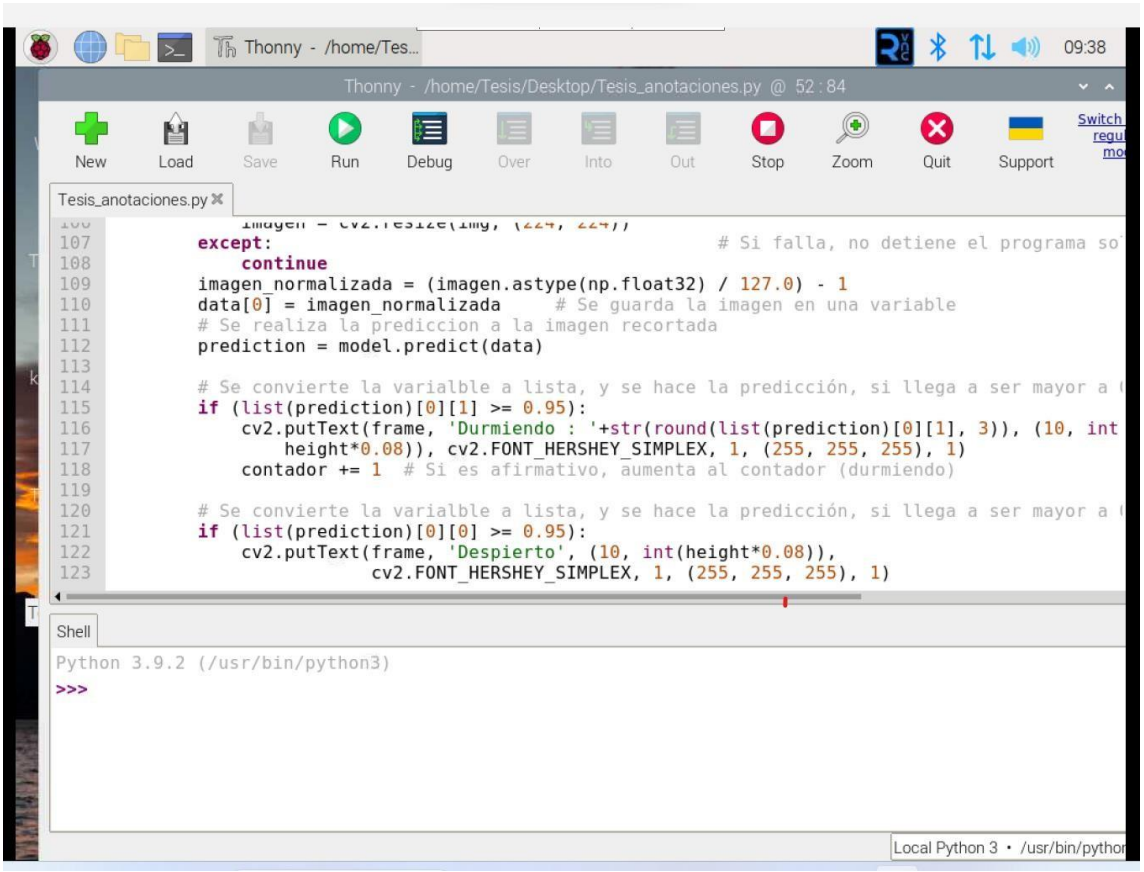
En las líneas 109 y 110 se crea una variable para que la imagen tomada sea guardada; luego se realiza la predicción a la imagen.

En la línea 115 se convierte la variable a lista, y se hace la predicción, si este valor es mayor o igual a 1 o llega a ser mayor que 0.95 entonces se dice que este dato es válido, el contador durmiendo aumenta.

En la línea 120 se realiza el mismo procedimiento solo que esta vez el contador despierto tiene la función de que si la imagen captada es menor o igual a 1 o llega a 0.95 el contador despierto aumenta, como se puede apreciar en la figura 30.

Figura 30

Se guarda la imagen en una variable.

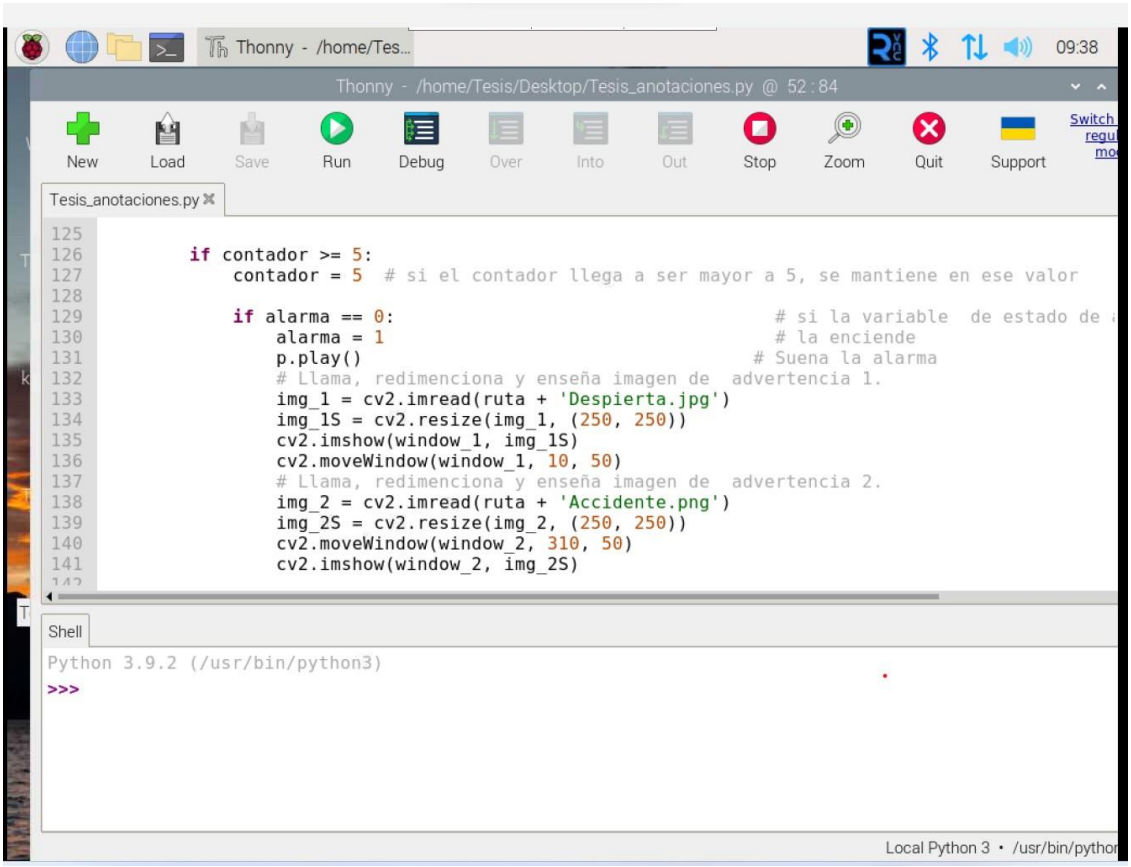


Si el contador durmiendo es mayor o igual a 5 (fotogramas captados), el estado de la alarma cambia de apagado (0) a encendido (1).

La figura 31, representa en la línea 133 se llama a la imagen de advertencia 1 desde la ruta guardada en la tarjeta Raspberry; hace el redimensionamiento de la imagen y la muestra en una pestaña, luego de esto se realiza la misma operación, pero esta vez para la pestaña advertencia 2.

Figura 31

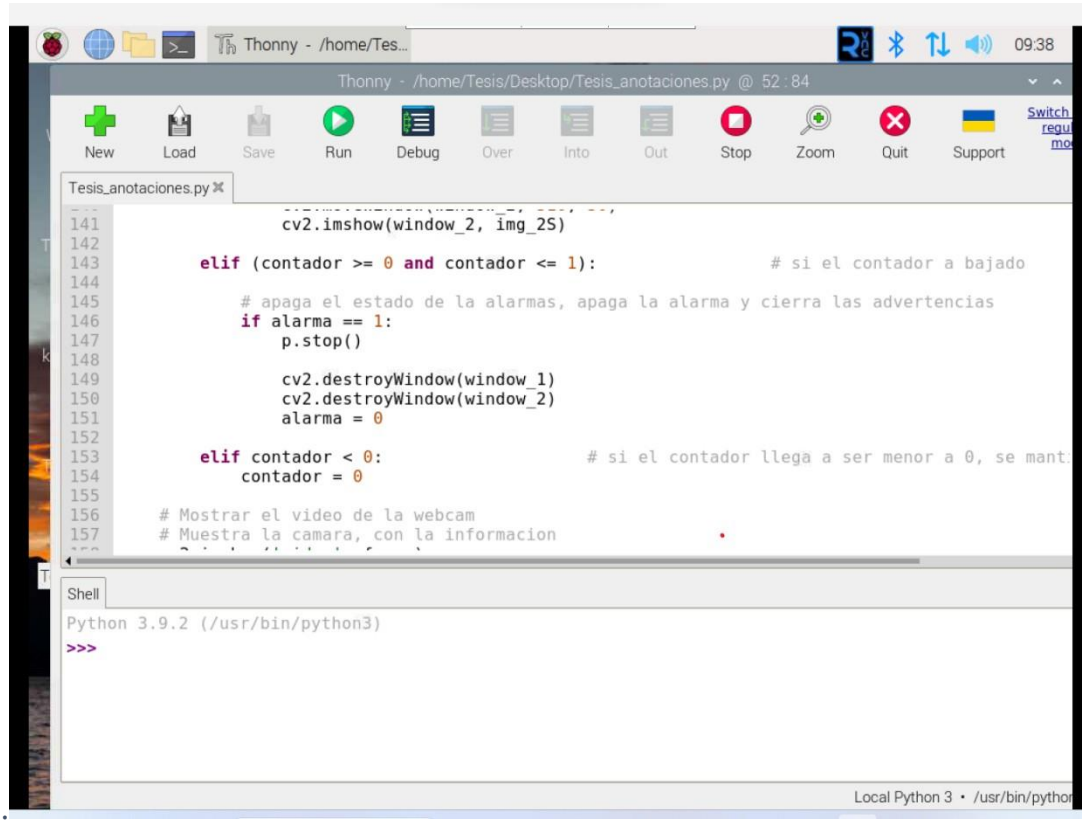
Activación de alarma visual y sonora.



Si el contador ha bajado, apaga el estado de las alarmas, apaga la alarma sonora y cierra las pestañas de advertencia, tal como se aprecia en la figura 32.

Figura 32

Se apaga la alarma y se resetea en contador.



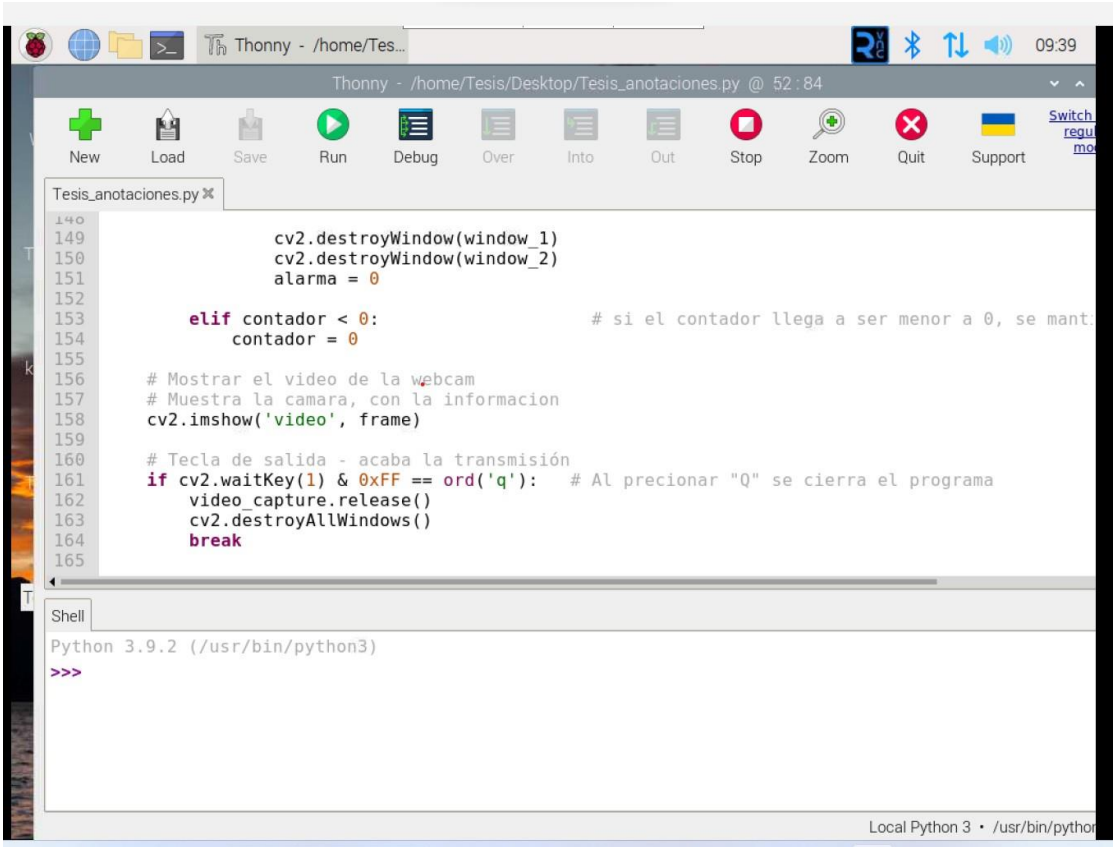
Si el contador llega a ser menor a 0, mantiene ese valor.

En la línea 158 se muestra el vídeo de la cámara Raspberry con la información del estado de sueño o de despierto, como se representa en la figura 33.

Finalmente se presiona tecla de salida en la pestaña de la cámara y se detiene la transmisión, luego al presionar la tecla Q se cierra el programa.

Figura 33

Algoritmo comienza a detectar nuevamente.


Thonny - /home/Tes...
09:39
Thonny - /home/Tesis/Desktop/Tesis_anotaciones.py @ 52 : 84
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Support
Tesis_anotaciones.py
cv2.destroyWindow(window_1)
cv2.destroyWindow(window_2)
alarma = 0
elif contador < 0:
# si el contador llega a ser menor a 0, se mant
contador = 0
# Mostrar el video de la webcam
# Muestra la camara, con la informacion
cv2.imshow('video', frame)
# Tecla de salida - acaba la transmisión
if cv2.waitKey(1) & 0xFF == ord('q'):
# Al precionar "Q" se cierra el programa
video_capture.release()
cv2.destroyAllWindows()
break
Shell
Python 3.9.2 (/usr/bin/python3)
>>>
Local Python 3 • /usr/bin/python

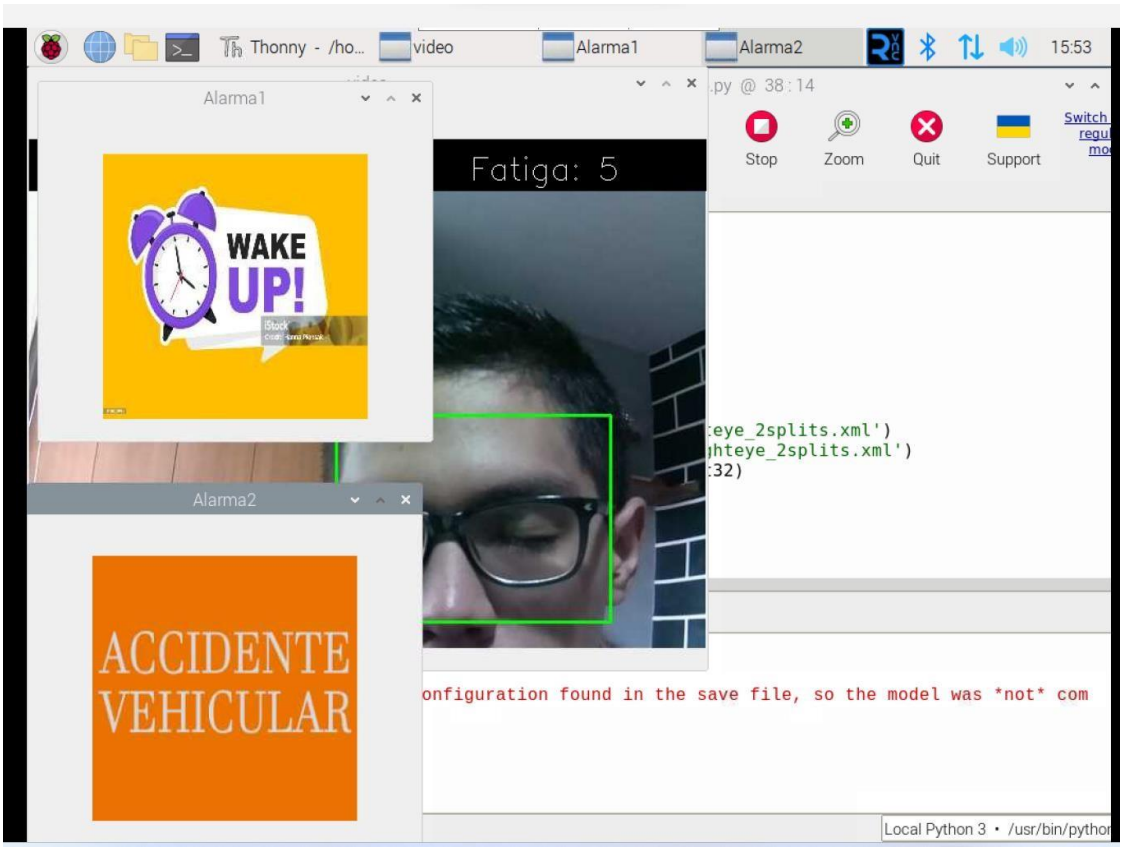
VI. RESULTADOS

En esta sección se muestran los resultados obtenidos en el proyecto, se presentan las gráficas y capturas de pantallas tomadas durante la puesta en marcha del prototipo; se plantearon escenarios con un sujeto de prueba simulando el estado de somnolencia y también despierto, obteniendo resultados satisfactorios.

Cuando el prototipo detecta 5 fotogramas del sujeto de prueba con los ojos cerrados, se considera que la persona está en estado de somnolencia y se activan los mensajes de alerta como la alarma sonora, como se visualiza el contenido en la figura 34.

Figura 34

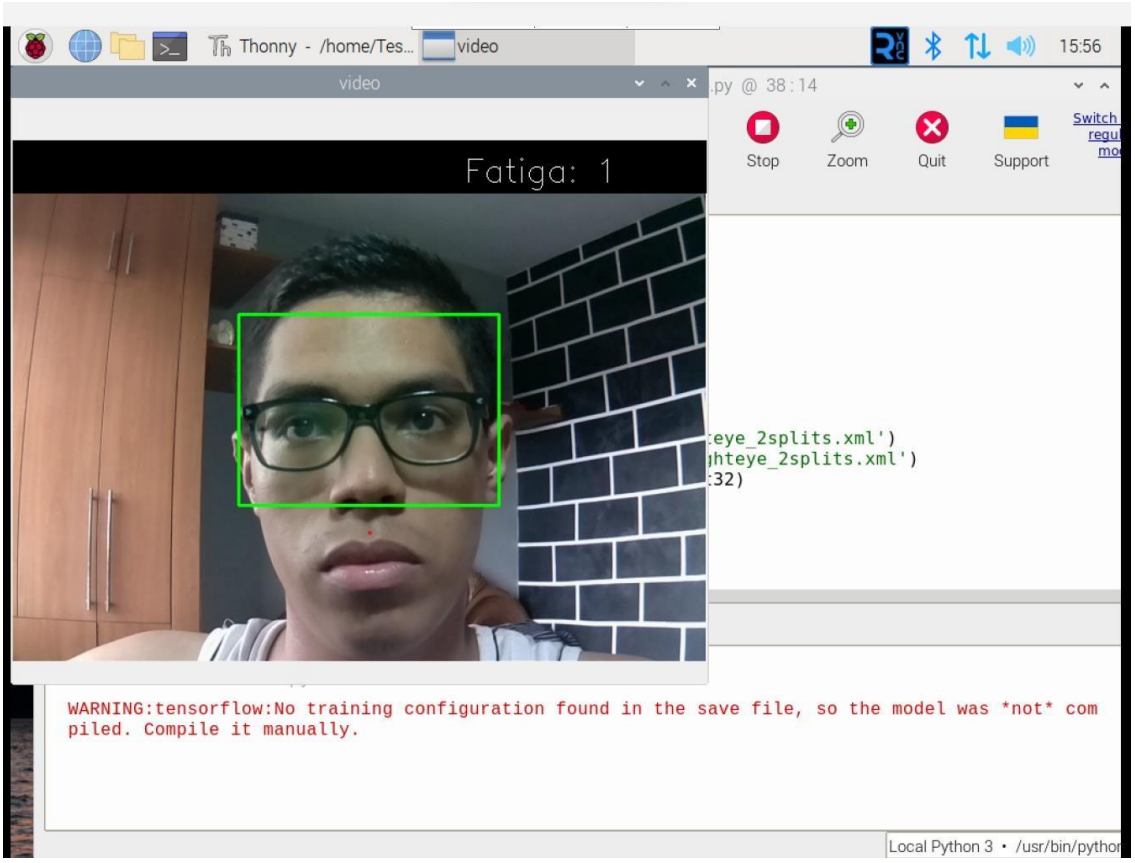
Algoritmo detectó fatiga.



En la figura 35 se puede apreciar que el prototipo detecta 1 fotograma de somnolencia, esto quiere decir que el individuo se encuentra despierto por lo tanto algoritmo no activa la alarma sonora ni la de la interfaz visual.

Figura 35

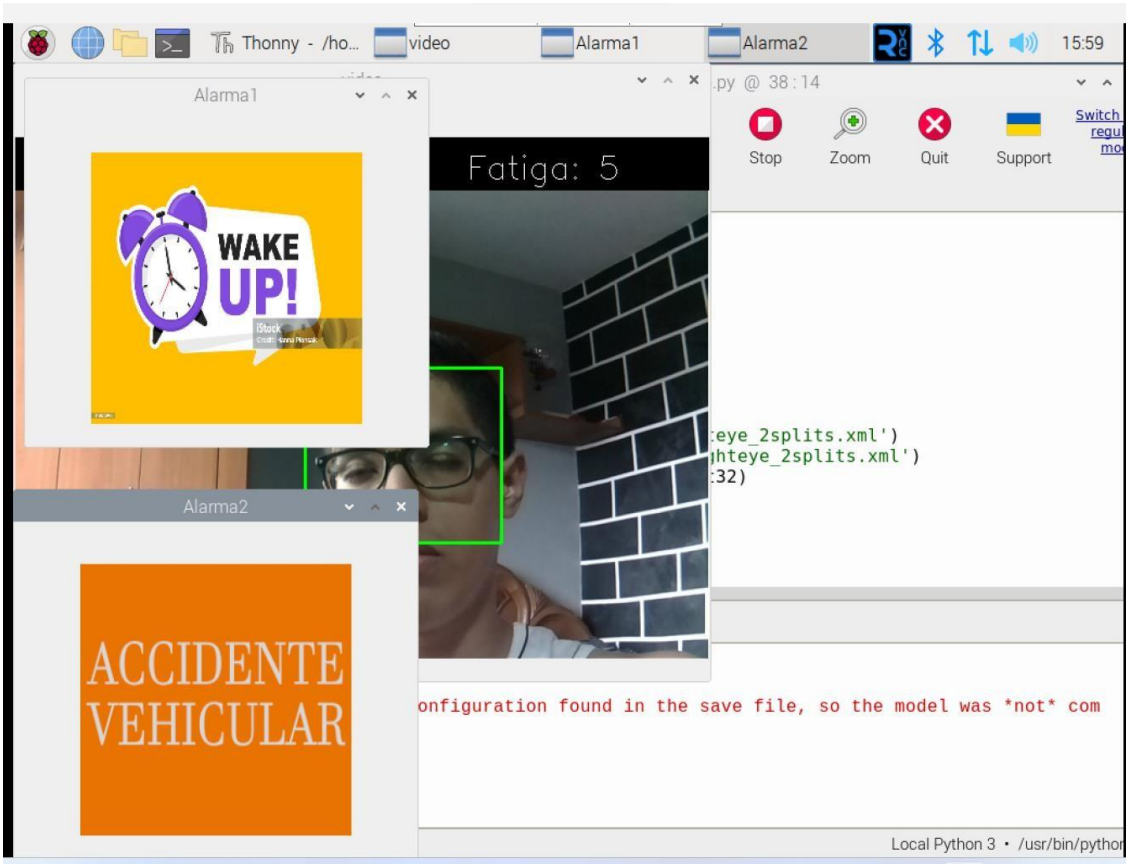
Algoritmo detecta persona con poca somnolencia.



En la figura 36 se puede observar que el algoritmo detecto 5 fotogramas que indican que el sujeto de prueba esta en estado de somnolencia, por lo tanto se procede a activar alarmas sonora y visuales.

Figura 36

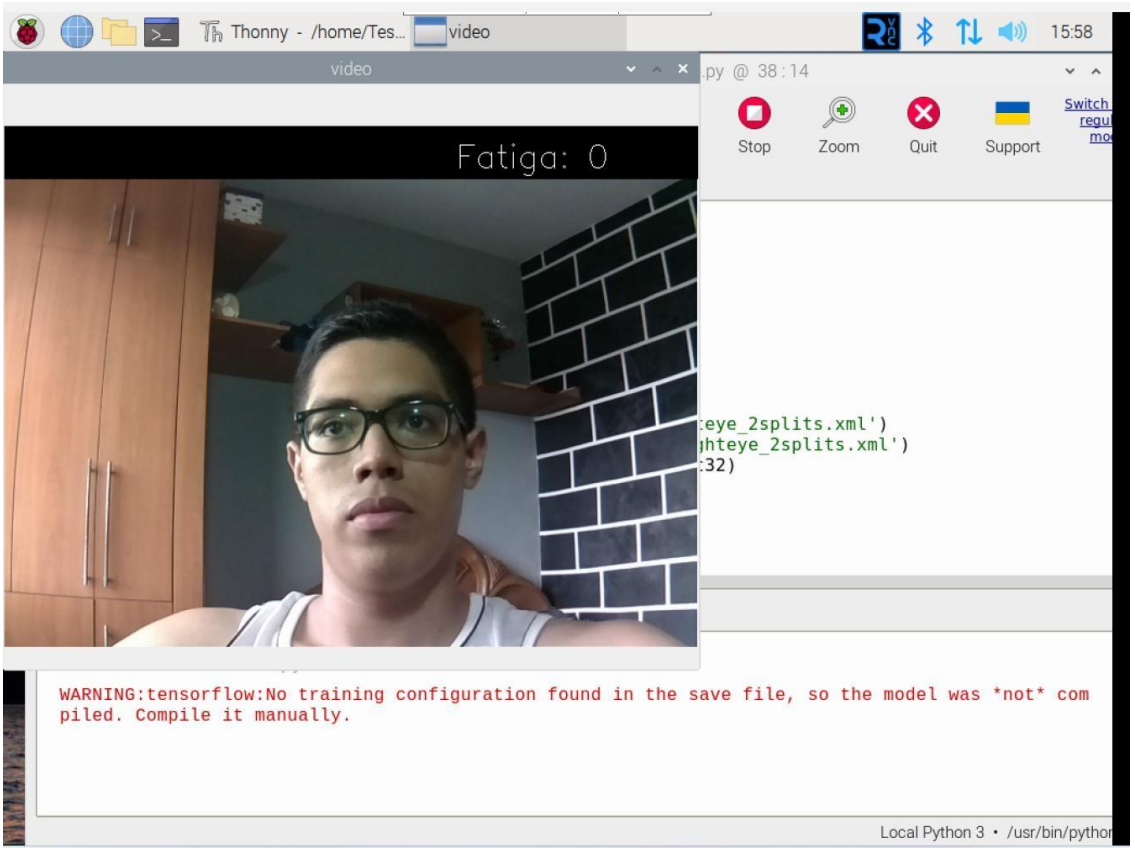
Se detecta fatiga en el individuo.



Sujeto de prueba escucha la alarma sonora y se despierta, luego contador se reinicia y comienza a testear los fotogramas nuevamente, de acuerdo a lo que se puede ver en la figura 37.

Figura 37

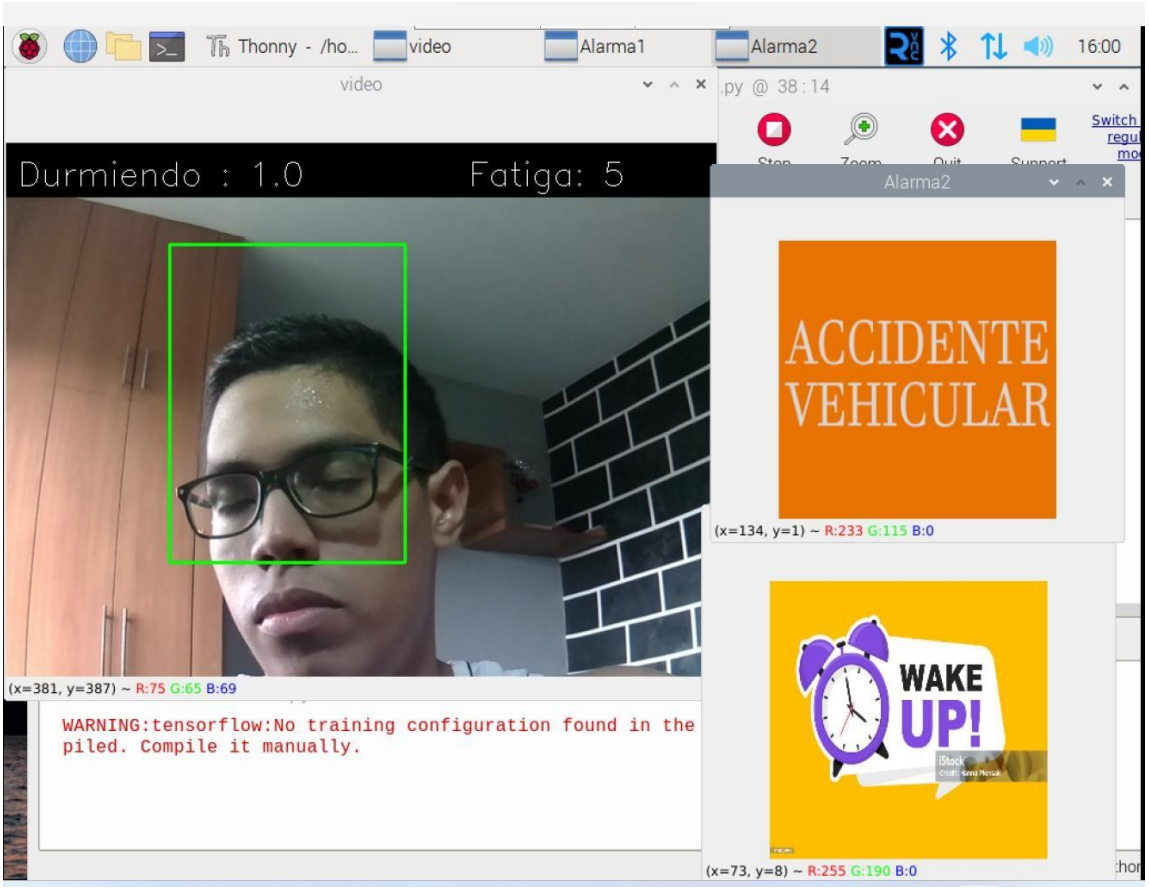
Reseteo de contador.



Algoritmo detecta satisfactoriamente a individuo de prueba en estado de somnolencia, por consiguiente, se activan las alarmas visuales y sonoras, como se puede apreciar en la figura 38.

Figura 38

Algoritmo detecta sujeto con somnolencia.



6.1 Análisis de resultados

Los hallazgos obtenidos revelan aspectos importantes sobre la eficacia y limitaciones del modelo desarrollado para la identificación de los signos de adormecimiento en conductores. Los hallazgos indican una alta fiabilidad del algoritmo, especialmente en condiciones de iluminación media, donde demostró ser eficiente tanto para conductores somnolientos como para conductores despiertos. Las alarmas se activaron de manera efectiva, y la capacidad del sistema para reiniciar la detección cuando el conductor se despertó es un indicador positivo de su funcionalidad en situaciones prácticas.

No obstante, es crucial abordar ciertas limitaciones identificadas durante las pruebas. La investigación de Alba Neppas (2020), respalda la rapidez del prototipo, con un tiempo de respuesta de 0.82 segundos para detectar señales de somnolencia y activar

las alarmas correspondientes. A pesar de esta eficiencia, se observa una vulnerabilidad del algoritmo en imágenes con saturación de colores, donde la detección de rostros de conductores se ve comprometida. Este aspecto señala un área de mejora para futuras iteraciones del prototipo.

Adicionalmente, los resultados obtenidos en relación con la investigación de Baiza Lovato (2020) sugieren que el prototipo presenta cierta variabilidad en su tiempo de respuesta, particularmente en entornos reales. Sin embargo, la mejora en el tiempo de respuesta en entornos controlados es alentadora y respalda la efectividad del sistema en ambientes más predecibles.

VII. CRONOGRAMA

En la Tabla 2 se muestra la propuesta del plan de actividades organizado por semanas, abarcando un total de 20 semanas de trabajo, para la realización del Proyecto de Titulación durante los meses de noviembre a enero, correspondientes al período académico 63.

Tabla *2*.

Cronograma de actividades para llevar a cabo el desarrollo del Proyecto de Titulación

Tareas realizadas durante el anteproyecto	NOVIEMBRE				DICEMBRE				ENERO		
	Semanas										
	1	2	3	4	1	2	3	4	1	2	3
Entrenamiento de modelo de Teachable Machine.											
Desarrollo de codificación para el control de la tarjeta Raspberry Pi IV.											
Armado y pruebas de prototipo.											
Revisiones con el tutor y validación del tema de Titulación.											

Presentación del progreso del documento al tutor (primera revisión).	
Presentación del progreso del documento al tutor (segunda revisión).	
Culminación del documento de Titulación.	

VIII. PRESUPUESTO

Se presenta a continuación en la tabla 3, el presupuesto utilizado para el proyecto de un prototipo de identificación de somnolencia para conductores.

Tabla 3

Presupuesto de proyecto

DESCRIPCIÓN	**TOTAL**
1 Raspberry Pi IV	$ 100,00
1 cámara OV5647	$20,00
1 módulo de audio	$25,00
1 soporte para prototipo en vehículo	$15,00
1 Router	$30,00
1 cargador para Raspberry	$10,00
1 cable patchcord cat 6	$5,00
Herramientas de trabajo como: cautín, pinzas, desarmadores, alicates, multímetro, etc.	$50,00
TOTAL	$255,00

IX. CONCLUSIONES

Este proyecto tuvo como objetivo la detección de somnolencia en conductores, de mano de un algoritmo de predicción.

A través de este prototipo se logró que la cámara detecte a un sujeto de prueba con los ojos cerrados inmediatamente active una alarma visual y una alarma sonora, para que el sujeto de prueba vuelva a estar consciente y evitar un accidente vehicular.

Los resultados de las pruebas realizadas con la base de datos y la cámara de la tarjeta Raspberry demostraron que el algoritmo, aunque es un prototipo experimental, presenta un buen rendimiento, esto significa que es confiable y puede usarse como un sistema experimental para desarrollar más pruebas que mejoren los tiempos de respuestas y detección de somnolencia en ambientes con falta de iluminación.

X. RECOMENDACIONES

Considerando la importancia que tienen los accidentes vehiculares en la vida diaria y en función de los hallazgos obtenidos se exponen las pròximas sugerencias, esto con la finalidad de contribuir a la seguridad en las carreteras tanto para los conductores como para los peatones, para eso se hacen presentes las siguientes recomendaciones:

- Enriquecer el algoritmo de identificación de cansancio en conductores, con una base de datos más amplia.
- Utilizar una cámara de mayor resolución para permitir que el algoritmo trabaje sin dificultad en entornos de poca luz.
- Elaborar una conexión entre la tarjeta Raspberry y el router de forma inalámbrica, con el fin de que el prototipo sea más fácil de montar en un vehículo.

X. REFERENCIAS

AAA Foundation for Traffic Safety. (2016). *Prevalence of Drowsy-Driving Crashes.* Obtenido de Estimates from a Large-Scale Naturalistic Driving Study: https://cdn.cnn.com/cnn/2018/images/02/07/drowsy.driving.aaa.pdf

Agencia Nacional de Tránsito. (2024). *Estadísticas siniestros de tránsito*. Obtenido de Visor de Siniestralidad Nacional : https://www.ant.gob.ec/estadisticas-siniestros-de-transito-prueba/#.VsfKwuYgVj8

Brownlee, J. (26 de septiembre de 2023). *Machine Learning Mastery*. Obtenido de How to Save and Load Your Keras Deep Learning Model: https://machinelearningmastery.com/save-load-keras-deep-learning-models/

Crego, A. P. (21 de agosto de 2023). *DEUSTO FORMACIÓN* . Obtenido de TensorFlow: lo que debes saber: https://www.deustoformacion.com/blog/programacion-y-tic/libreria-tensorflow

Cuervo, S. (25 de Febrero de 2023). *Infobae*. Obtenido de Esta web analiza en tiempo real movimientos y voz para crear una inteligencia artificial: https://www.infobae.com/tecno/2023/02/25/esta-web-analiza-en-tiempo-real-movimientos-y-voz-para-crear-una-inteligencia-artificial/

Cultura geek. (10 de noviembre de 2017). *Gadgets y Hardware, tecnoticias*. Obtenido de Ford: ¿cómo funciona el sistema de mantenimiento de carril con detector de fatiga?: https://culturageek.com.ar/ford-sistema-carril-detector-fatiga/

electronica, E. (2024). *Electronica estudio*. Obtenido de Raspberry Pi: https://www.estudioelectronica.com/tienda/raspberry-pi/camara-raspberry-pi-infrarroja-5mp-ov5647/

Google. (2023). *Teachable Machine*. Obtenido de Primeros pasos: https://teachablemachine.withgoogle.com/

INTEF. (10 de 2018). *INTEF*. Obtenido de Programación y Robótica: https://formacion.intef.es/mod/book/view.php?id=2625&chapterid=2408&lang=es

Krishna, A. (6 de 4 de 2020). *IBM*. Obtenido de https://www.ibm.com/mx-es/topics/computer-vision

Marín, R. (12 de 02 de 2020). *INESEM*. Obtenido de revista digital Inesem: https://www.inesem.es/revistadigital/informatica-y-tics/opencv/

MCI electronics. (6 de 2017). Obtenido de https://raspberrypi.cl/que-es-raspberry/

Mercedes-Benz of Easton. (2024). *Mercedes-Benz ATTENTION ASSIST Explained*. Obtenido de How Does ATTENTION ASSIST Work?: https://www.mercedesbenzofeaston.com/mercedes-benz-attention-assist/

RTE INEN 101. (2017). Reglamento Técnico Ecuatoriano RTE INEN 101. *Artefactos Electrodomésticos para Cocción por Inducción*. Ecuador.

Support google. (2024). *Wi-Fi*. Obtenido de ¿Qué es un router?: https://support.google.com/googlenest/answer/6274087?hl=es-419#:~:text=Un%20router%20es%20un%20dispositivo,de%20%C3%A1rea%20local%20(LAN).

Uikey, P. (03 de abril de 2023). *DataTrained* . Obtenido de Import Numpy as np: https://datatrained.com/post/import-numpy-as-np/#:~:text=Importing%20Library%3A,name%20%E2%80%9Cnumpy%E2%80%9D%20each%20time.

UNASEV. (16 de 12 de 2020). *gub.uy*. Obtenido de https://www.gub.uy/unidad-nacional-seguridad-vial/comunicacion/publicaciones/somnolencia-conduccion

Unidad Nacional para la Gestión del Riesgo de Desastres. (05 de Diciembre de 2011). *Colombia potencia de la vida*. Obtenido de ALERTAS Y ALARMAS Mecanismos que salvan vidas: https://portal.gestiondelriesgo.gov.co/paginas/old_noticias/1240.aspx

VideoLAN Wiki. (02 de Noviembre de 2021). *VideoLAN Wiki*. Obtenido de libVLC: https://wiki.videolan.org/LibVLC

Volvo. (16 de 03 de 2023). *Tu vehículo*. Obtenido de Driver Alert Control: https://www.volvocars.com/pe/support/car/xc40/article/41b16f26897fe720c0a8015138d5d35c

XI. ANEXO

Enlace con programas y archivos utilizados:

https://estliveupsedu-my.sharepoint.com/:f:/g/personal/iventura_est_ups_edu_ec/En4ViicnC6xCl9XB2PQrAU4BeGaMEgLNvGmPmk3gN2Gu9w?e=JacezX

Código del programa principal de Python

```
# Importando Bibliotecas

import cv2 # OpenCV para trabajas con imagenes

import vlc  # Para reproducir archivos mp3

import numpy as np # para convertir la imagenes a numeros

# Para por der utilizar nuestro modelo entrenado

from keras.models import load_model


ruta = '/home/Tesis/Desktop/Tesis Somnolencia/'


# Inicializando Variables

video_capture = cv2.VideoCapture(0)

# modelo de identificacion de identificaión de ojo derecho

eyeLeft = cv2.CascadeClassifier(ruta + 'haarcascade_lefteye_2splits.xml')

# modelo de identificacion de identificaión de ojo izquierdo

eyeRight = cv2.CascadeClassifier(ruta + 'haarcascade_righteye_2splits.xml')

# Fotmato de objeto data usado por el modelo

data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Cargar modelo
```

```
model = load_model(ruta + 'keras_model.h5')

# Cargar archivo mp3

p = vlc.MediaPlayer("/home/Tesis/Desktop/Tesis/despierta.mp3")

left_x, left_y, left_w, left_h = 0, 0, 0, 0  # Generar Variables

right_x, right_y, right_w, right_h = 0, 0, 0, 0  # Generar Variables

contador = 0 # Guarda la cantidad de veces que el modelo detecte que este durmiendo

alarma = 0      # Variable encenido y apagado de alarma

window_1 = "Alarma1"      # Ventana de alarma 1

window_2 = "Alarma2"      # Ventana de alarma 2


# Ejecutando Video

while True:

    ret, frame = video_capture.read()  # Abre la camara

    # Extrae informacion de anchura y altura e la cámara

    height, width = frame.shape[:2]

    # convertir a blanco y negro, mejor interpretado para lenguaje math

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    # Mostrando Contador

    # (ventana, P, incial,          P. final       , color  ,grosor)

    # ( frame , (  0,  0), (width, int(height*0.1)), (0,0,0), -1)

    cv2.rectangle(frame, (0, 0), (width, int(height*0.1)), (0, 0, 0), -1)

    # (ventana, texto                ,          p. ubicacion           ,          fuente         , tamaño,
color , grosor)
```

```
    # (frame,  'Sueño: '+str(contador),(int(width*0.65), int(height*0.08)),
cv2.FONT_HERSHEY_SIMPLEX,    1  , (255,255,255), 1)
    cv2.putText(frame, 'Sueño: '+str(contador), (int(width*0.65),
                int(height*0.08)), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 1)

    # Identificando el Ojo Derecho

    # Saca la ubicación de cada ojo para encerrarlos en un cuadrado
    # a cada uno una vez identificados

    ojo_der = eyeRight.detectMultiScale(    # detección de objetos en imágenes
        gray,                           # fotograma convertido a blanco y negro
        # especifica cuánto se reduce el tamaño de la imagen
        scaleFactor=1.15,
        # especifica cuántos "vecinos" debería tener cada rectángulo
        minNeighbors=3,
        # Tamaño mínimo posible del objeto. Los objetos más pequeños que eso son
ignorados.
        minSize=(30, 30)
    )
    for (x, y, w, h) in ojo_der:
        # cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        right_x, right_y, right_w, right_h = x, y, w, h
        break
```

```
# Identificando el Ojo Izquierdo
ojo_izq = eyeLeft.detectMultiScale(
    gray,
    scaleFactor=1.15,
    minNeighbors=3,
    minSize=(30, 30)
)
for (x, y, w, h) in ojo_izq:
    # cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    left_x, left_y, left_w, left_h = x, y, w, h
    break

# Identificando Coordenadas x,y iniciales y finales para extraer la foto de los ojos

    # Ubicados el cuadrado de cada ojo, se procede a sacar las coordenadas
    # para encerralosen un unico cuadrado, por ello se revisa cual está a
    # mayor altura par encerralos a ambos del mayor a menor

if (left_x > right_x):
    start_x, end_x = right_x, (left_x+left_w)
else:
    start_x, end_x = left_x, (right_x+right_w)

if (left_y > right_y):
```

```
        start_y, end_y = right_y, (left_y+left_h)
    else:
        start_y, end_y = left_y, (right_y+right_h)

    # Algoritmo de deteccion de sueño

        # se realiza una resta para que sea mayor 120,
        # si llega ser menor debe ser muy posible que solo veo un ojo
    if ((end_x-start_x) > 120 and (end_y-start_y) < 200):
        start_x, start_y, end_x, end_y = start_x-30, start_y-50, end_x+30, end_y+50
        cv2.rectangle(frame, (start_x, start_y),
                      (end_x, end_y), (0, 255, 0), 2)       #
        # Se extrae el rectangulo que cubre los ojos para hacer la comparación con el
modelo
        img = frame[start_y:end_y, start_x:end_x]

        try:                                    # Por errores esporadicos se utiliza esta linea
            # Se cambia el tamaño para que se ajuste al tamaño de el modelo
            imagen = cv2.resize(img, (224, 224))
        except:                                 # Si falla, no detiene el programa solo espera al
siguiente fotogrma
            continue
        imagen_normalizada = (imagen.astype(np.float32) / 127.0) - 1
        data[0] = imagen_normalizada      # Se guarda la imagen en una variable
        # Se realiza la prediccion a la imagen recortada
```

```
        prediction = model.predict(data)

        # Se convierte la varialble a lista, y se hace la predicción, si llega a ser mayor a
0.95 es valida
        if (list(prediction)[0][1] >= 0.95):
            cv2.putText(frame, 'Durmiendo : '+str(round(list(prediction)[0][1], 3)), (10, int(
                height*0.08)), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 1)
            contador += 1  # Si es afirmativo, aumenta al contador (durmiendo)

        # Se convierte la varialble a lista, y se hace la predicción, si llega a ser mayor a
0.95 es valida
        if (list(prediction)[0][0] >= 0.95):
            cv2.putText(frame, 'Despierto', (10, int(height*0.08)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 1)
            contador -= 1  # Si es afirmativo, aumenta al contador (despierto)

        if contador >= 5:
            contador = 5  # si el contador llega a ser mayor a 5, se mantiene en ese valor

            if alarma == 0:                          # si la variable  de estado de alarma esta
apagado
                alarma = 1                           # la enciende
                p.play()                             # Suena la alarma
                # Llama, redimenciona y enseña imagen de  advertencia 1.
                img_1 = cv2.imread(ruta + 'Despierta.jpg')
```

```
            img_1S = cv2.resize(img_1, (250, 250))
            cv2.imshow(window_1, img_1S)
            cv2.moveWindow(window_1, 10, 50)
            # Llama, redimenciona y enseña imagen de  advertencia 2.
            img_2 = cv2.imread(ruta + 'Accidente.png')
            img_2S = cv2.resize(img_2, (250, 250))
            cv2.moveWindow(window_2, 310, 50)
            cv2.imshow(window_2, img_2S)

        elif (contador >= 0 and contador <= 1):              # si el contador a bajado

            # apaga el estado de la alarmas, apaga la alarma y cierra las advertencias
            if alarma == 1:
                p.stop()

                cv2.destroyWindow(window_1)
                cv2.destroyWindow(window_2)
                alarma = 0

        elif contador < 0:                  # si el contador llega a ser menor a 0, se mantiene en
ese valor
            contador = 0

    # Mostrar el video de la webcam
    # Muestra la camara, con la informacion
```

```
cv2.imshow('video', frame)


# Tecla de salida - acaba la transmisión
if cv2.waitKey(1) & 0xFF == ord('q'): # Al precionar "Q" se cierra el programa
    video_capture.release()
    cv2.destroyAllWindows()
    break
```