Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo fin de grado

DISEÑO Y DESARROLLO DE UN SISTEMA DE DETECCIÓN DE SOMNOLENCIA

Alumno: Rubén Domenech López

Tutor: Prof. D. Ildefonso Ruano Ruano
Depto.: Ingeniería de Telecomunicación

Julio, 2023

Universidad de Jaén
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DISEÑO Y DESARROLLO DE UN SISTEMA DE DETECCIÓN DE SOMNOLENCIA

Firma autor/a

DOMENECH LOPEZ RUBEN - 26524045P

Firmado digitalmente por DOMENECH LOPEZ RUBEN - 26524045P
Fecha: 2023.06.21 07:26:35 +02'00'

Firma tutor/a

RUANO RUANO ILDEFONSO - 02889873S

Firmado digitalmente por RUANO RUANO ILDEFONSO - 02889873S
Fecha: 2023.06.21 10:21:04 +02'00'

ÍNDICE GENERAL

Resumen 10
Abstract 11
Aspectos generales 12
1. Introducción 12
2. Objetivos 15
3. Estudio previo 16
3.1. Somnolencia. Causas y efectos. 16
3.2. Diferentes técnicas para detectar somnolencia. 17
3.2.1. Detección mediante ROI y distancias euclídeas 18
3.2.2. Detección de patrones por aprendizaje automático 19
3.2.3. Análisis de la frecuencia de parpadeo 19
3.2.4. Justificación elección aprendizaje automático 20
3.3. Evolución de la tecnología para prevenir riesgos en la conducción 21
3.3.1. Sistemas de Ayuda Avanzada a la Conducción (ADAS) 21
3.3.1.1. Continental Driver Focus 22
3.3.1.2. Sistema de detección de fatiga de Bosch 23
4. Tecnologías 25
4.1. Recursos Software 25
4.2. Recursos Hardware 26
5. Motivación 27
Sistema de detección de somnolencia. Visión global 29
6. Lenguaje de programación 29
6.1. Python 29
7. Entorno de desarrollo 30
7.1. Visual Studio Code 30
8. Tecnologías empleadas 30
8.1. Internet de las Cosas (IoT) 30
8.2. Inteligencia artificial (IA) 31
8.3. Deep Learning 32
8.4. Machine Learning 34
Sistema de detección de somnolencia. Software 36
9. Bloques funcionales 36
9.1. Bloque 1. Detección de somnolencia 36

9.1.1. Bibliotecas 37
9.1.1.1. Cv2 (OpenCV) 37
9.1.1.2. Keras 38
9.1.1.3. Numpy 38
9.1.1.4. Math 39
9.1.1.5. Time 39
9.1.1.6. Pygame 40
9.1.2. Modelo de entrenamiento 41
9.1.2.1. Dataset 41
9.1.2.2. Red neuronal convolucional 42
9.1.2.3. Compilación del modelo y resultados 43
9.1.3. Funcionamiento del bloque 44
9.2. Bloque 2. Asistente virtual 47
9.2.1. Bibliotecas 48
9.2.1.1. Random 48
9.2.1.2. JSON 48
9.2.1.3. Io 49
9.2.1.4. Os 50
9.2.1.5. Dill 50
9.2.1.6. NLTK (NLP, en español) 50
9.2.1.7. Pydub y ffmpeg 52
9.2.1.8. Speech Recognition 53
9.2.1.9. Pyttsx3 53
9.2.1.10. Whisper 54
9.2.2. Modelo de entrenamiento 56
9.2.2.1. Preprocesamiento de datos 56
9.2.2.2. Preparación de los datos 57
9.2.2.3. Red neuronal artificial (RNA) 57
9.2.2.4. Entrenamiento del modelo 58
9.2.3. Integración de voz en el asistente virtual 61
9.2.4. Funcionamiento 63
9.3. Bloque 3. Envío de mensajes a la nube 65
9.3.1. Plataforma de servicio. Firebase 65
9.3.2. Bibliotecas 66
9.3.2.1. Requests 66

9.3.2.2. API IpStack....67
9.3.2.3. Datetime....68
9.3.2.4. Firebase_admin....68
9.3.3. Protocolo de envío de mensajes....68
9.3.4. Funcionamiento....69
Sistema de detección de somnolencia. Prototipo....71
10. Prototipo físico....71
10.1. Dispositivos y accesorios....72
10.1.1. Raspberry PI....72
10.1.2. Cámara Pi....72
10.1.3. Tarjeta SD....73
10.1.4. Adaptador de corriente....74
10.1.5. Micrófono USB....74
10.1.6. Altavoz....75
10.1.7. Punto de acceso....76
10.2. Inserción software en Raspberry....76
11. Resultados finales....78
12. Estudio económico....84
12.1. Materiales....84
12.2. Mano de obra....85
12.3. Resumen....86
13. Líneas futuras....87
14. Conclusiones....88
15. Anexos....89
Anexo I. Configuración inicial de Raspberry....89
Anexo II. Acceso remoto a Raspberry....94
Anexo III. Aspectos clave para correcto funcionamiento....101
16. Bibliografía....104

ÍNDICE FIGURAS

Figura 1. Diagrama de flujo general del proyecto 12
Figura 2. Causas y riesgos de la somnolencia. 16
Figura 3. Sistemas ADA. 22
Figura 4. Continental Driver Focus 23
Figura 5. Sistema de detección de fatiga de Bosch 24
Figura 6. Ejemplo del sistema Bosch 25
Figura 7. Funcionamiento ecosistema IoT 31
Figura 8. Ejemplo de Deep Learning. 33
Figura 9. Diagrama general de bloques. 36
Figura 10. Instalación primaria OpenCV 37
Figura 11. Instalación completa OpenCV 38
Figura 12. Instalación keras 38
Figura 13. Instalación Numpy 39
Figura 14. Instalación pygame 40
Figura 15. Clasificación imágenes dataset 41
Figura 16. Resultados del entrenamiento 44
Figura 17. Diagrama de bloques para la detección de somnolencia 45
Figura 18. Archivo JSON 49
Figura 19. Instalar Dill 50
Figura 20. Instalar NLTK 1 51
Figura 21. Recursos adicionales NLTK 1 51
Figura 22. Instalar pydub 1 52
Figura 23. Instalar SpeechRecognition. 53
Figura 24. Instalar pyttsx3 1 53
Figura 25. Arquitectura Whisper 55
Figura 26. Instalación whisper simple 56
Figura 27. Instalar Whisper AI OpenAI 56
Figura 28. Primeras épocas entrenamiento 59
Figura 29. Ecuador épocas entrenamiento 60
Figura 30. Época final entrenamiento 1 60
Figura 31. Diagrama de bloques AV. 1 63
Figura 32. Realtime Database Firebase 66
Figura 33. Instalación Requests 67

Figura 34. Obtener geolocalización 67
Figura 35. Instalar firebase_admin 68
Figura 36. Protocolo HTTP 69
Figura 37. Diagrama de bloques del protocolo de alto riesgo 70
Figura 38. Prototipo físico 71
Figura 39. Partes de la Raspberry Pi 4B 72
Figura 40. Cámara nativa de Raspberry 73
Figura 41. Tarjeta SD para Raspberry 74
Figura 42. Adaptador de corriente para Raspberry 74
Figura 43. Micrófono USB 75
Figura 44. Altavoz Bluetooth 75
Figura 45. Comando ejecución del proyecto en Raspberry 77
Figura 46. Compilación software en equipo. 78
Figura 47. Prototipo físico en funcionamiento 79
Figura 48. Detección ojos abiertos 80
Figura 49. Detección ojos cerrados 80
Figura 50. Detección de somnolencia 81
Figura 51. Entrada de datos y respuesta del asistente virtual. 81
Figura 52. Momento exacto de la detección de somnolencia 82
Figura 53. Protocolo de alto riesgo 82
Figura 54. Exposición de los datos enviados en Firebase 83
Figura 55. Raspberry PI Imager 89
Figura 56. Entorno Imager 90
Figura 57. Selección OS para Raspberry 90
Figura 58. OS y Almacenamiento 90
Figura 59. Opciones de configuración 1 91
Figura 60. Opciones de configuración 2 92
Figura 61. Opciones de configuración 3 92
Figura 62. Opciones de configuración 4 93
Figura 63. Insertar SD en Raspberry 93
Figura 64. Conexión Raspberry al router 94
Figura 65. Escaneo de red 95
Figura 66. Especificaciones adaptador inalámbrico 95
Figura 67. Acceso a raspberry por SSH 96
Figura 68. Menú de configuración de Raspberry 97

Figura 69. Menú de configuración de Raspberry 2 97
Figura 70. Habilitar SSH 98
Figura 71. Establecimiento de conexión VNC. 98
Figura 72. Establecer conexión 1 99
Figura 73. Establecer conexión 2 99
Figura 74. Credenciales 100
Figura 75. Escritorio remoto de Raspberry 100
Figura 76. API Key IPStack 101

ÍNDICE TABLAS

Tabla 1. Organización del Dataset 42
Tabla 2. Requerimientos en Whisper 55
Tabla 3. Comparación modelos Whisper.....62
Tabla 4. Presupuesto de materiales.....84
Tabla 5. Presupuesto de mano de obra 85
Tabla 6. Presupuesto final.....86

Resumen

Este trabajo, como indica el título del mismo, se basa en el diseño y desarrollo de un sistema tecnológico para la detección de somnolencia en tiempo real. La aplicación principal de este tipo de sistemas es la mejora de la seguridad de los medios de transporte en los que se encuentran conductores dirigiendo los vehículos, aunque se podría combinar con otro tipo de sistemas que aumentaran la seguridad en caso de detección de somnolencia positiva aplicando técnicas de conducción autónoma.

Se ha desarrollado un sistema basado en Python, un lenguaje de programación muy versátil que ofrece un gran número de librerías útiles y una comunidad de soporte muy amplia y activa en torno a él. Se han usado técnicas de inteligencia artificial (aprendizaje automático) para realizar la detección de imágenes de forma automática basada en modelos y el reconocimiento de patrones a través del lenguaje. Cuando se produce una detección de somnolencia grave, algunos datos como la matrícula del vehículo, la fecha y hora de cuando se ha producido el suceso y las coordenadas geográficas serán enviados a una base de datos en tiempo real hospedada en la nube con objeto de que los servicios de emergencias puedan hacer uso de esa información para ayudar, esto ocurrirá siempre y cuando se disponga de la conexión necesaria, por lo que además se puede catalogar como un sistema IoT (Internet de las cosas).

El resultado final ha sido doble, por un lado, se ha obtenido un sistema de pruebas que puede ser ejecutado en ordenadores que dispongan de cámara, micrófono y altavoces y, por otro, se ha obtenido un prototipo de bajo coste basado en un microprocesador Raspberry Pi 4B con su cámara nativa, un micro USB y un altavoz. En ambos casos se puede realizar un reconocimiento facial en tiempo real que permite monitorizar el parpadeo de los ojos para, aplicando técnicas de aprendizaje automático, detectar cuando se puede producir un estado de somnolencia previo a la falta de conciencia y comunicar resultados a una base de datos en tiempo real alojada en Internet.

Abstract

This work, as its title indicates, is based on the design and development of a technological system for the detection of drowsiness in real time. The main application of this type of system is to improve the safety of means of transport in which drivers are in charge of the vehicles, although it could be combined with other types of systems that increase safety in the event of positive drowsiness detection by applying autonomous driving techniques.

The system has been developed based on Python, a very versatile programming language that offers a large number of useful libraries and a very large and active support community around it. Artificial intelligence (machine learning) techniques have been used to perform automatic model-based image detection and pattern recognition through language. When a detection of severe drowsiness occurs, data such as vehicle registration number, date and time of the event and geographical coordinates will be sent to a real-time database hosted in the cloud so that emergency services can make use of this information to help, provided the necessary connection is available, so it can also be categorised as an IoT (Internet of Things) system.

The final result has been double. On the one hand, a test system has been obtained that can be run on computers with a camera, microphone and speakers and on the other hand, a low-cost prototype has been obtained based on a Raspberry Pi 4B microprocessor with its native camera, a micro USB and a speaker. In both cases, real-time facial recognition can be performed to monitor the blinking of the eyes and, by applying machine learning techniques, detect when a state of drowsiness may occur prior to lack of consciousness and communicate results to a real-time database hosted on the Internet.

Aspectos generales

1. Introducción

En la actualidad, la somnolencia es una de los principales problemas y que en el transcurso del año deja una alta tasa de accidentalidad en las carreteras. Por este motivo, surge la necesidad de crear sistemas de prevención, que ayudarán a mejorar la seguridad vial. Por ende, entra en juego la tecnología.

En los últimos años se ha producido un auge de sistemas que implementan inteligencia artificial que tienen por objeto realizar tareas similares a las del ser humano [1]. Esto no sería posible sin hacer uso de Machine Learning, esta tecnología ha hecho que las máquinas evolucionen en la forma de aprender, y sean capaces de realizar tareas cada vez más complejas y sofisticadas con una alta eficacia, actualmente los algoritmos que se utilizan se dividen en tres modelos de aprendizaje: supervisado, no supervisado y aprendizaje por refuerzo. [2]

El presente proyecto trata sobre un sistema de detección de somnolencia, basándose en el monitoreo del parpadeo en tiempo real. Para comprender el funcionamiento lógico del proyecto, servirá de ayuda la siguiente figura (figura 1):

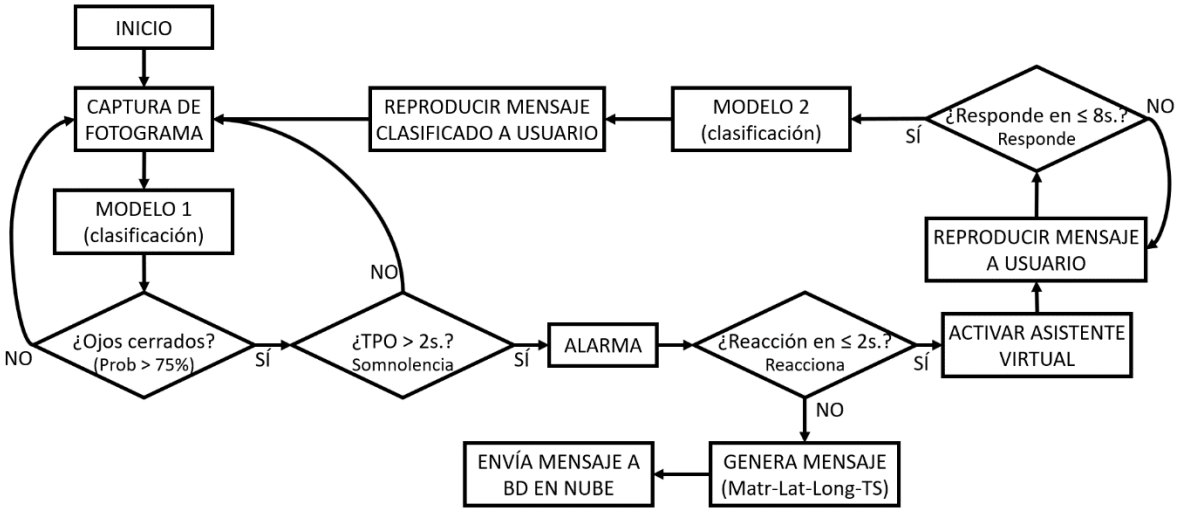

Figura 1. Diagrama de flujo general del proyecto

El funcionamiento del proyecto comienza haciendo uso de técnicas de visión por computación, gracias a las cuales se puede realizar una monitorización de objetos e identificación de los mismos, de modo que resulta útil para la detección de rostros faciales

frente a una cámara. Además, es necesario el procesamiento de las imágenes capturadas, con objeto de establecer los fotogramas capturados por la cámara a una relación de datos acorde a los modelos de entrenamiento de los que hace uso el modelo.

Una vez que se realiza el procesamiento de las imágenes, se detecta si el usuario está experimentando somnolencia o no. En caso de experimentar somnolencia, el programa procede a activar una alarma de sonoridad aguda, con el objetivo de que el usuario reaccione ante dicho estimulo. En caso de ser efectiva (reacción antes transcurrir 2/3 segundos desde el inicio de la alarma), asistirá al usuario un asistente virtual, donde mantendrán una pequeña conversación con la finalidad de persuadir al sujeto para realizar una parada en un lugar seguro y tomar así las medidas que se estimen oportunas. Por el contrario, en caso de no surgir efecto la alarma, el sistema procede a activar la conducción inteligente del vehículo para ejecutar una maniobra de parada, de la forma más segura posible. Además, el sistema procederá a enviar un mensaje a una base de datos en tiempo real, que se encuentra hospedada en la nube y gestionada por los servicios de emergencia, los datos enviados son de principal interés ya que es una estructura formada por la matricula correspondiente al vehículo, la geolocalización con sus coordenadas de longitud y latitud correspondientes y una marca temporal (TimeStamp) con la fecha y la hora en la que se ha producido el suceso.

Para lograr la finalidad del proyecto, éste se apoya en modelos de aprendizaje supervisado con métodos de clasificación. Estos modelos se entrenan utilizando conjuntos de datos etiquetados (Dataset). El primer modelo se centra en la clasificación del estado de los ojos, diferenciando entre ojos abiertos y cerrados, para el entrenamiento del modelo se recurre a un dataset con más de 80 mil fotos con multitud de escenarios en cada fotograma (ver apartado 9.1.2.1), este modelo se compone de una red neuronal convolucional, técnica de Deep Learning, consta de varias capas algunas para detectar características relevantes de los fotogramas capturados y otras para realizar la clasificación en base a los datos obtenidos en las capas anteriores.

El segundo modelo del que hace uso el sistema, se basa en procesamiento del lenguaje natural (NLTK (NLP, en español)), este modelo tiene como objetivo principal comprender y responder a las respuestas de los usuarios en base al estado anímico actual después de haberse producido la detección de un micro sueño. Este modelo comprende de diferentes etapas. Primero, carga y procesa un archivo JSON que contiene las intenciones y patrones asociados con las consultas de los usuarios. La segunda etapa trata

de construir un conjunto de datos de entrenamiento utilizando la técnica de bolsa de palabras (bag of words), donde se generan vectores que indican la presencia o ausencia de cada palabra clave en una consulta. Por último, se lleva a cabo un entrenamiento haciendo uso de una Red neuronal artificial (RNA).

Además, el proyecto no solo recurre a tecnologías que están siendo punteras en campos como la ingeniería o ciencias de datos, sino que también se puede englobar dentro del ámbito del Internet de las Cosas (IoT) [3], ya que el sistema interconecta el vehículo con una base de datos hospedada en la nube, y recopila información en tiempo real como la geolocalización, fecha y hora del suceso, que, junto con otros datos, los envía a la nube para su posterior tratamiento por terceros.

Por último, a lo largo de la memoria se puede observar que el proyecto discurre en dos vertientes, puesto que se ha logrado crear un software genérico capaz de funcionar sobre cualquier ordenador convencional que disponga de cámara y altavoz y por otro lado, se ha logrado la consecución de un prototipo físico, capaz de detectar somnolencia en un entorno real, haciendo uso de Raspberry PI.

2. Objetivos

El objetivo principal de este proyecto se puede desglosar en dos partes, por un lado, el objetivo reside en la creación de un sistema de detección de somnolencia en tiempo real a modo de pruebas, que se pueda ejecutar en cualquier ordenador que disponga de una serie mínima de requisitos como, por ejemplo: una cámara, unos altavoces y a poder ser un procesador y memoria RAM de gama media, con objeto de tener una mayor fluidez en la actividad del programa. Por otro lado, la creación de un modelo físico experimental que consta de una Raspberry PI 4B, un micrófono USB y cámara nativa de Raspberry, siendo capaz de detectar somnolencia y mejorar así la seguridad vial en un entorno real.

Como segundo objetivo y surgiendo de la necesidad de crear un valor de interés añadido al proyecto, realizar la creación de un asistente de voz virtual inteligente siendo el papel de éste, fundamental en la toma de decisiones del usuario, ya que pretende influir positivamente en las decisiones para salvaguardar la integridad del conductor. Este objetivo tiene aplicación en las dos vertientes del objetivo principal

Como tercer objetivo, se propone la creación de un sistema que sea capaz de enviar información de interés a la nube, concretamente, a una base de datos en tiempo real que estará gestionada por los servicios de emergencia. Los datos a enviar serán la geolocalización de donde se ha producido el suceso, la fecha junto con la hora y la matrícula del vehículo implicado en el suceso. Al igual que el objetivo anterior, éste también tiene aplicación en las dos vertientes del objetivo principal.

3. Estudio previo

En este apartado se realizará un análisis sobre las diferentes formas de detectar un micro sueño en tiempo real, además se explicará que es un micro sueño y condiciones comunes que llevan a dicha situación, por último, se realizará un inciso en el avance de la tecnología en el sector automovilístico destacando los proyectos que se están llevando a cabo con respecto al presente proyecto por empresas de gran relevancia en el sector de la automoción.

3.1. Somnolencia. Causas y efectos.

La somnolencia en la conducción es un estado intermedio entre la vigilia y el sueño que puede ser causado por diferentes factores, como la falta de descanso, la privación del sueño, el síndrome de apnea del sueño, el insomnio, la toma de medicamentos, la ingesta de alcohol y drogas, entre otros. La somnolencia puede afectar gravemente la capacidad de conducción de una persona y aumentar el riesgo de accidentes de tráfico. Por esta razón, es importante conocer las causas y efectos de la somnolencia en la conducción, así como los métodos para detectarla y prevenirla [4]. En la siguiente figura (figura 2), se pueden observar las causas y sus correspondientes riesgos.

Figura 2. Causas y riesgos de la somnolencia.

Fuente: https://acortar.link/bTzvd7

Con respecto a los efectos que produce la somnolencia en un conductor, la Dirección General de Tráfico (DGT) menciona los siguientes:

1. **Incremento del tiempo de reacción.** Este problema se puede traducir fácilmente en la disminución de la capacidad de reacción.

2. **Menor concentración y más distracciones.** Esta condición se suele dar sobre todo en tramos donde la carretera es muy monótona, es decir, carreteras con una recta muy larga, poco tráfico, etcétera.
3. **Toma de decisiones más lenta y más errores.** Cuando se experimenta somnolencia, la capacidad para procesar la información que se recibe del entorno y responder adecuadamente disminuye. También es más probable tomar decisiones equivocadas, especialmente en situaciones complejas donde se requiere una respuesta rápida.
4. **Alteraciones motoras.** La somnolencia provoca que los músculos se destensen y por tanto los brazos estén más flojos, dando lugar a errores, además entra en juego la fatiga que hace empezar a sentir cosquilleos en los brazos, los pies, etcétera.
5. **Movimientos más automatizados.** Debido al cansancio el conductor baja la guardia y se confía con las condiciones del entorno, desactivando así su estado de alerta y haciendo todo de manera más automática y errática.
6. **Alteración de las funciones sensoriales.** Llegados a este punto se necesitan unos estímulos más fuertes para reaccionar, la vista empieza a desenfocarse y se produce la visión borrosa y la fatiga ocular. Por lo tanto, resulta más fácil que un conductor en un momento determinado pueda dar un volantazo por deslumbramiento y producirse así un accidente.
7. **Aparición de micro sueños.** Probablemente el efecto más negativo de la somnolencia, como se comenta en apartados anteriores, es un periodo de tiempo en el cual el conductor pierde completamente la conciencia, se considera micro sueño a partir de los dos o tres segundos y por ello se producen accidentes de extrema gravedad.
8. **Alteraciones de la percepción.** Se produce cuando el sujeto lleva demasiadas horas sin descansar, es decir, una privación del sueño excesiva. En esta fase se pueden llegar a experimentar alucinaciones e incluso ilusiones visuales.
9. **Cambios en el comportamiento.** La somnolencia al volante puede provocar cambios en tu estado de ánimo y comportamiento. Es posible que te sientas más tenso, nervioso o agresivo de lo normal.

3.2. Diferentes técnicas para detectar somnolencia.

La somnolencia se puede detectar mediante diversos métodos, algunos son más sofisticados que otros y por ende, más seguros. A continuación, se presentan algunos de los métodos más comunes para la realización de dicha tarea. Además, se realizará una justificación sobre que método se ha llevado a cabo en el desarrollo del proyecto.

Todos ellos harán uso de algoritmos de visión por computadora, tomando los ficheros en cascada y OpenCV, para la detección facial y extraer así la información necesaria para poder procesarla y determinar el estado.

3.2.1. Detección mediante ROI y distancias euclídeas

Esta técnica consiste en la monitorización del parpadeo, el mecanismo es sencillo, por definición parpadeo consiste en cerrar y abrir los ojos, de modo que, si se cierran y se abren en un tiempo posterior a dos o tres segundos, se considera que la persona ha experimentado un micro sueño.

Este método utilizará técnicas de visión por computación asi como un fichero “shape_landmark_face.xml”, el cual cuenta con una serie de puntos que marcan los contornos de cada parte de nuestro rostro, que posteriormente serán útiles para declarar las regiones de interés y poder calcular una serie de distancias que determinarán el estado de nuestros ojos.

Lo primero que se realiza es la captura del rostro haciendo uso de la biblioteca Cv2 (OpenCV), una vez capturados los frames, se detectarán los puntos del rostro encontrado en dichos fotogramas. Después los puntos se convertirán a pixeles, esto se realiza multiplicando los puntos obtenidos por el ancho de la ventana.

Posteriormente, se detectan los puntos tanto del parpado superior como del parpado inferior, del ojo izquierdo y derecho. A continuación, se procede a hallar la longitud que hay entre ambos puntos en cada uno de los ojos.

Por otro lado, se deberá determinar un umbral, dicho umbral marcará el estado de los ojos, de modo que, si la longitud hallada anteriormente es menor que el umbral, quiere decir que los ojos se encuentran cerrados.

Para determinar que se está produciendo somnolencia es necesario recurrir a un contador temporal, por ejemplo, se puede hacer uso de la biblioteca Time. Se puede establecer el inicio de la marca temporal cuándo se cierran los ojos y otra marca temporal final cuando se abren, conociendo asi, el tiempo que los ojos permanecen cerrados.

En caso de que esa marca temporal supere el umbral establecido de micro sueño, se puede determinar que la persona está experimentando somnolencia.

3.2.2. Detección de patrones por aprendizaje automático

Durante la implementación de esta técnica se hace uso de redes neuronales convolucionales, los modelos de entrenamiento con los que cuenta esta técnica requieren de **Dataset,** que son datos estructurados y procesados para el entrenamiento, la mayor parte de imágenes en este caso de las que se compone el dataset, se utilizarán para entrenamiento y otro pequeño porcentaje para validación.

Una vez que el modelo de entrenamiento es válido, se hace uso de técnicas de visión por computadora, asi como de ficheros Haar Cascade. Cada fotograma será convertido a escala de grises, ya que no es información relevante el color y se tardarían más en procesar, y se llevará a cabo la detección de caras en la imagen utilizando el clasificador Haar cascade de la cara frontal.

Para cada cara detectada en los frames capturados, se detectarán los ojos utilizando los clasificadores Haar cascades (existe uno para cada ojo). A continuación, se realizará una segmentación, es decir, se procesará cada ojo por separado para agilizar el procesamiento de los datos. Se ajustará su tamaño, se normalizará y se realizará una predicción utilizando el modelo de CNN.

En la siguiente fase se establecerá el estado de los ojos, dependiendo siempre de la probabilidad obtenida anteriormente. Si ambos ojos se detectaran como cerrados, se establecería un contador de tiempo y se comprobaría si ha alcanzado un umbral determinado para considerarlo un estado de somnolencia.

Recordar que un micro sueño es la perdida de la conciencia durante unos segundos, concretamente, se puede considerar micro sueño a partir de 2-3 segundos **(ver apartado Somnolencia. Causas y efectos.).** Luego, ese umbral debe de ser los segundos como de referencia.

3.2.3. Análisis de la frecuencia de parpadeo

A continuación, se mostrará una técnica muy similar a la comentada anteriormente (ver apartado 3.2.1), pero la diferencia reside en que la otra técnica cuenta la distancia de duración del parpadeo, mientras que ésta se basa en la frecuencia recurrente del parpadeo en un intervalo de tiempo pequeño.

Este modelo se basa en la detección de parpadeo mediante la monitorización de un área de interés, dicho área de interés será zona ocular del rostro. Dicha detección se hará mediante técnicas de visión por computadora, haciendo uso de la librería Cv2 (OpenCV), además hará uso de ficheros Haar Cascade, para facilitar la obtención de la región de interés.

Para facilitar el hallazgo de la ROI (Región de Interés), se recurrirá a la segmentación de los párpados en determinados intervalos de tiempo. Para cada intervalo de tiempo, se calcularán características relevantes de los párpados, como el área, la altura o la relación de apertura de los párpados. Estas características representarán la forma y la posición de los párpados.

A continuación, se deberá definir el umbral, este umbral será el que marque la diferencia de estados, de modo que siempre se realizará comparación con éste para determinar si los ojos se encuentran abiertos o cerrados.

Luego, se procederá al cálculo de las distancias euclídeas de apertura de los párpados. La distancia euclidiana medirá la similitud entre las características actuales y los valores de referencia. Por tanto, si la distancia euclidiana calculada es menor que el umbral establecido, se considera que ha ocurrido un parpadeo.

Si el parpadeo es demasiado recurrente en un intervalo temporal pequeño, se puede determinar que el usuario está experimentando somnolencia.

3.2.4. Justificación elección aprendizaje automático

Una vez realizado el análisis sobre algunos de los modelos que se pueden implementar para la detección de somnolencia basándose en técnicas de visión por computación. El método que parece más certero y el que se ha implementado en este proyecto es el realizado con aprendizaje automático (ver apartado 3.2.2).

Se ha implementado este método debido a la posibilidad de dotar al proyecto de inteligencia, mediante ésta técnica se puede realizar un modelo de aprendizaje supervisado, el cual se puede entrenar en mayor o menor medida con una cierta cantidad de datos, en este caso, el modelo se ha entrenado haciendo uso de un Dataset que consta de 80 mil imágenes, por tanto, el modelo será capaz de detectar el estado de los ojos con

mayor facilidad que los demás métodos, por otra parte también aportará más eficacia, debido a que el Dataset cuenta con imágenes de todo tipo de luminosidad, reflejo, etcétera.

El uso de la inteligencia artificial y el Machine Learning en la detección de somnolencia ofrece numerosas ventajas en comparación con los métodos tradicionales. Además de la capacidad de aprendizaje y adaptación, estos sistemas pueden procesar grandes cantidades de datos en tiempo real, lo que permite una detección instantánea de la somnolencia y una respuesta rápida para evitar situaciones de riesgo.

3.3. Evolución de la tecnología para prevenir riesgos en la conducción

La tecnología dentro de la industria de la automoción ha evolucionado significativamente con el paso de los años. Uno de los avances más importantes es el desarrollo de los sistemas avanzados de asistencia al conductor (ADAS, por sus siglas en inglés), los cuales han ido incorporándose en muchos vehículos modernos, ayudando así a mejorar la seguridad y la satisfacción en la conducción del usuario.

3.3.1. Sistemas de Ayuda Avanzada a la Conducción (ADAS)

Los sistemas ADAS se utilizan para mejorar la seguridad del conductor y de los pasajeros en el vehículo. Estos sistemas están diseñados para utilizar sensores, cámaras y otros dispositivos para monitorear el entorno del vehículo y ayudar al conductor con una variedad de tareas, desde mantener el carril hasta alertar sobre la proximidad de otros vehículos o peatones.

En particular, los sistemas ADAS pueden ayudar a prevenir la somnolencia y la fatiga del conductor mediante la monitorización del comportamiento del conductor y de los pasajeros, alertando cuando se detectan signos de fatiga. Algunos sistemas ADAS también pueden ayudar a prevenir otros accidentes, como los que ocurren por cambio involuntario de carril o por falta de atención al detectar objetos en puntos ciegos.

Algunos de los sistemas ADAS más comunes incluyen el control de crucero adaptativo (ACC), el frenado de emergencia autónomo (AEB), la asistencia de mantenimiento de carril (LKA), la alerta de punto ciego (BSM) y la detección de peatones o ciclistas. Estos sistemas pueden ayudar a prevenir accidentes y hacer que la conducción sea más cómoda y segura.

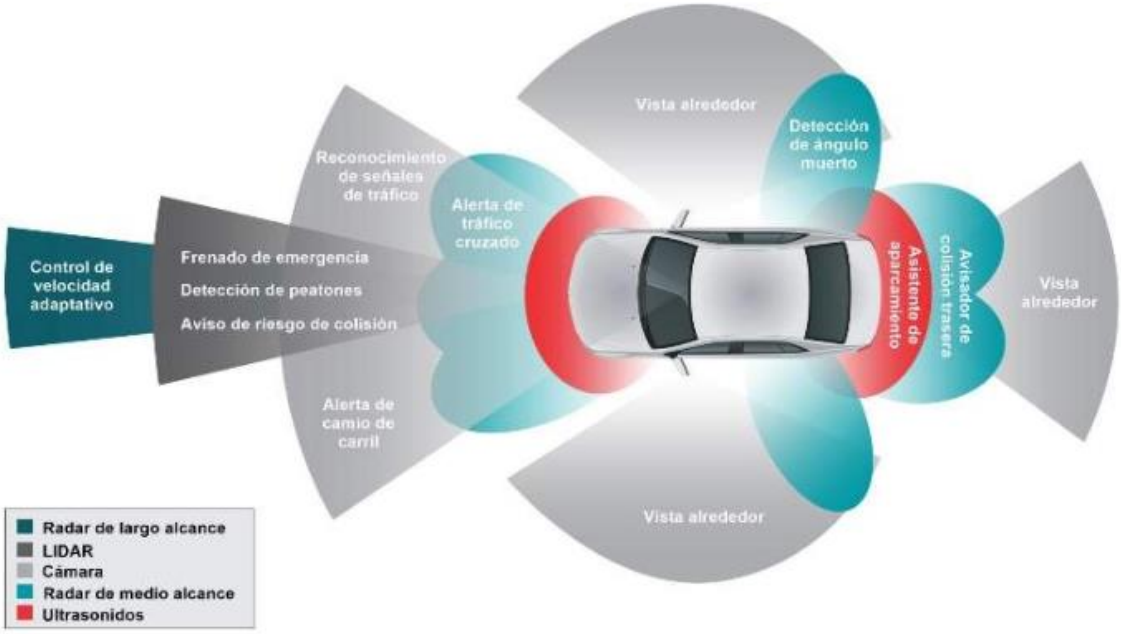

Figura 3. Sistemas ADA.

Fuente: https://www.eleconomista.es/ecomotor/motor/noticias/9391326/09/18/.

Aunque los sistemas de detección de somnolencia todavía no se han asentado al cien por cien, son multitud de empresas las que están investigando y realizando su desarrollo, entre ellas tenemos a Ford, Continental, Bosch, PSA Peugeot y Citroën [5]. Cada uno de los sistemas tiene un funcionamiento distinto, aunque objetivos similares. En los próximos puntos se describirán dichos proyectos con más de detalle.

3.3.1.1. Continental Driver Focus

El sistema ADAS consta de una cámara de infrarrojos que monitorea en todo momento al conductor, tratando de captar distracciones o signos de fatiga. Si el conductor mira a otro lado que no sea al frente, es decir, a la carretera, y lo hace durante un tiempo excesivo, se activa un mecanismo que consta de una pequeña luz que tratará de llevar los ojos del conductor hacia el campo de visión correcto. Y si se dan signos de somnolencia, es decir, se cierran los parpados más tiempo de lo normal o se realizan movimientos constantes de cabeza como si se estuvieran dando cabezadas, se activa el mismo mecanismo.[5]

Además, incorpora alertas sonoras para avisar al conductor y recomendar que tome un descanso. Si el conductor no responde a estas alertas, el sistema puede tomar medidas

adicionales, como disminuir la velocidad del vehículo o cambiar la posición del asiento del conductor para mejorar la postura.

Figura 4. Continental Driver Focus

Fuente: [estudio-somnolencia-al-volante.pdf (fundacioncea.es)]

3.3.1.2. Sistema de detección de fatiga de Bosch

Dicho sistema es capaz de detectar somnolencia analizando el comportamiento del conductor al volante, y avisarlo en caso de riesgo. Dispone de una cámara integrada en el volante que se encarga de monitorear el movimiento de los párpados, estudiando así la pesadez de estos, de modo que controla tanto la abertura como el cierre. Por otro lado, la cámara también vigilará si el conductor se distrae o mira hacia otros pasajeros demasiado tiempo, esto se debe a que la cámara también controla el comportamiento de los pasajeros. [6]

Obviamente, no se realiza desde la misma cámara, ya que la primera va integrada en el volante y apunta directamente al conductor, pero existirá otra cámara colocada en la parte superior del coche, más concretamente donde se encuentra ubicado el espejo retrovisor interior, y controlará el comportamiento de los pasajeros.[6]

Este sistema se ayudará de inteligencia artificial junto con los algoritmos de procesamiento de imágenes, para hacer un estudio más efectivo de la situación y en caso de que dicho conductor se encuentre en riesgo, a través del cuadro de mandos y de una

señal acústica, el sistema le recomendará que haga una parada y descansar unos minutos, de inmediato.[6]

Bosch también ha desarrollado este sistema pensando en la conducción autónoma de nivel 3. Es decir, en caso de que el conductor se encuentre muy fatigado, podrá activar mediante voz el sistema de conducción inteligente, asegurándose así que se llega al área de descanso más cercana, donde podrá descansar y recuperarse para seguir con su viaje. De modo que, este sistema es totalmente apto para evaluar si el conductor se encuentra en condiciones de continuar o no.[6]

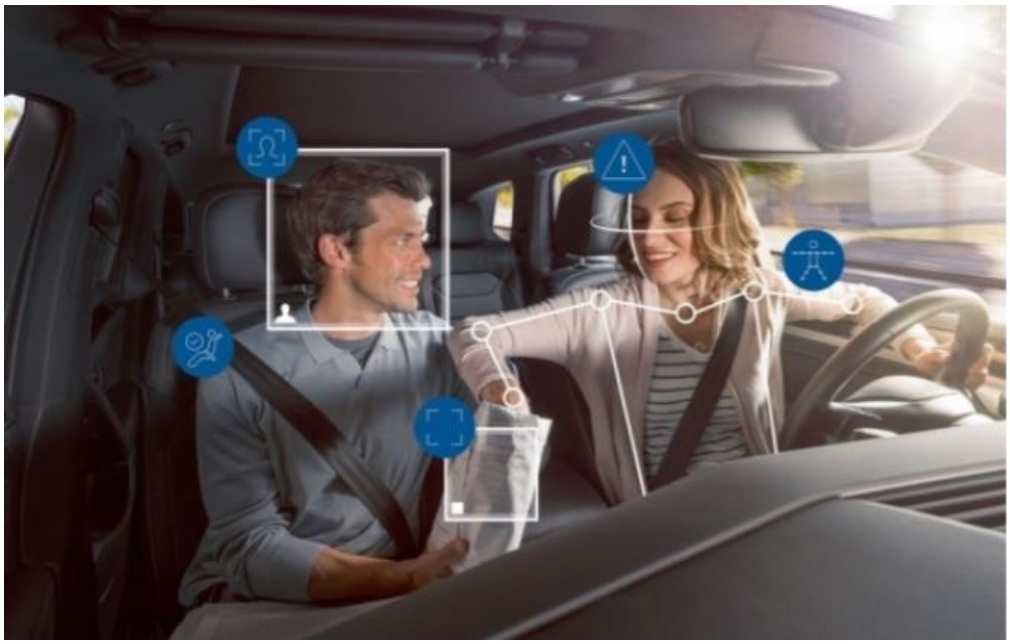

Figura 5. Sistema de detección de fatiga de Bosch

Fuente: https://www.motor.es/noticias/bosch-detector-fatiga-inteligencia-artificial-ces-2020-201963373.html.

La novedad que incorpora este sistema con respecto a sus competidores es que detector de fatiga es capaz de analizar el estilo de conducción a raíz de los datos que capta en el momento y de información adicional sobre la forma de conducción, y además el volante dispone de un sensor de ángulo de giro que viene integrado como parte del sistema ESP. El sistema identifica lo que se conoce como “puntos muertos” es decir, instantes en los que el conductor no está conduciendo (durante un periodo corto de tiempo) que finalizan con un movimiento brusco del volante, esto viene a decir que ciertas capacidades que se requieren en la conducción como por ejemplo la concentración, están decayendo. [5]

El sistema es tan efectivo porque tiene en cuenta diversas variables, es capaz de combinar la frecuencia y la fuerza de las reacciones a los denominados “puntos muertos”

y cruzar dichos datos con la velocidad del vehículo, la hora del día y el uso de los mandos del vehículo. De esta forma se hace mucho más confiable el cálculo predictivo de la detección de cansancio.[5]

Figura 6. Ejemplo del sistema Bosch

Fuente: https://www.autofacil.es/tecnica/sistema-deteccion-fatiga-funcionamiento/176204.html

4. Tecnologías

Para la consecución de este proyecto se han utilizado una serie de tecnologías y recursos hardware que se van a enumerar a continuación. Además, se nombrará el apartado correspondiente de esta memoria donde se van a comentar algunas pinceladas necesarias para entender su funcionamiento y justificar su uso en este trabajo.

4.1. Recursos Software

Para la realización del software ha sido necesaria la intervención de varias tecnologías, son las siguientes:

- **Internet de las Cosas (IoT)** (ver apartado 8.1), esta tecnología ha servido para dotar al proyecto de un servicio sumamente importante, poder enviar mensajes a una base de datos en tiempo real alojada en la nube a través de la cuál se puede saber la localización geográfica del vehículo que se encuentra en una situación de riesgo.
- **Inteligencia artificial (IA)** (ver apartado 8.2), útil para la realización del asistente de voz inteligente, debido a la necesidad de transcribir audio

capturado por el micrófono en la respuesta del usuario a texto, para realizar un procesamiento de los datos introducidos y poder asociar esa entrada de datos a un patrón concreto con el que dar una respuesta más ajustada a las necesidades del usuario.

- **Deep Learning** (ver apartado 8.3), está tecnología es imprescindible en la realización del proyecto, ya que los modelos de entrenamiento existentes tanto para la detección de somnolencia como para el chat de voz se basan se en redes neuronales una de ellas convolucional y la otra en artificial basada en redes densas.
- **Machine Learning** (ver apartado 8.4), tecnología imprescindible a día de hoy, ya que gracias a ella se ha conseguido que las máquinas aprendan automáticamente a partir de los datos.

Por otro lado, los componentes necesarios para poder ejecutar el software son, la necesidad de instalar todas sus librerías en el equipo, tener un equipo medianamente potente y lo más importante es, que disponga de cámara y altavoces y de una conexión a internet. Las bibliotecas se pueden ver cuáles son en los apartados correspondientes de cada bloque funcional **(9.1.1, 9.2.1, 9.3.2),** además, se especifica para que se usa cada una de ellas y se indica como proceder a su instalación.

4.2. Recursos Hardware

En cuanto a recursos hardware, se van a mencionar los elementos necesarios para la ejecución del proyecto. Son los siguientes:

- **Raspberry PI 4B (Ver apartado 10.1.1),** es un mini ordenador que se encargará de ejecutar el proyecto, su sistema operativo está basado en Linux y es ideal para este tipo de proyectos desarrollados en Python.
- **Cámara nativa Raspberry PI (ver apartado 10.1.2),** necesaria para poder llevar a cabo la detección de somnolencia en tiempo real, mediante el monitoreo del parpadeo de los ojos del usuario.
- **Altavoz (ver apartado 10.1.6),** este elemento es vital tanto para la salida de audio del asistente de voz inteligente como para la señal acústica de emergencia, cabe la posibilidad de que esté conectado mediante puerto Jack o USB, para este proyecto se ha hecho uso de la tecnología Bluetooth, reduciendo así la necesidad de añadir cableado al prototipo físico.

- **Micrófono USB (ver apartado 10.1.5),** este dispositivo es necesario para la captura de audio que posteriormente será transcrito y procesado por el asistente virtual inteligente.
- **Punto de acceso (ver apartado 10.1.7),** para poder cumplimentar con existo el tercer bloque (ver apartado 9.3), es necesaria su utilización con objeto de poder enviar mensajes a la nube en caso de que el usuario experimente somnolencia profunda. Además, será útil para descargar el proyecto desde Internet, ya que Raspberry cuenta con la posibilidad de disponer de interfaz gráfica, de modo que nos ofrece una mayor facilidad para disponer del proyecto en el dispositivo Raspberry.

Por otro lado, en cuanto a aplicaciones se hará uso de dos herramientas vitales para la conexión remota con el dispositivo, y proceder a controlarlo sin necesidad de disponer de accesorios concretos que tendrían que estar dedicados exclusivamente a la raspberry como monitor, teclado y ratón, en el apartado de **Anexos,** se pueden ver los pasos a seguir con las siguientes aplicaciones para conectarnos de forma remota, las aplicaciones son las siguientes:

- **IpScanner (Anexo II.** Acceso remoto a Raspberry**,** es un software que corre en el equipo sin necesidad de instalación, será útil para encontrar la dirección IP de dispositivo en un rango determinado de direccionamiento IP
- **RealVNC (Anexo II.** Acceso remoto a Raspberry**,** es una aplicación que servirá de ayuda para conectar nuestro equipo, en este caso un portátil, con la Raspberry PI de forma remota y tener acceso a su interfaz gráfica y poder manipularla cómodamente.

5. Motivación

La motivación por realizar este proyecto surge de la necesidad de realizar algo creativo y poco conocido en nuestro entorno, además de la posibilidad de poder realizar una fusión con temas relacionados directamente con nuestro campo, como lo es el Internet de las Cosas (IoT) y el aprendizaje automático.

Uno de los retos más apasionantes era poder combinar tecnologías, como la inteligencia artificial y el machine learning con el Internet de las cosas (IoT). Resultaba interesante poder llevar a cabo la creación de un sistema que fuera capaz de enviar de

información relevante de manera instantánea con objeto de ayudar a tener asistencia de la forma más rápida posible.

Por otro lado, es importante recalcar que además de un sistema que detectará somnolencia, también surgía la necesidad de crear un asistente virtual que ayudará a persuadir al usuario en la toma de decisión, haciendo que escoja la más correcta ante la aparición de ese tipo de situaciones desafortunadas, de modo que la creación de un sistema que entendiera y comprendiera tu estado, resultaba estimulante para la realización del mismo.

Por esas razones, este proyecto resultaba atractivo y la vez un reto, ya que las nociones en Python eran básicas. Por tanto, este TFG ha servido como aprendizaje de cara al futuro tanto para el desarrollo académico, laboral y personal.

6. Lenguaje de programación

6.1. Python

Para la consecución de los objetivos del proyecto tanto a nivel genérico de software como en la creación del prototipo físico, el lenguaje más recomendable analizando las tareas a realizar es Python. Dicho lenguaje es muy utilizado a día de hoy en los campos de ciencias de datos e inteligencia artificial de modo que es la mejor opción para realizar un procesamiento de datos como en este caso ocurre con la captura de los fotogramas y el procesamiento para determinar si los ojos se encuentran cerrados o abiertos y con el aprendizaje automático del que se pueden valer los modelos de entrenamiento ya sea para el asistente virtual inteligente o para el modelo de entrenamiento de detección de somnolencia. [7]

Además, es compatible con multitud de plataformas y se ajusta perfectamente al entorno escogido para el desarrollo del proyecto, como en este caso es Visual Studio Code (ver apartado 7.1). Por otro lado, Python es completamente compatible con el sistema Raspbian, hasta tal punto que cuando se instala Raspbian ya viene Python instalado, por defecto. Por lo que el proyecto genérico basado en la ejecución en cualquier equipo, se puede ejecutar casi directamente en el dispositivo Raspbian sin necesidad de hacer grandes cambios en el código, simplemente ajustando la configuración requerida, es decir, la descarga de las bibliotecas necesarias para la ejecución del proyecto.

Python ofrece las siguientes ventajas y por tanto es altamente recomendable para este tipo de proyecto: [8]

1. **Amplia variedad de bibliotecas y herramientas:** Python tiene una amplia variedad de bibliotecas y herramientas que son específicas para el procesamiento de datos y el aprendizaje automático.
2. **Fácil de leer y entender:** Python se considera uno de los lenguajes de programación más fáciles de aprender y usar. La sintaxis es simple y clara.
3. **Portabilidad:** Python se puede ejecutar en una amplia variedad de plataformas y sistemas operativos, lo que lo hace ideal para proyectos que deben ser portátiles y ejecutarse en diferentes sistemas.

4. **Comunidad activa:** Python cuenta con una gran comunidad de desarrolladores y entusiastas que trabajan constantemente en mejorar el lenguaje y crear nuevas herramientas y bibliotecas.

7. Entorno de desarrollo

7.1. Visual Studio Code

Visual Studio Code ha sido elegido como herramienta de trabajo para el desarrollo del software debido a su gran facilidad para instalar. Además, es un entorno muy intuitivo con mucha capacidad de personalización. La instalación del lenguaje de programación (Python) se hace de manera muy sencilla, haciendo uso del apartado de extensiones que se encuentra en la barra lateral izquierda.

Hoy día es uno de los entornos de programación más utilizado debido a las grandes ventajas que nos ofrece como: [9]

- **Multiplataforma:** Visual Studio Code está disponible para todos los sistemas operativos Windows, GNU/Linux y macOS.
- **IntelliSense:** Esta es una de las grandes ventajas, es capaz de autocompletar y resaltar la sintaxis, proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables, funciones, etcétera.
- **Depuración:** Visual Studio Code incluye la función de depuración que ayuda a detectar errores en el código.
- **Uso de control de versiones:** Tiene GitHub integrado por lo que podemos disponer de todas las funciones que ello conlleva como push, pull, commit, etcétera.
- **Extensiones:** Las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada.

8. Tecnologías empleadas

8.1. Internet de las Cosas (IoT)

El internet de las cosas es una tecnología que ha aumentado notablemente en los últimos años, consiste en la interconexión de dispositivos, objetos y sistemas a través de Internet para permitir la comunicación y la recopilación de datos. Es decir, se trata de

conectar cosas cotidianas, como electrodomésticos, sensores, automóviles y dispositivos móviles, a Internet para que puedan intercambiar información y tomar decisiones en función de esa información. [10]

Un ecosistema de IoT se compone de dispositivos inteligentes dotados de unas determinadas capacidades por las cuales pueden recopilar, enviar y actuar sobre los datos que se adquieren de sus entornos. En ocasiones, los dispositivos de IoT dentro de un mismo ecosistema interactúan entre sí y actúan sobre la información que obtienen unos de otros, dicha información se almacena en la nube y se tiene fácil acceso. Un ejemplo de funcionamiento de un ecosistema de IoT, sería: [11]

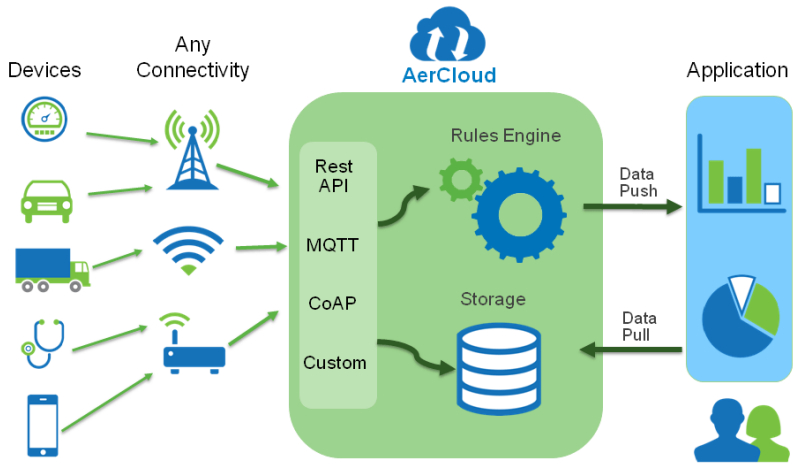

Figura 7. Funcionamiento ecosistema IoT

Fuente: https://aprendiendoarduino.wordpress.com/2018/11/11/ecosistema-iot/

En este proyecto el dispositivo que actúa como inteligente, y recopila los datos, sería la Raspberry PI, internamente a través del software programado. Recogerá datos como la matrícula del vehículo, la geolocalización, fecha y hora. Posteriormente una vez montados estos datos en un formato JSON, serán enviados a una base de datos alojada en internet, esto se realizará a través de peticiones HTTP, además, los datos enviados podrán ser visibles en la plataforma utilizada para este proyecto que pertenece a Google, Firebase.

8.2. Inteligencia artificial (IA)

La inteligencia artificial (IA) es un campo de la informática que se enfoca en la creación de algoritmos y sistemas que imitan la inteligencia humana. En términos

generales, la inteligencia artificial se refiere a la capacidad de una máquina para realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento, la percepción y la comprensión del lenguaje natural. [12]

En este proyecto, los motores de inteligencia artificial han sido muy útiles y beneficiosos, ya que gracias a ellos junto con otras tecnologías se ha podido proceder a la creación de un asistente de voz inteligente, con el que puedes interactuar. De modo que la IA ha tomado gran protagonismo, ya que ha ayudado en la transcripción de audio a texto. Lo cual resulta una tarea compleja y que puede consumir mucho tiempo para un ser humano. La inteligencia artificial (IA) haciendo uso de los algoritmos de procesamiento del lenguaje natural (NLP) y el aprendizaje automático automatizan gran parte del proceso.

El software de transcripción basado en IA puede procesar el audio y convertirlo en texto de manera automatizada. Los algoritmos de IA pueden identificar las palabras habladas y hacer ajustes en tiempo real para tener en cuenta los acentos, las pausas, los ruidos de fondo y otros factores que pueden influir en la precisión de la transcripción.

En el proyecto se han probado diferentes herramientas para la transcripción de audio a texto, además de optimizarse al máximo para obtener la mayor velocidad de procesamiento y de la forma más nítida posible, eliminando el ruido de fondo. En el apartado de reconocimiento del habla y transcripción **(ver apartado 9.2.3)** se detallará más sobre lo conseguido en el proyecto.

8.3. Deep Learning

Es uno de los campos más importantes del Machine Learning, este campo se enfoca en enseñar a los ordenadores a aprender patrones cada vez más complejos a partir de grandes cantidades de datos, utilizando redes neuronales profundas, para que acabe realizando tareas similares a las de los seres humanos, como la identificación de imágenes, el reconocimiento del habla o la realización de predicciones, de forma progresiva. [13]

Luego, lo que se pretende con el Deep Learning es emular el cerebro humano con la ayuda de las redes neuronales profundas, éstas se organizan en varias capas, conectándose unas con otras dando lugar a una estructura mallada capaz de identificar ciertos patrones y poder clasificar diferentes tipos de información. Por tanto, para entenderlo de una forma más simplificada, este algoritmo se centra en recompensar los

comportamientos o resultados deseados y penaliza los menos deseados o erróneos hasta lograr el resultado más favorable y eficiente para la aplicación. [14]

Los tipos de redes neuronales de los que hace uso el Deep Learning de forma más habitual son:

- Redes Neuronales Convolucionales (CNN)
- Redes Neuronales Recurrentes (RNN)
- Redes Generativas Antagónicas (GAN)

Los algoritmos utilizan capas de neuronas artificiales, y cada capa procesa los datos de entrada. La información de la capa se transmite a la siguiente capa y el proceso se repite de forma recursiva mediante el aprendizaje no supervisado, supervisado, o una combinación de ambos. Durante el entrenamiento, las conexiones entre las capas se ajustan con pesos específicos, lo que permite a la red neuronal mejorar su capacidad para hacer predicciones precisas a través de la retropropagación o feed-back.

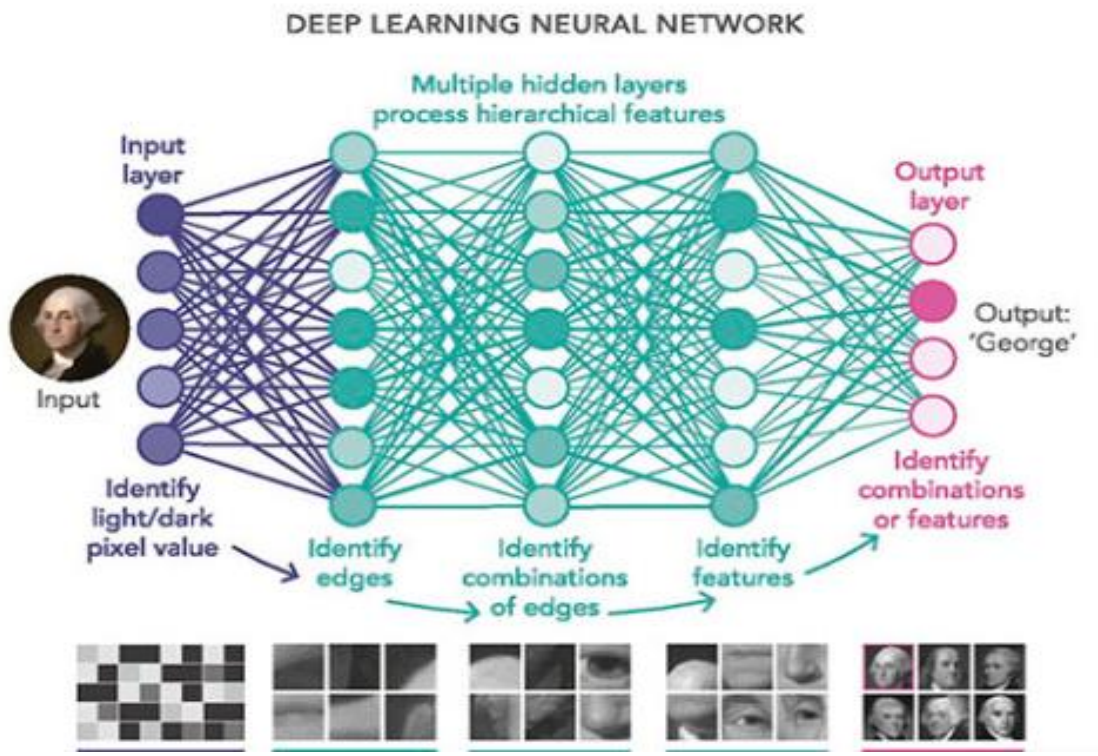

Figura 8. Ejemplo de Deep Learning.

Fuente: https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-1-preludio/

Esta tecnología ha estado muy presente en este proyecto, ya que ha sido necesaria para la realización de los modelos de entrenamiento, este proyecto consta de dos modelos de entrenamiento en los cuales se ha creado una red neuronal convolucional de aprendizaje profundo para uno de ellos, por un lado está el modelo necesario para la detección de somnolencia acorde al estado de los ojos, para este modelo se ha utilizado un dataset de 80 mil fotos, donde se pueden ver distintos tipos de fotografías, así como distintas escenas, con el objetivo de que el modelo aprenda y pueda sin ningún tipo de problema detectar el estado del ojo del usuario, ya sea abierto o cerrado. En cuanto a la creación de la red neuronal específica para este modelo, es necesario realizar una serie de técnicas para evitar el sobre ajuste del modelo de entrenamiento.

Por otro lado, el Deep Learning se ha utilizado para la creación del modelo de entrenamiento del asistente virtual inteligente, la entrada de datos de la que dispone el modelo para poder entrenarse se corresponde con un archivo JSON, donde se pueden encontrar patrones específicos asociados a un comportamiento, así como la respuestas que se asocian a cada patrón, al igual que en modelo anterior, también cabe la necesidad de aplicar técnicas especiales en la red neuronal para que no se produzca un sobreajuste a la hora de entrenar el modelo.

La técnica, para evitar el sobre ajuste consiste en utilizar “Dropout”, este método hará que el porcentaje de neuronas que se le introduzca se apaguen aleatoriamente de forma intermitente durante el entrenamiento, con el objetivo de sacar el máximo rendimiento de las demás.

8.4. Machine Learning

El Machine Learning es considerado un subcampo de la inteligencia artificial, se centra en desarrollar algoritmos capaces de aprender y mejorar automáticamente a través de la experiencia y los datos. Se basa en modelos matemáticos y estadísticos que permiten a las computadoras identificar patrones complejos y realizar tareas específicas sin una programación explícita.

El ML (Machine Learning), consta de tres categorías de aprendizaje:

- Aprendizaje supervisado: resulta ideal para modelos de clasificación, ya que parte de unos datos predefinidos de entrada.

- Aprendizaje NO supervisado: ideal para cuando se quiere la búsqueda de patrones.
- Aprendizaje por refuerzo: ideal para que el prototipo aprenda a base de interactuar con el entorno.

En el desarrollo de este proyecto, se hace uso de técnicas de aprendizaje supervisado, ya que al modelo de entrenamiento se le proporciona una serie de datos predefinidos y clasificados y en base al entrenamiento que realiza posteriormente tendrá que clasificar por un lado el estado de los ojos, en este caso, si se encuentran abiertos o cerrados y por otro lado a raíz de la entrada de datos del usuario, más concretamente el audio transcrito de respuesta, clasificarlo en una familia donde hay determinados patrones, para poder así asociar una respuesta a dicha entrada de datos.

Sistema de detección de somnolencia. Software

9. Bloques funcionales

En este apartado se procederá a explicar los diferentes bloques en los que se divide el proyecto, en este caso, cuenta con tres bloques; detección de somnolencia, asistente de voz inteligente y envío de mensaje de situación de extremo peligro a la nube. Para entender un poco más el funcionamiento genérico del conjunto de bloques, se puede observar la siguiente figura (figura 9), puesta en la Introducción.

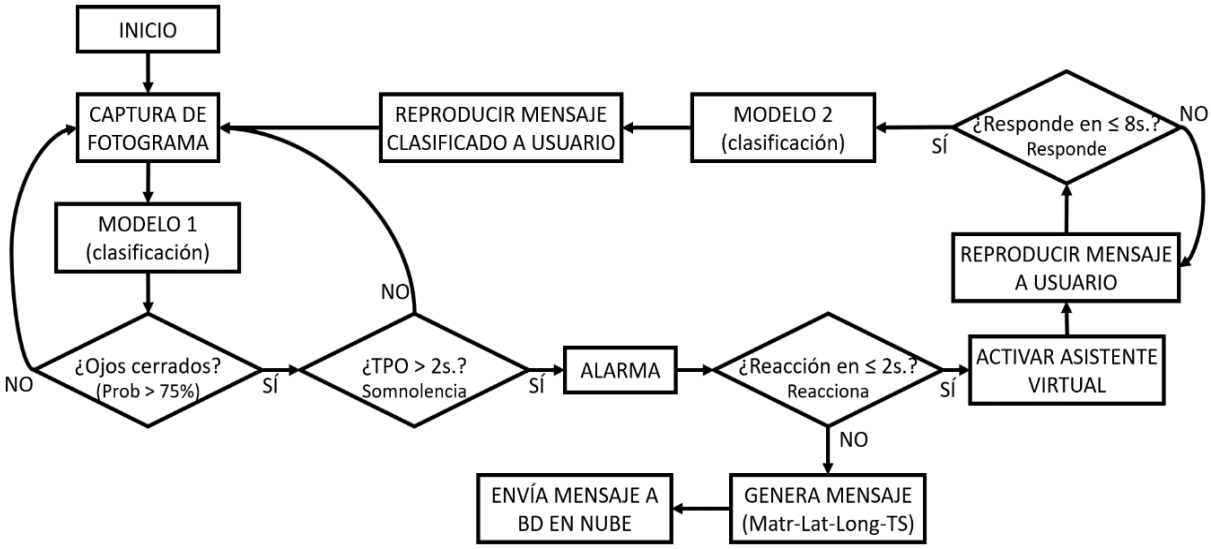

Figura 9. Diagrama general de bloques.

9.1. Bloque 1. Detección de somnolencia

En los próximos subapartados correspondientes al primer bloque funcional que consiste en la detección de somnolencia, se explicarán las bibliotecas necesarias a instalar para asegurar el correcto funcionamiento del proyecto, ya que su desarrollo se basa en torno a ellas. Además, se llevará a cabo una explicación del modelo de entrenamiento específico que se encuentra en este bloque funcional, así como la dedicación de un subapartado específico para entender el funcionamiento concreto del bloque al completo.

9.1.1. Bibliotecas

La instalación de las bibliotecas se puede llevar a cabo de varias formas, se puede realizar a nivel de sistema o bien dentro del entorno virtual donde se ha creado el proyecto. A nivel de entorno virtual, la ventaja que ofrece es que las dependencias solamente se quedarán instaladas ahí y no se producirá ningún conflicto con ningún archivo del sistema. Por el contrario, si se hace a nivel global, las bibliotecas quedarán disponibles para usarse en cualquier proyecto que corra a nivel local. Aclarar, que, en las siguientes figuras, el comando que aparece es específico para el sistema operativo de Windows, por lo que para Linux y para MAC, se deberá realizar la instalación de las bibliotecas de otra manera.

9.1.1.1. Cv2 (OpenCV)

La biblioteca de Python llamada "cv2" se refiere a OpenCV (Open Source Computer Vision Library), se utiliza para el procesamiento de imágenes y visión por computadora. Proporciona una amplia gama de funciones y algoritmos que permiten realizar tareas relacionadas con el procesamiento y análisis de imágenes. Algunas de las principales funciones y características proporcionadas por la biblioteca cv2: [15]

- Lectura y escritura de imágenes y videos.
- Procesamiento de imágenes.
- Detección de características.
- Detección y seguimiento de objetos.
- Visión estéreo y calibración de cámaras.
- Procesamiento de imágenes en tiempo real.

Su instalación se realiza de forma sencilla, haciendo uso del comando pip, instalador de Python con el que se pueden descargar bibliotecas desde el símbolo del sistema, como podemos ver en la siguiente figura (Figura 10):

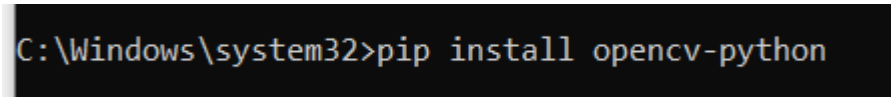

Figura 10. Instalación primaria OpenCV

De la forma anterior se realizará una instalación de los módulos principales, por el contrario, si se quiere tener tanto los módulos principales como los extras, habrá que descargar Open CV de la siguiente forma (ver figura 11):

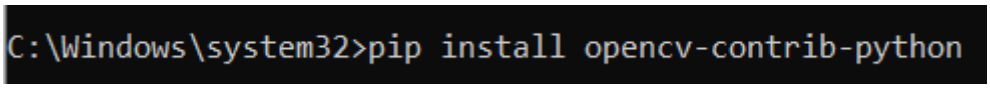

Figura 11. Instalación completa OpenCV

9.1.1.2. Keras

Es una biblioteca desarrollada en Python cuyo objetivo es agilizar la creación de redes neuronales y el aprendizaje automático. Es tan ágil debido a que se puede hacer uso de ella como una API, ya que funciona como un Framework independiente. Proporciona bloques modulares sobre los que se pueden desarrollar modelos complejos de aprendizaje profundo, además trabaja con Tensorflow proporcionando más eficacia a la hora de trabajar en Deep Learning. [16]

Por tanto, está biblioteca ha resultado ser muy útil en este proyecto ya que se ha utilizado en los diversos modelos de entrenamiento empleados, tanto el modelo de entrenamiento de imágenes para la detección del estado del parpado, como para el modelo de entrenamiento de la red neuronal para el asistente virtual.

Para su instalación simplemente se debe ejecutar el siguiente comando, que se puede ver en la figura 12.

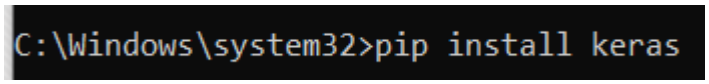

Figura 12. Instalación keras

9.1.1.3. Numpy

Es una biblioteca especializada en el cálculo numérico y análisis de datos, Numpy crea un objeto de matriz multidimensional de alto rendimiento y objetos derivados con los que se puede trabajar de manera efectiva, ideal para el Machine Learning. [17]

La biblioteca NumPy se utiliza para una variedad de aplicaciones, incluyendo:

- Matrices multidimensionales.
- Cálculos matemáticos.
- Manipulación de datos.
- Integración con otras bibliotecas.

Esta biblioteca ha sido de gran utilidad en este proyecto, ya que ha sido utilizada para tareas de procesamiento, cálculo de dimensiones para ajustar la detección del ojo al tamaño especifico con el que trabaja el modelo de entrenamiento, etcétera.

Para realizar la instalación de la biblioteca se debe de hacer uso del comando que aparece en la siguiente figura (figura 13).

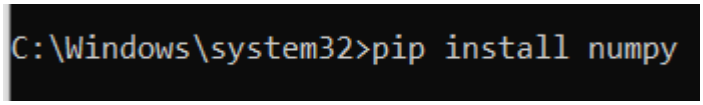

Figura 13. Instalación Numpy

9.1.1.4. Math

Math es una biblioteca incorporada en Python que proporciona un conjunto de funciones matemáticas y constantes. Esta biblioteca permite realizar operaciones matemáticas más avanzadas y complejas que las operaciones básicas proporcionadas por el lenguaje. Es importante tener en cuenta que la biblioteca math trabaja principalmente con números de punto flotante (float). [18]

En este proyecto una de las aplicaciones que tiene es para calcular el número de segundos que se encuentra los ojos cerrados, ya que se obtiene el valor absoluto de la diferencia entre el tiempo inicial momento en el que los ojos cambian de estado de abierto a cerrado y el tiempo que se va contando a medida que los ojos permanecen cerrados.

Esta biblioteca ya viene por defecto instalada con Python desde la versión 3.4, por lo que no es necesario ejecutar ningún comando para disponer de ella.

9.1.1.5. Time

La biblioteca Time (Tiempo, en inglés), como su propio nombre indica ofrece diversas herramientas para poder trabajar haciendo uso de nuestro huso horario. Algunas de las funciones más representativas que podemos encontrar en esta librería son: [19]

- **time.time():** devuelve el tiempo actual en segundos desde el 1 de enero de 1970 (conocido como el "epoch" o "época" en inglés). Es útil para medir el tiempo transcurrido en un programa.
- **time.sleep(segundos):** detiene la ejecución del programa durante el número especificado de segundos.

- **time.localtime():** devuelve la hora local actual en forma de una estructura de tiempo. Esta estructura contiene campos como el año, mes, día, hora, minuto, segundo, etc.
- **time.strftime(formato, estructura_de_tiempo):** toma una estructura de tiempo y la formatea según el formato especificado.
- **time.gmtime():** similar a time.localtime(), pero devuelve la hora UTC (Tiempo Universal Coordinado) en lugar de la hora local.

Todos los husos horarios se definen con relación al tiempo UTC cuyo huso horario se sitúa centrado sobre el Meridiano Cero o Meridiano de Greenwich. La biblioteca también incluye otras funciones para medir el tiempo de ejecución, manipular fechas y horas, así como trabajar con temporizadores y marcas de tiempo.

En este caso, caso se ha hecho uso de las funciones time.time() y time.strtime(), la instalación de la biblioteca no es necesaria puesto que viene integrada con las últimas versiones de Python.

9.1.1.6. Pygame

Pygame es un conjunto de módulos y funciones diseñados para facilitar el desarrollo de videojuegos y aplicaciones multimedia interactivas en Python, también ayuda en la reproducción de sonidos, motivo por el cuál ha sido usada en este proyecto.

Concretamente está biblioteca es utilizada en el proyecto para reproducir el sonido de alerta cuando se detecta un micro sueño en el usuario, con el objetivo de que este reaccione ante dicho estimulo.

Con esta biblioteca no sucede como con las dos anteriores, por lo que hay que instalarla a parte ya que no viene integrada en Python, al igual que en bibliotecas anteriores se debe ejecutar el comando que aparece en la siguiente figura (figura 14).

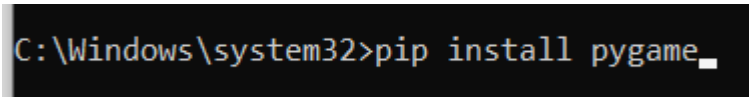

Figura 14. Instalación pygame

9.1.2. Modelo de entrenamiento

Para la detección de somnolencia a través del monitoreo del parpadeo, se ha realizado un modelo de entrenamiento que consta de un dataset con más de ochenta mil fotografías, donde se pueden apreciar distintos tipos de ojos, así como posiciones de los parpados y claridad con la que se ve la imagen. Además, se ha hecho uso de una red neuronal convolucional, que consta de diferentes capas con un modelo de optimización específico.

9.1.2.1. Dataset

Un dataset es un conjunto de datos estructurados y organizados que se utiliza para realizar análisis, investigaciones y entrenar modelos de machine learning. Puede consistir en diferentes tipos de datos, como texto, imágenes, videos, audio, tablas o cualquier otra forma de información. [20]

El dataset utilizado en este proyecto ha sido cogido de internet en [21], el dataset cuenta con más de 80 mil fotos donde está organizado de tal forma que el 90% de las fotografías se usan para el modelo de entrenamiento y el 10% restante se usa para la prueba. Las propias fotografías también se encuentran ordenadas de una determinada forma, es decir, en la siguiente imagen (figura 15) se puede observar que el nombre de la imagen va regido por unos códigos.

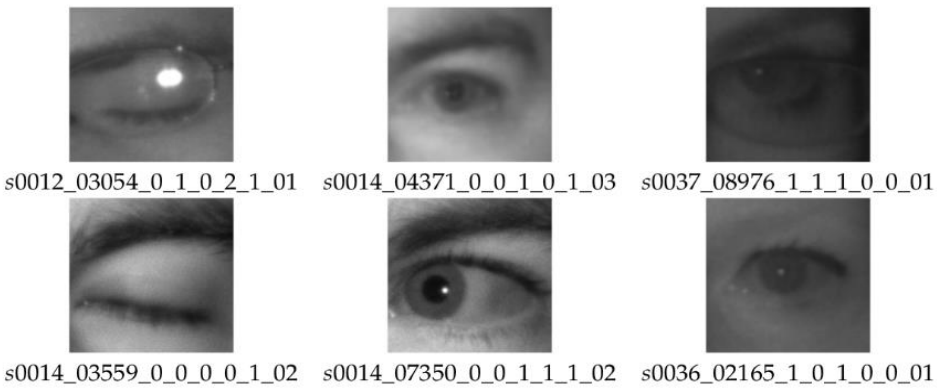

Figura 15. Clasificación imágenes dataset

Fuente: **http://mrl.cs.vsb.cz/eyedataset**

La clasificación de los códigos del título de las imágenes se corresponde a:

- **ID del sujeto [sXXXX];** en el conjunto de datos, datos de 37 personas (33 hombres y 4 mujeres).
- **ID de la imagen [xxxxx]**; el conjunto de datos consta de 84.898 imágenes.
- **Sexo [0 - hombre, 1 - mujer]**
- **Gafas [0 - no, 1 - sí]**
- **Estado del ojo [0 - cerrado, 1 - abierto]**
- **Reflejos [0 - ninguno, 1 - pequeño, 2 - grande]**
- **Condiciones de iluminación [0 - mala, 1 - buena]**
- **ID del sensor [01 - RealSense, 02 - IDS, 03 - Aptina]** el conjunto de datos contiene las imágenes capturadas por tres sensores diferentes (sensor Intel RealSense RS 300 con una resolución de 640 x 480, sensor IDS Imaging con una resolución de 1280 x 1024 y sensor Aptina con una resolución de 752 x 480)

Por tanto, la estructuración del Dataset queda comprendida de la siguiente forma:

TEST	Ojos cerrados	1.566 imágenes
	Ojos abiertos	1.657 imágenes
TRAIN	Ojos cerrados	40.380 imágenes
	Ojos abiertos	41.295 imágenes

Tabla 1. Organización del Dataset

9.1.2.2. Red neuronal convolucional

En la red neuronal empleada en el proyecto para este bloque especifico lo primero que se ha realizado es la creación de una instancia a través del modelo Sequential de Keras, donde posteriormente se irán añadiendo capas las capas. Las primeras capas agregadas son capas convolucionales que aplican filtros a la entrada para extraer características relevantes de las imágenes, se hace uso de diferentes números de filtros, concretamente 32 y 64, con un tamaño del kernel de 3x3, y se usa la función de activación ‘relu’ para potenciar la no linealidad de la red convolucional.

Después de cada capa convolucional, es conveniente agregar una capa de ‘Max Pooling’. Estas capas reducen la dimensionalidad de las características espaciales y ayudan a reducir la cantidad de parámetros en la red.

Posteriormente, se agrega una capa de “Dropout” con una tasa de 0.25. Esta capa apaga aleatoriamente un 25% de las neuronas durante el entrenamiento, lo que ayuda a prevenir el sobreajuste y mejora la generalización del modelo.

Se añade una capa Flatten que aplana las características obtenidas de las capas anteriores en un vector unidimensional. Esto permite conectar las capas convolucionales con las capas completamente conectadas. A continuación, se agrega una capa completamente conectada (Dense) con 128 neuronas y función de activación 'relu'. Esta capa realiza una combinación lineal de las características extraídas por las capas convolucionales. Además, para aumentar la funcionalidad se añade otra capa ‘Dropout’ con una tasa del 0.5, que ayudará a reducir el sobreajuste aún más.

La última capa que se añade es una capa completamente conectada (Dense) con 2 neuronas y función de activación 'softmax'. Esta capa produce las probabilidades de clasificación para las dos clases de salida. La función de activación 'softmax' garantiza que las probabilidades sumen 1 y permite interpretar la salida como la probabilidad de pertenecer a cada clase.

9.1.2.3. Compilación del modelo y resultados

Después de definir la arquitectura del modelo, se compila utilizando el optimizador 'adam', que es un método de optimización popular para el entrenamiento de redes neuronales. Se especifica la función de pérdida 'categorical_crossentropy' que es adecuada para problemas de clasificación con más de dos clases. También se agrega la métrica de 'accuracy' para evaluar el rendimiento del modelo durante el entrenamiento.

Finalmente, se entrena el modelo utilizando los datos de entrenamiento (train_batch) y los datos de validación (valid_batch). Se especifica el número de épocas (epochs) que representa el número de veces que el modelo recorrerá todo el conjunto de datos de entrenamiento. Se ajustan los parámetros steps_per_epoch y validation_steps que indicarán cuántos pasos se deben tomar en cada época para recorrer todo el conjunto de datos de entrenamiento y validación, respectivamente.

Una vez que se ha compilado el modelo se pueden observar los siguientes resultados en la figura 16.

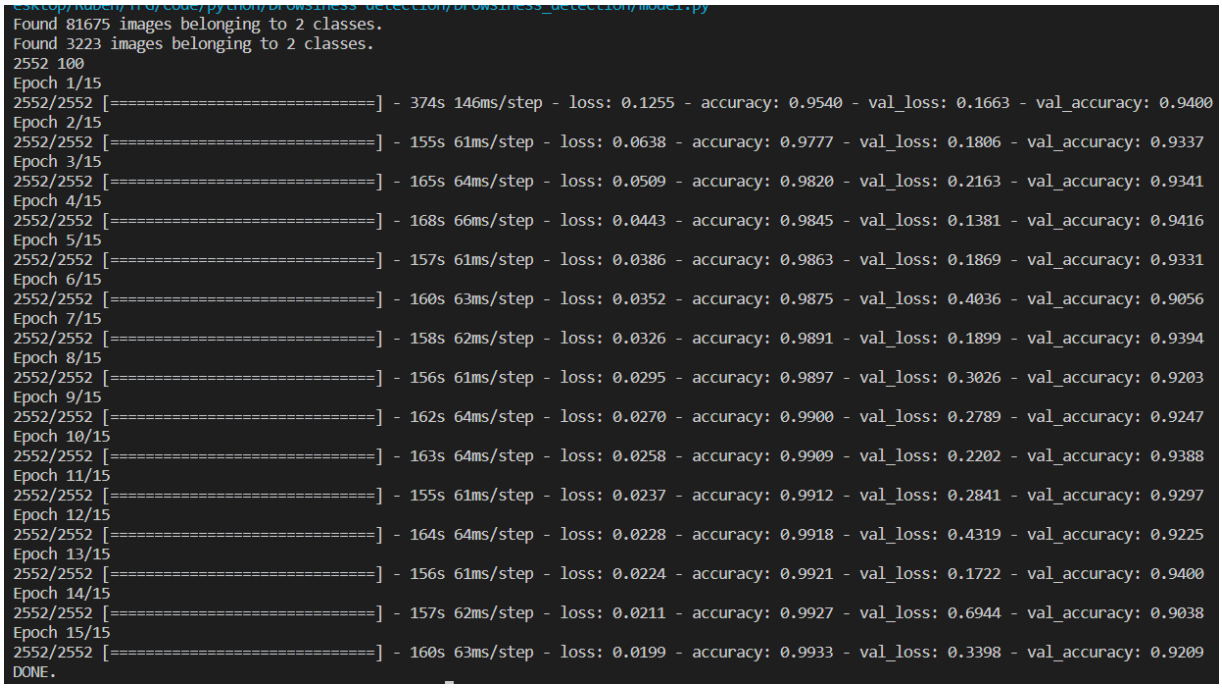

Figura 16. Resultados del entrenamiento

9.1.3. Funcionamiento del bloque

Una vez explicadas las diferentes partes de las que se compone el bloque de detección de somnolencia, se va a proceder a explicar con detalle el funcionamiento del mismo, antes se puede observar un diagrama de bloques de lo que sería el funcionamiento lógico de dicho bloque.

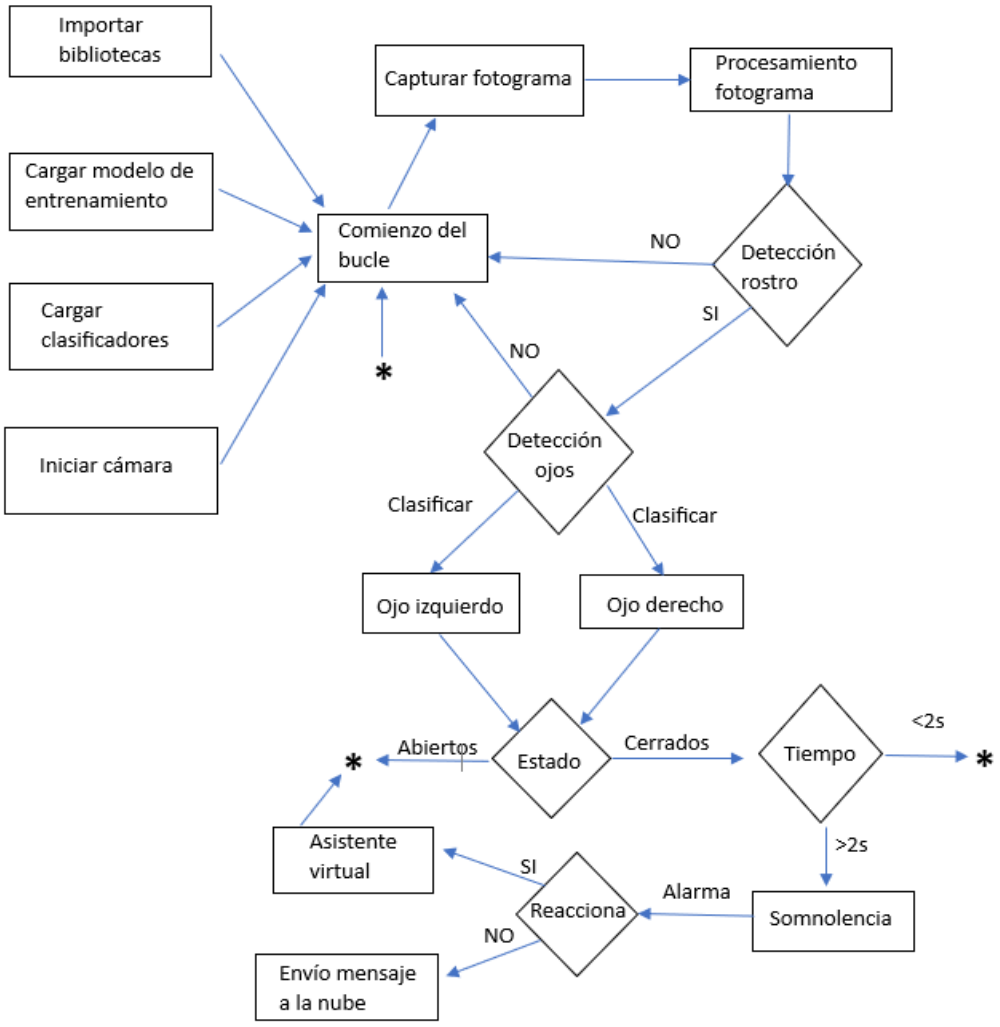

Figura 17. Diagrama de bloques para la detección de somnolencia

Para llevar a cabo la monitorización del parpadeo, primero, se han de cargar los archivos cascade. Los archivos cascade son archivos XML que contienen información sobre características visuales específicas de un objeto que se desea detectar. Estos archivos son utilizados por los clasificadores Haar en la detección de objetos en imágenes.

Los clasificadores Haar son algoritmos de aprendizaje automático que se basan en características visuales simples, como bordes, esquinas y texturas, para identificar objetos en una imagen. Estos clasificadores se entrenan utilizando un conjunto de imágenes positivas (que contienen el objeto de interés) y negativas (que no contienen el objeto de interés). [22]

En este proyecto se ha hecho uso de tres archivos cascade: "haarcascade_frontalface_alt.xml" para la detección de rostros de manera frontal, "haarcascade_lefteye_2splits.xml" para la detección del ojo izquierdo y "haarcascade_righteye_2splits.xml" para la detección del ojo derecho. Estos archivos contienen información sobre las características visuales relevantes para la detección de cada objeto y se utilizan en conjunto con la función detectMultiScale de OpenCV para identificar y localizar las regiones de interés en la imagen.

Por otro lado, es necesario cargar el modelo de entrenamiento creado y entrenado previamente, ya que este será de vital importancia para juzgar si los ojos se encuentran abiertos o cerrados.

Lo siguiente sería crear el bucle infinito que está constantemente evaluando si los ojos se encuentran en un estado u otro, el bucle o el programa será infinitos hasta que el usuario quiera. Dentro de este bucle se producirá todo el desarrollo del bloque.

Primero se deberá establecer la conexión con la cámara. Posteriormente, se procederá a realizar la captura de fotogramas, haciendo uso de una herramienta de OpenCV, esta herramienta convierte los fotogramas a escala de grises, lo cual es útil debido a que la información de color en la captura de los ojos es irrelevante. A continuación, se procede a realizar la detección de caras en los fotogramas capturados, haciendo uso de la herramienta 'DetectMutiScale' que busca patrones específicos en la imagen y devuelve las coordenadas de las regiones donde se encuentran las caras.

Posteriormente, mediante un bucle for, se recorren las caras detectadas realizándose una serie de acciones:

1. Dibuja un cuadrado alrededor de la cara detectada.
2. Recorrer tanto el ojo derecho como izquierdo, realizándose las siguientes acciones:
    a. Extraer el área de interés, conocida como ROI, y se preprocesa para que coincida con el formato de entrada del modelo CNN.
    b. Realiza algunas operaciones de preprocesamiento en el área del ojo para prepararla para la predicción del modelo.
    c. Hace una predicción utilizando el modelo CNN y determina si el ojo está abierto o cerrado.

d. Actualiza una variable creada denominada en este caso “lbl” (correspondiente a la etiqueta que nos dice si el ojo está abierto o cerrado) según el resultado de la predicción.

3. Después de haber obtenido la información correspondiente a los dos ojos, siendo ésta almacenada en una tupla, se comprueba y si es mayor o menor que un determinado valor que se asigne en función de la precisión que queramos, nos dirá si el ojo se encuentra abierto o cerrado.
4. Como se comentó anteriormente, se define microsueño a un episodio breve e involuntario de sueño que ocurre durante breve periodo de tiempo. Por esa razón disponemos de una marca de tiempo la cual se inicia estando los ojos abiertos, quedándonos con el último segundo que estuvo abierto, en el momento que se cierra, se entra en una condición en la que cada instante que pasa se va contabilizando los segundos, de modo que si el tiempo que permanecen cerrados los ojos es superior a dos segundos se detecta un micro sueño y empieza a sonar una alarma de emergencia, a la cual el usuario debe reaccionar antes de los 5 segundos, si no, se activará la conducción autónoma inteligente del vehículo y se enviara un mensaje a la nube para los servicios de emergencia. En caso de que el usuario reaccione y abra los ojos antes de esos 5 segundos, procede a activarse el asistente de voz inteligente, el cuál preguntara al usuario por su estado, y en función de la entrada de datos que le proporcione le aconsejará cual es la mejor opción para su seguridad.

9.2. Bloque 2. Asistente virtual

En este bloque se va a proceder a explicar las diferentes partes que componen el asistente virtual y que son necesarias para el correcto funcionamiento del mismo, de modo que, se explicaran las bibliotecas necesarias, el modelo de entrenamiento que se ha realizado para dotar de inteligencia al asistente virtual y la integración de voz dentro del mismo, además se dedicará un apartado específico para explicar el funcionamiento del bloque, después de haber adquirido las nociones básicas de los componentes que dan lugar al asistente de voz.

9.2.1. Bibliotecas

En los siguientes subapartados se mencionarán las bibliotecas necesarias para el correcto funcionamiento del bloque, además se indicará como realizar la instalación de las mismas. Algunas de las bibliotecas usadas se corresponden con las explicadas en el bloque anterior, de modo que, simplemente se mencionaran:

- **Numpy** (ver apartado **9.1.1.3**).
- **Time** (ver apartado **9.1.1.5**)
- **Keras** (ver apartado **9.1.1.2**)

9.2.1.1. Random

La biblioteca random de Python proporciona herramientas capaces de generar números pseudoaleatorios, es muy útil cuando se necesita introducir un grado de aleatoriedad. Algunas de las funciones más comunes son: [23]

- **random():** Genera un número decimal aleatorio entre 0 y 1.
- **randint(a, b):** Genera un número entero aleatorio dentro de un rango especificado.
- **choice(seq):** Devuelve un elemento aleatorio de una secuencia dada.
- **shuffle(seq):** Mezcla aleatoriamente los elementos de una secuencia.
- **sample(population, k):** Genera una muestra aleatoria de tamaño k a partir de una población dada sin reemplazo.

Una de las utilidades que tiene en este proyecto es la de escoger de forma pseudo aleatoria alguna de las respuestas predeterminadas a una serie de patrones y su comportamiento. Su instalación no es necesaria puesto que Python nos provee de dicha biblioteca.

9.2.1.2. JSON

La biblioteca JSON en Python es un módulo incorporado que proporciona funciones para trabajar con el formato de datos JSON (JavaScript Object Notation). Proporciona funciones para realizar las siguientes operaciones: [24]

- **Codificación (serialización):** La función json.dumps() convierte un objeto de Python en una cadena de texto en formato JSON.

- **Decodificación (deserialización):** La función json.loads() convierte una cadena de texto en formato JSON en un objeto de Python.

En este proyecto ha sido extremadamente útil para realizar el envío de mensajes a la nube mediante HTTP, además también se ha utilizado para cargar el fichero de datos de para el modelo de entrenamiento del asistente virtual ya que la entrada de datos para el entrenamiento proporcionada a este bloque, es un formato JSON, que como se puede observar a continuación en la siguiente figura (figura 18), se esquematiza mediante un comportamiento que actúa como etiqueta, unos patrones que serían los relacionados a la entrada de datos del usuario y sus respuestas asociadas a los patrones que se producen dentro de cada familia.

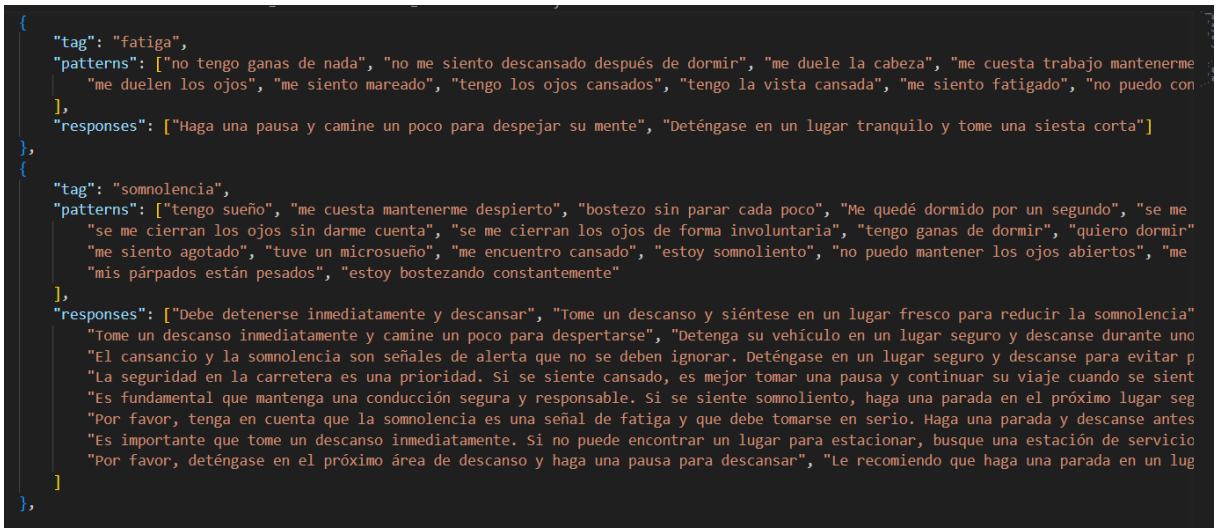

Figura 18. Archivo JSON

No requiere de instalación por parte del usuario puesto que ya viene integrada con Python.

9.2.1.3. Io

La biblioteca io ofrece una gran facilidad para trabajar con operaciones de entrada/salida, ya sea en formato texto, binario o sin formato. Esta biblioteca permite definir por ejemplo las clases de flujos de entrada y salida, también proporciona las clases que envuelven operaciones de archivos y permiten interactuar con ellos, estas últimas clases se refieren a la permisividad de leer y escribir archivos de una manera eficiente y flexible. [25]

No requiere de instalación por parte del usuario ya que viene integrada en Python.

9.2.1.4. Os

Es una biblioteca que proporciona una serie de herramientas para interactuar con el sistema operativo en el que se está ejecutando el programa. Permite acceder y manipular una variedad de características y funcionalidades del sistema operativo, como el sistema de archivos, los procesos, las variables de entorno y más. [26]

Suele ser utilizada y en este caso ha sido utilizada, para obtener las rutas de determinados ficheros necesarios para el correcto funcionamiento del programa. No requiere acción de instalación por parte del usuario puesto que esta biblioteca viene integrada en Python.

9.2.1.5. Dill

Dill es una extensión del módulo pickle que ayuda a serializar y deserializar objetos de Python a la mayoría de los tipos que tiene Python integrado. La serialización es el proceso de convertir un objeto en un flujo de bytes, y lo contrario es volver a convertir un flujo de bytes en una jerarquía de objetos de Python. dill se puede usar para almacenar objetos python en un archivo. [27]

Para realizar la instalación de esta biblioteca, se debe ejecutar el siguiente comando, como se observa en la siguiente figura (figura 19).

Figura 19. Instalar Dill

9.2.1.6. NLTK (NLP, en español)

Para comprender la importancia de esta biblioteca, primero es necesario hablar del lenguaje de procesamiento natural (NPL, siglas en inglés). NPL es una rama de la inteligencia artificial que se centra en ser intermediario entre máquina y humano, teniendo por objetivo el entendimiento y la comprensión total entre ambos haciendo uso del lenguaje natural. La mayoría de las técnicas del NPL se centran en el aprendizaje automático con el que tratan de comprender y dar sentido al significado de los diferentes idiomas existentes. [28]

NLTK (Natural Language ToollKit, en inglés) es la biblioteca más popular para el procesamiento natural del lenguaje debido a su simplicidad y efectividad. NLTK se utiliza para una variedad de tareas relacionadas con el procesamiento del lenguaje natural, como la tokenización de texto (dividir el texto en palabras o frases), el etiquetado gramatical (asignar etiquetas a las palabras según su función gramatical), el análisis sintáctico (análisis de la estructura gramatical de las oraciones), el análisis semántico (extracción de significado de las oraciones), la desambiguación léxica (resolver ambigüedades de palabras), el análisis de sentimientos (determinar la polaridad emocional en el texto), entre otras. [28]

En este proyecto se ha hecho uso de muchas de sus técnicas más comunes como, por ejemplo:

- **Tokenizar:** es la capacidad de poder separar el texto por palabras y clasificarlas como entidades denominadas 'tokens'.
- **Lematización**: es un proceso mediante el cual se busca reducir las palabras a su forma base o lema. El lema de una palabra es su forma canónica o fundamental.
- **Bag of words**: es una forma de representar el texto como una colección desordenada de palabras y sus frecuencias.

Para realizar la instalación de dicha biblioteca, es necesario ejecutar el siguiente comando, como aparece en la siguiente figura (figura 20).

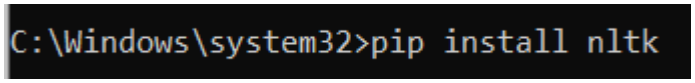

Figura 20. Instalar NLTK 1

Además de descargar esta biblioteca para el lenguaje del procesamiento natural, también ha sido necesaria la descargar de recursos adicionales, estos recursos se ponen en el propio código fuente y se puede ver en la figura 21.

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger') # descarga modelos de etiquetado POS
nltk.download('cess_esp') # descarga corpus en español
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
```

Figura 21. Recursos adicionales NLTK 1

9.2.1.7. Pydub y ffmpeg

PyDub es una biblioteca de manipulación de audio en Python. Se usa para realizar diversas operaciones en archivos de audio, como cargar, reproducir, recortar, combinar, convertir formatos, aplicar efectos, ajustar el volumen, etcétera. Es importante para usarla tener instalado “ffmpeg”. FFmpeg es necesario cuando se utiliza PyDub porque proporciona las capacidades fundamentales de procesamiento de audio, como la conversión de formatos, el recorte, la combinación, el ajuste de volumen, la codificación y decodificación de códecs. Algunas de las tareas que se pueden realizar con pydub son: [29]

- Cargar y guardar archivos de audio en diferentes formatos. (En este proyecto se han usado los formatos de audio flac, ya que era el formato más rápido de procesar).
- Reproducir archivos de audio.
- Extraer fragmentos específicos de un archivo de audio.
- Generar tonos y señales de audio.
- Convertir archivos de audio de un formato a otro.

Para realizar la instalación de pydub, se ejecutará el comando que aparece en la siguiente figura (figura 22).

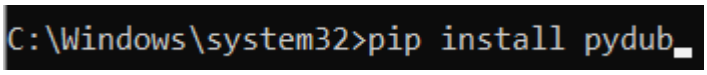

Figura 22. Instalar pydub 1

Instalar ffpmeg es un poco más laborioso, los pasos a seguir son los siguientes:

1. Ir al sitio web oficial de FFmpeg en https://www.ffmpeg.org/.
2. En la sección de "Download", buscar la opción "Windows" y hacer clic en ella.
3. Descargar el archivo ZIP correspondiente a la arquitectura del sistema (en este caso 64 bits).
4. Extraer el contenido del archivo ZIP.
5. Agregar la ruta de la carpeta de FFmpeg extraída al entorno PATH de Windows.

9.2.1.8. Speech Recognition

Herramienta que permite la transcripción y reconocimiento de voz. Es una de las herramientas principales cuando se quiere capturar audio por computador ya que puede capturar y reconocer voz en tiempo real a través de un micrófono, lo que permite la interacción en tiempo real con aplicaciones basadas en voz., además es una herramienta, capaz de transcribir audio utilizando APIs de diferentes empresas como por ejemplo Google o IBM.

En este proyecto Speech_recognition ha sido muy útil para la grabación de audio a través del micrófono, ya que la transcripción de audio se ha realizado mediante Whisper AI (ver apartado 9.2.1.10) debido a las grandes características que proporciona.

La forma de realizar la instalación de esta biblioteca es a través de la ejecución del siguiente comando.

```
C:\Windows\system32>pip install SpeechRecognition
```

Figura 23. Instalar SpeechRecognition.

9.2.1.9. Pyttsx3

Pyttsx es una biblioteca que proporciona una forma sencilla de convertir texto en voz humana y controlar diferentes aspectos de la generación de voz. Esto puede ser utilizado para mejorar la accesibilidad de las aplicaciones, proporcionar información hablada a los usuarios y crear experiencias interactivas basadas en voz. Con esta biblioteca se puede controlar la velocidad y el volumen del audio, así como la posibilidad de personalizar el tono de voz y reproducir la voz generada en tiempo real o guardarla como un archivo de audio. [30]

Para realizar la instalación de dicha biblioteca se debe ejecutar el siguiente comando que aparece en la figura 24.

```
C:\Windows\system32>pip install pyttsx3
```

Figura 24. Instalar pyttsx3 1

9.2.1.10. Whisper

Whisper es un sistema de reconocimiento automático de voz (ASR), que ha sido entrenado con más de 680 mil horas de audio en diversas lenguas, lo que lo hace fuertemente versátil debido a una mayor solidez a los acentos, el ruido de fondo y el lenguaje técnico. Es la herramienta ideal para la transcripción de audio a texto, así como la posibilidad de traducción de cualquier lengua al inglés. [31]

Whisper se basa en una red neuronal profunda que es capaz de entender el lenguaje humano y procesar la información de manera similar a como lo hace el cerebro humano. Puede ser integrada en diferentes aplicaciones y sistemas de comunicación, como chatbots, asistentes virtuales, correo electrónico, redes sociales, entre otros. Una de las características más interesantes de Whisper es su capacidad para analizar el sentimiento de las conversaciones y proporcionar información sobre cómo se sienten los usuarios. [32]

Su estructura lo hace ser un sistema altamente efectivo y sofisticado, por cómo se han estructurado los datos de entrada para su entrenamiento, se basa en una arquitectura de codificadores y decodificadores con Transformers, que necesita de una entrada de datos, ya sea por ejemplo un archivo de audio, con el que posteriormente creará una representación vectorial de cada palabra, además se realiza una codificación posicional de cada palabra, posteriormente se pasa por una capa de seis codificadores que posteriormente pasará por otra capa de seis decodificadores, estos últimos son los encargados de calcular las probabilidades que hay de que se generen nuevas palabras en base de las palabras originales, previamente codificadas. Además, los decodificadores realizan otras diversas tareas como detectar el lenguaje en el que se está hablando, poder traducir un texto, detectar el idioma del texto, detectar cuando se está hablando y cuando no, reconocer signos de puntuación, etcétera. Es decir, se trata de un modelo de inteligencia artificial basado en redes neuronales de aprendizaje profundo multitarea.

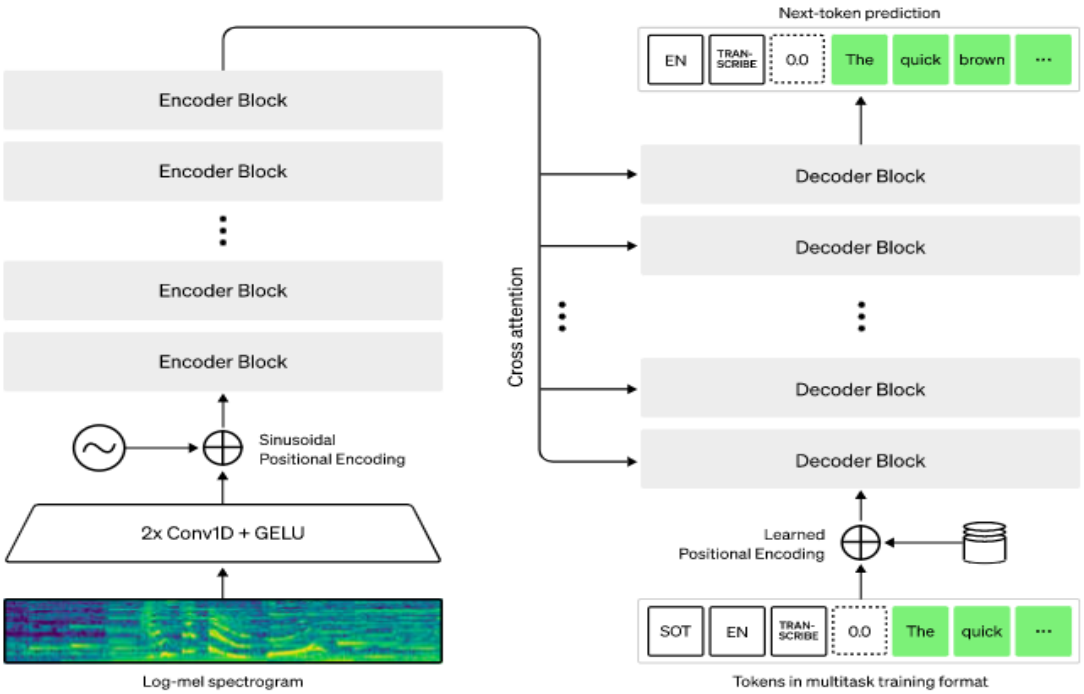

Figura 25. Arquitectura Whisper
Fuente: https://openai.com/research/whisper

Whisper es una herramienta muy potente, pero obviamente para poder explotar todo su potencial, es necesario tener unos medios de nivel elevado, es por ello por lo que Whisper posee diferentes modelos separados en función de la calidad del reconocimiento, aquí se muestra una tabla con las características requeridas para el uso de cada modelo específico. [32]

Tamaño	Parámetros	Modelo multilenguaje	VRAM requerida	Velocidad relativa
Tiny	39 M	tiny	~1 GB	~32x
Base	74 M	Base	~ 1 GB	~16x
Small	244 M	small	~ 2 GB	~6x
Medium	769 M	medium	~ 5 GB	~2x
Large	1550 M	large	~ 10 GB	1x

Tabla 2. Requerimientos en Whisper

Para la instalación de Whisper es necesario realizar los siguientes pasos, por prevención es conveniente instalar whisper haciendo uso del comando pip en el símbolo del sistema como se puede apreciar en la siguiente figura.

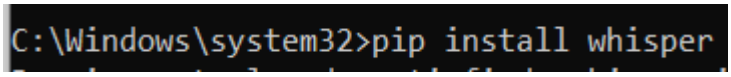

Figura 26. Instalación whisper simple

Pero para que funcione de forma correcta, es necesario instalar whisper desde un repositorio GitHub de OpenAI, de modo que debemos de ejecutar el siguiente comando en la ventana símbolo del sistema. [33]

Figura 27. Instalar Whisper AI OpenAI

9.2.2. Modelo de entrenamiento

A continuación, se va a proceder a explicar las partes de las que consta el modelo de entrenamiento, además se mostrarán los resultados del modelo de entrenamiento una vez compilado.

9.2.2.1. Preprocesamiento de datos

En primer lugar, se ha de realizar un archivo JSON, en el cual se va a crear una estructura de datos siguiendo el siguiente orden:

- **Etiquetas:** Es la palabra clave con la que se van a etiquetar determinados patrones y sus respuestas.
- **Patrones:** Son frases que se suelen decir acorde a un determinado tema específico y que dirá el conductor y serán asociadas con la etiqueta.
- **Respuestas:** Son las respuestas que manejara el chatbot para responder al usuario acorde a la predicción que realice.

Una vez que se tiene el JSON creado, se realizará el modelo de entrenamiento, como es obvio primero se trabajará con el archivo JSON, para ello se deberá cargar y posteriormente se almacenarán los patrones en palabras y las clases de intenciones en listas separadas. Posteriormente, se aplicarán técnicas para la normalización de las palabras haciendo uso de la herramienta “WordNetLemmatizer” de la librería NLTK (NLP,

en español), y además se creará una lista con todos los signos de puntuación que se quieran eliminar posteriormente para realizar una mejor predicción.

A continuación, se procederá a realizar la serialización de los datos, es conveniente que previamente se hayan ordenado en orden alfabético las listas de palabras y las clases, y se hará uso de archivos binarios para obtener un mejor rendimiento (.pkl).

9.2.2.2. Preparación de los datos

Una vez concluidos los pasos del punto anterior, es necesario preparar los datos para el entrenamiento, para ello se realizarán los siguientes pasos:

1. Creación de una lista vacía que guarda los datos de entrenamiento procesados.
2. Creación de una lista de ceros de la misma longitud que la lista de las palabras.
3. Creación de un bucle que recorra la lista de documentos:
    a. Creación de una bolsa de palabras “bag of words” para cada documento, donde se marca con 1 si una palabra de la lista de palabras está presente en el documento y con 0 si no lo está.
    b. Se crea una lista output_row con la misma longitud que output_empty y se marca con “1” en la posición correspondiente a la clase del documento actual.
    c. Se agrega la bolsa de palabras y la lista de salida (output_row) al conjunto de entrenamiento (training).
4. Se realiza una mezcla aleatoria y preparación final de los datos: Para evitar sesgos en el entrenamiento del modelo, se mezclan aleatoriamente los datos de entrenamiento utilizando random.shuffle. Luego, los datos de entrenamiento se convierten en un array NumPy con el tipo de datos object.
5. Se convierten las secuencias de palabras en arrays NumPy de longitud fija. Se crea un array train_x que contiene las bolsas de palabras de las entradas de entrenamiento y un array train_y que contiene las listas de salida correspondientes. Estos arrays se utilizan como entrada y salida durante el entrenamiento del modelo.

9.2.2.3. Red neuronal artificial (RNA)

A continuación, se presentará una descripción detallada de la arquitectura del modelo de red neuronal artificial implementada en este bloque del proyecto utilizando una arquitectura de red neuronal secuencial con capas densas, función de activación 'relu' en

las capas ocultas, función de activación 'softmax' en la capa de salida y Dropout para prevenir el sobreajuste. La construcción del modelo se lleva a cabo siguiendo el siguiente proceso:

1. Creación de un objeto modelo haciendo uso de Sequential() de la biblioteca Keras.
2. Creación de una capa de 128 neuronas, que tiene una entrada de la longitud de la bolsa de palabras y añadimos la función de activación ReLU para potenciar la no linealidad del modelo.
3. Se añade una capa de “Dropout” con una tasa de 0.5, para ayudar a prevenir el sobreajuste al "apagar" aleatoriamente un porcentaje de las neuronas durante el entrenamiento.
4. Se añade una segunda capa de 64 neuronas, a la que ponemos también la función de activación ReLU para potenciar la no linealidad.
5. Posteriormente, se añade otra capa de dropout, nuevamente con una tasa del 0.5.
6. La última capa densa que se añade tiene un número de neuronas igual al número de clases de salida y se añade la función de activación “softmax”, Softmax se utiliza para la clasificación multiclase y produce una distribución de probabilidad sobre las clases de salida.
7. Se compila el modelo utilizando el método compile. Se especifica la función de pérdida (loss) como categorical_crossentropy, que es comúnmente utilizada en problemas de clasificación multiclase. Como optimizador, se utiliza Adagrad con una tasa de aprendizaje de 0.01 y un decaimiento de 1e-6. Adagrad es un optimizador adaptativo que ajusta la tasa de aprendizaje para cada parámetro del modelo según su historia de actualizaciones anteriores. También se especifica que se desea calcular la métrica de precisión (accuracy) durante el entrenamiento.

9.2.2.4. Entrenamiento del modelo

Se hace uso del método fit para entrenar el modelo. Se pasan los arrays NumPy train_x y train_y como datos de entrenamiento. Se especifica un número de épocas de 200, lo que significa que el modelo verá los datos de entrenamiento completo 200 veces durante el entrenamiento. El tamaño de lote (batch_size) se establece en 5, lo que significa que se actualizarán los pesos del modelo después de cada lote de 5 ejemplos de entrenamiento. El argumento verbose=1 se utiliza para mostrar información detallada sobre el progreso del entrenamiento durante cada época. Finalmente, el modelo entrenado se guarda en un archivo HDF5.

A continuación, se procede a ilustrar los resultados del modelo de entrenamiento, y debido a que se dota de 200 épocas como se comentó anteriormente, se mostrará únicamente los resultados de principio (figura 28), mitad (figura 29) y final (figura 30):

```
Epoch 1/200
8/8 [==============================] - 0s 3ms/step - loss: 1.5730 - accuracy: 0.2564
Epoch 2/200
8/8 [==============================] - 0s 1ms/step - loss: 1.4719 - accuracy: 0.4103
Epoch 3/200
8/8 [==============================] - 0s 1ms/step - loss: 1.4392 - accuracy: 0.5385
Epoch 4/200
8/8 [==============================] - 0s 1ms/step - loss: 1.4262 - accuracy: 0.5128
Epoch 5/200
8/8 [==============================] - 0s 1ms/step - loss: 1.3649 - accuracy: 0.5128
Epoch 6/200
8/8 [==============================] - 0s 1ms/step - loss: 1.3696 - accuracy: 0.4615
Epoch 7/200
8/8 [==============================] - 0s 1ms/step - loss: 1.2726 - accuracy: 0.5128
Epoch 8/200
8/8 [==============================] - 0s 1ms/step - loss: 1.2681 - accuracy: 0.5385
Epoch 9/200
8/8 [==============================] - 0s 905us/step - loss: 1.2357 - accuracy: 0.5128
Epoch 10/200
8/8 [==============================] - 0s 1ms/step - loss: 1.2484 - accuracy: 0.5128
Epoch 11/200
8/8 [==============================] - 0s 1ms/step - loss: 1.2321 - accuracy: 0.5128
Epoch 12/200
8/8 [==============================] - 0s 1ms/step - loss: 1.2939 - accuracy: 0.5128
Epoch 13/200
8/8 [==============================] - 0s 1ms/step - loss: 1.2015 - accuracy: 0.5385
Epoch 14/200
8/8 [==============================] - 0s 1ms/step - loss: 1.1825 - accuracy: 0.5385
Epoch 15/200
8/8 [==============================] - 0s 1ms/step - loss: 1.1417 - accuracy: 0.5897
Epoch 16/200
8/8 [==============================] - 0s 714us/step - loss: 1.1509 - accuracy: 0.5641
```

Figura 28. Primeras épocas entrenamiento

```
Epoch 93/200
8/8 [==============================] - 0s 1ms/step - loss: 0.4847 - accuracy: 0.8205
Epoch 94/200
8/8 [==============================] - 0s 859us/step - loss: 0.4603 - accuracy: 0.8462
Epoch 95/200
8/8 [==============================] - 0s 1ms/step - loss: 0.3964 - accuracy: 0.8974
Epoch 96/200
8/8 [==============================] - 0s 859us/step - loss: 0.4428 - accuracy: 0.8462
Epoch 97/200
8/8 [==============================] - 0s 1ms/step - loss: 0.4655 - accuracy: 0.7949
Epoch 98/200
8/8 [==============================] - 0s 857us/step - loss: 0.4130 - accuracy: 0.8974
Epoch 99/200
8/8 [==============================] - 0s 1ms/step - loss: 0.4386 - accuracy: 0.8205
Epoch 100/200
8/8 [==============================] - 0s 857us/step - loss: 0.4795 - accuracy: 0.8462
Epoch 101/200
8/8 [==============================] - 0s 1ms/step - loss: 0.3055 - accuracy: 0.9487
Epoch 102/200
8/8 [==============================] - 0s 857us/step - loss: 0.3602 - accuracy: 0.8974
Epoch 103/200
8/8 [==============================] - 0s 1ms/step - loss: 0.4223 - accuracy: 0.8974
Epoch 104/200
8/8 [==============================] - 0s 857us/step - loss: 0.4236 - accuracy: 0.8974
Epoch 105/200
8/8 [==============================] - 0s 1000us/step - loss: 0.4335 - accuracy: 0.7949
Epoch 106/200
8/8 [==============================] - 0s 1ms/step - loss: 0.3962 - accuracy: 0.7949
Epoch 107/200
8/8 [==============================] - 0s 859us/step - loss: 0.2778 - accuracy: 0.9487
Epoch 108/200
8/8 [==============================] - 0s 1ms/step - loss: 0.3631 - accuracy: 0.8974
Epoch 109/200
8/8 [==============================] - 0s 859us/step - loss: 0.3554 - accuracy: 0.9231
```

Figura 29. Ecuador épocas entrenamiento

```
Epoch 186/200
8/8 [==============================] - 0s 1ms/step - loss: 0.2124 - accuracy: 0.9487
Epoch 187/200
8/8 [==============================] - 0s 714us/step - loss: 0.1556 - accuracy: 0.9744
Epoch 188/200
8/8 [==============================] - 0s 1ms/step - loss: 0.1140 - accuracy: 1.0000
Epoch 189/200
8/8 [==============================] - 0s 715us/step - loss: 0.2044 - accuracy: 0.9744
Epoch 190/200
8/8 [==============================] - 0s 1ms/step - loss: 0.1724 - accuracy: 0.9744
Epoch 191/200
8/8 [==============================] - 0s 858us/step - loss: 0.1687 - accuracy: 0.9487
Epoch 192/200
8/8 [==============================] - 0s 1ms/step - loss: 0.1512 - accuracy: 1.0000
Epoch 193/200
8/8 [==============================] - 0s 857us/step - loss: 0.1452 - accuracy: 0.9744
Epoch 194/200
8/8 [==============================] - 0s 1ms/step - loss: 0.1938 - accuracy: 0.9487
Epoch 195/200
8/8 [==============================] - 0s 859us/step - loss: 0.2104 - accuracy: 0.9744
Epoch 196/200
8/8 [==============================] - 0s 1ms/step - loss: 0.2297 - accuracy: 0.9487
Epoch 197/200
8/8 [==============================] - 0s 857us/step - loss: 0.2560 - accuracy: 0.9231
Epoch 198/200
8/8 [==============================] - 0s 1000us/step - loss: 0.2690 - accuracy: 0.9231
Epoch 199/200
8/8 [==============================] - 0s 1ms/step - loss: 0.1539 - accuracy: 0.9744
Epoch 200/200
8/8 [==============================] - 0s 857us/step - loss: 0.1515 - accuracy: 0.9744
Done
```

Figura 30. Época final entrenamiento 1

9.2.3. Integración de voz en el asistente virtual

Uno de los aspectos interesantes de este proyecto, es la creación de un asistente virtual al estilo Siri de iPhone, el cuál ayudará y aconsejará en la toma de decisiones cuando se detecta un micro sueño. En esta parte es donde se recopila y se procesa el audio, para posteriormente mandarlo al asistente para que realice la predicción de la respuesta en función de los patrones reconocidos y el campo asociado, para posteriormente dar una salida mediante voz.

Se debe poner en marcha el motor de pyttsx3, que será el que proporcionará la voz, ajustando la voz deseada y la velocidad a la que se retransmitirá el texto hablado. Posteriormente, se debe crear una serie de funciones obvias, como hablar, escuchar y transcribir.

Un matiz, cabe recordar que en la función de escuchar es conveniente ajustar el micrófono para eliminar el ruido ambiental de fondo, ya que esto ayudará a mejorar la calidad del audio capturado al reducir el ruido no deseado. Luego, el audio capturado hay que convertirlo a datos, ya según el formato que más guste, en este caso se ha utilizado formato .FLAC, ya que es más rápido de procesar que .WAV y .MP3 (decir que, depende del contexto, ya que .FLAC es un formato de compresión sin pérdida, por lo que a veces puede ser pesado dependiendo del archivo) en nuestro caso.

Una vez que se finaliza la función de escuchar, se lleva a cabo la función de la transcripción del audio recogido, en este proyecto se ha hecho uso de Whisper AI debido a su gran capacidad de transcribir con alta efectividad y debido a los numerosos modelos que nos ofrece. Es cierto que para poder tener una transcripción perfecta y rápida se necesita una computadora muy potente **(ver apartado 9.2.1.10),** en este proyecto se han obtenido las siguientes conclusiones de los modelos:

- **Tiny** → Se descartó debido a su pésimo nivel de transcripción
- **Base** → Adecuado, aunque a veces falla.
- **Small** → Adecuado, aunque a veces falla.
- **Medium** → Tarda bastante tiempo en transcribir, pero tiene una alta efectividad.
- **Large** → Impensable para un equipo de gama media.

Con respecto a los modelos más adecuados para este proyecto, por la rapidez y eficacia que brindan en la mayoría de las simulaciones se han realizado una serie de

comprobaciones (ver Tabla 3). Para ello se ha llevado a cabo la temporización de lo que tardan en procesar una frase los modelos “base” y “small”, la frase ha sido: “Me encuentro bastante cansado, tengo mucho sueño.”.

	Equipo gama media - baja	**Equipo gama media - alta**
SMALL	39.380 segundos	14.862 segundos
	39.776 segundos	15.597 segundos
	39.676 segundos	15.433 segundos
	40.876 segundos	15.005 segundos
BASE	14.862 segundos	6.408 segundos
	15.597 segundos	6.206 segundos
	15.433 segundos	6.310 segundos
	15.005 segundos	6.286 segundos

Tabla 3. Comparación modelos Whisper

Como se puede observar en la tabla anterior, con el modelo “BASE” tenemos más rapidez, de modo que será la elección correcta ya que en escenarios donde la toma de decisiones rápida puede ser crucial para salvaguardar la seguridad y el bienestar. Es fundamental considerar esta característica distintiva puesto que puede ser de vital importancia.

9.2.4. Funcionamiento

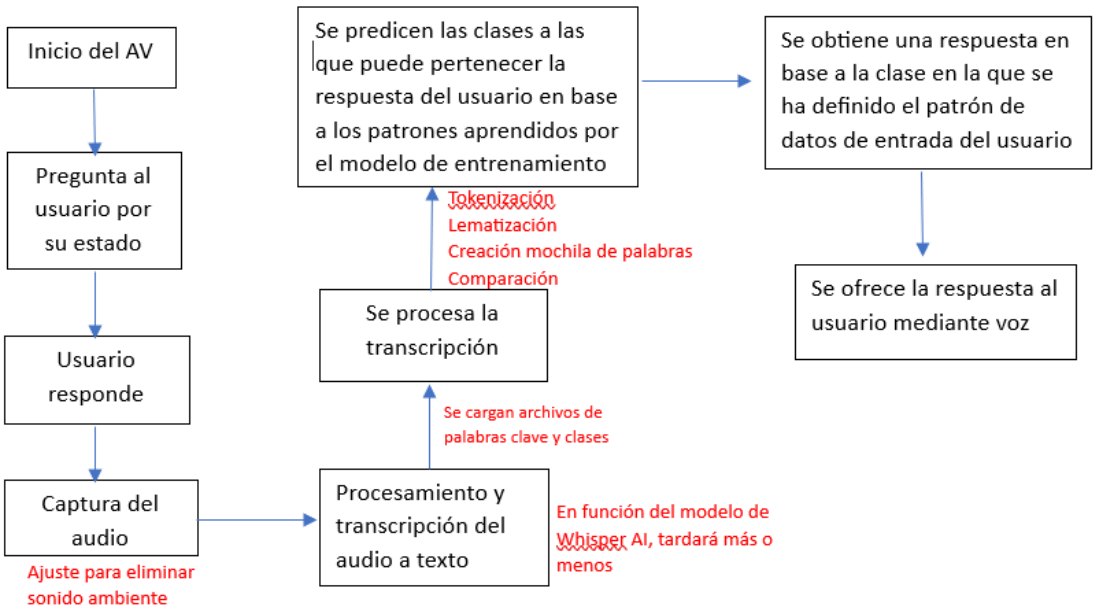

Figura 31. Diagrama de bloques AV. 1

Una vez que ya están todas las piezas completadas, solo queda unirlas. El mecanismo de funcionamiento cuando entra esta herramienta en juego es el siguiente:

1. Se hace uso de la función “talk()” para preguntar al conductor como se encuentra debido a la detección de un micro sueño.
2. Seguidamente, se hace uso de la función “listen()” creada anteriormente, que recogerá el testimonio durante unos segundos de la persona al volante. Y se pasará ese audio recogido a datos para procesarlo posteriormente.
3. Se procesan los datos y se realiza su correspondiente predicción para tratar de detectar determinados patrones en esos datos y asociarlos con una determinada clase, el funcionamiento interno sería:
    a. Lo primero que se realiza es la tokenización de la oración en palabras individuales usando la función “nltk.word_tokenize()”, y luego lematiza cada palabra utilizando “lemmatizer.lemmatize()”. Posteriormente, se devuelve una lista de palabras procesadas.
    b. Haciendo uso de la lista de palabras procesadas anteriormente se crea un vector de características (bag of words) de longitud igual a la cantidad de palabras en el conjunto de palabras words. Inicialmente, todos los valores

del vector se establecen en cero. Luego, para cada palabra en la lista de palabras procesadas, busca la posición de esa palabra en el conjunto de palabras words y establece el valor correspondiente en el vector de características en 1 si la palabra está presente en words. Finalmente, se devuelve el vector de características como un array numpy con 1 (si algunas palabras del array se han encontrado en la bolsa de palabras) o con 0 (en caso de que resulte no haber coincidencias).

c. Posteriormente se hace uso de la función que predice a que clase se asocia la respuesta del usuario, para ello, lo primero que se hace es coger el vector que se ha obtenido en la función anterior, luego, se establece un modelo de predicción para predecir las diferentes probabilidades de pertenecer a una clase basándose en los patrones del archivo JSON, es importante establecer un umbral, por ejemplo, en 0.25. Lo que significa que solo se considerarán las probabilidades que superen ese umbral. Luego, las probabilidades y las clases correspondientes se almacenan en una lista llamada “results”, que se ordena en orden descendente de probabilidades. Finalmente, se construye una lista de diccionarios con las intenciones y probabilidades correspondientes y se devuelve esa lista.

d. Después, se hace uso de una función “get_response(ints, intents_json)” que toma la lista de diccionarios de intenciones y probabilidades (ints) y el archivo JSON de intenciones (intents_json). Se recorren las intenciones en intents_json y busca la intención que coincide con la intención más probable en ints. Luego, selecciona una respuesta aleatoria de las respuestas asociadas a esa intención y la devuelve como resultado.

4. Finalmente, una vez que se ha obtenido la respuesta procesada, ésta se envía al motor de voz para ser reproducida de manera verbal.

9.3. Bloque 3. Envío de mensajes a la nube

En los siguientes subapartados se va a proceder a explicar las funcionalidades que se ofrecen en el tercer bloque. Primero se comenzará conociendo la plataforma de servicio usada para alojar en la nube los mensajes que se manda con ciertos datos de relevancia, posteriormente se comentarán las bibliotecas necesarias para la realización y correcta ejecución del bloque tres, asi como su instalación. Además, se hablará del protocolo utilizado para el envío de mensajes a la nube y por último se acabará con un apartado específico para entender con mayor profundidad el funcionamiento de este bloque.

9.3.1. Plataforma de servicio. Firebase

Finalmente, tras realizar pruebas en diversos entornos, la plataforma que más se ha ajustado a las necesidades del proyecto ha sido Firebase, Firebase es una plataforma en la nube de la cual es propietaria la tecnológica Google, dispone de varias secciones muy útiles dentro de su plataforma, en este caso, para la realización de este proyecto se ha hecho uso del servicio de base de datos en tiempo real que ofrece, siendo ésta capaz de almacenar y sincronizar datos con nuestra base de datos NoSQL alojada en la nube. Los datos se sincronizan con todos los clientes conectados en tiempo real, Los datos se almacenan en formato JSON. Es necesario descargar el fichero de credenciales que nos ofrece Firebase, puesto que después a la hora de realizar la petición será útil.

En la siguiente figura, podemos ver un ejemplo de cómo se almacenan los datos en dicha plataforma.

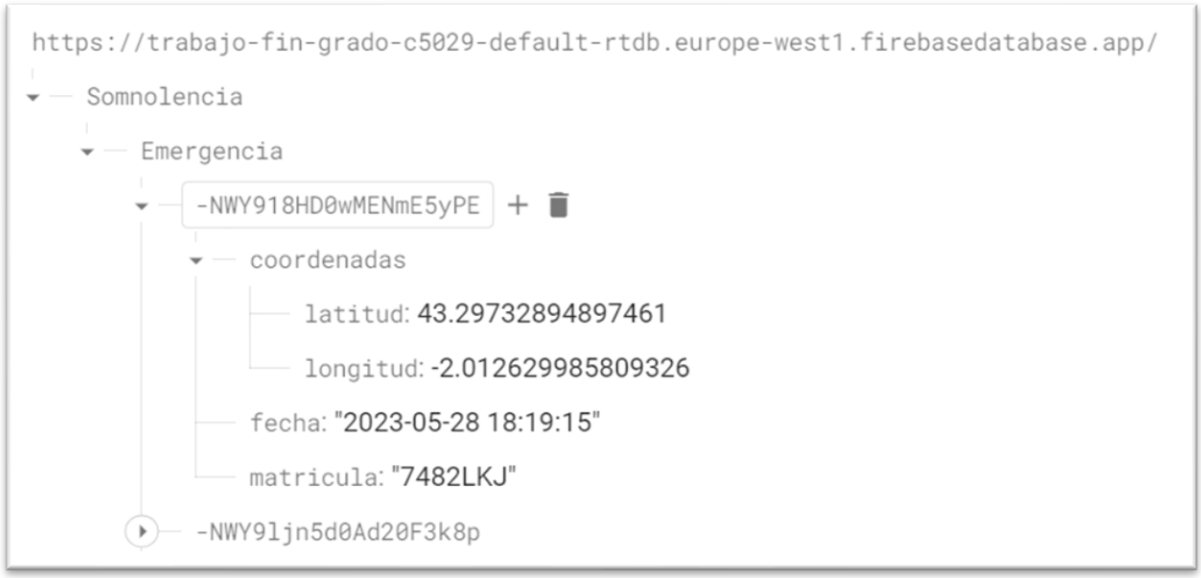

Figura 32. Realtime Database Firebase

9.3.2. Bibliotecas

En los siguientes subapartados se explicaran las bibliotecas necesarias para la realización de dicho bloque, una de ellas **Numpy** se puede **ver en el apartado 9.1.1.3**, por tanto, no aparecerá reflejada aquí pero es necesaria, por ende, al haber sido vista en el primer bloque, se entiende que ya se encuentra instalada, por lo que simplemente tendríamos que importarla nada más.

9.3.2.1. Requests

La biblioteca "requests" en Python es una herramienta que permite realizar solicitudes HTTP de manera sencilla y eficiente. Algunas de las funcionalidades clave de la biblioteca "requests" incluyen: [34]

- Realizar solicitudes HTTP: envío de solicitudes GET, POST, PUT, DELETE.
- Manejo de encabezados y cookies: establecer y gestionar encabezados personalizados en las solicitudes, así como administrar cookies.
- Pasar parámetros: adjuntar parámetros a las solicitudes, lo que es útil para enviar datos a través de la URL en solicitudes GET o enviar datos de formulario en solicitudes POST.
- Gestionar respuestas: acceder a la respuesta recibida, incluyendo el contenido, el código de estado, los encabezados y otros detalles relevantes.

En este proyecto es utilizada para enviar el mensaje de datos estructurado a la nube a través del protocolo HTTP. Para poder utilizarla es necesaria su instalación, por tanto, habría que ejecutar el comando que podemos observar en la siguiente figura (figura 33)

```
C:\Windows\system32>pip install requests
```

Figura 33. Instalación Requests

9.3.2.2. API IpStack

En este bloque del proyecto se hace uso de la API IpStack, con el objetivo de obtener las coordenadas geográficas del dispositivo en tiempo real, basándose en la dirección de Internet Protocol (IP) del dispositivo, la API brinda tanto la longitud como la latitud para que posteriormente estas puedan ser enviadas a la nube, como medio de información para la localización del vehículo con necesidad de asistencia.

Para su correcto funcionamiento es necesario tener una API KEY con la que podremos acceder al recurso en la nube para obtener las coordenadas geográficas, basándose en la dirección IP del dispositivo. La forma de realizar el proceso, se puede apreciar en la siguiente figura.

```python
def geolocation():
    localizacion=[]
    # Obtiene tu dirección IP actual
    ip_url = 'https://api.ipify.org?format=json'
    ip_response = requests.get(ip_url)
    ip_address = ip_response.json()['ip']

    # API key para ipstack (reemplaza con tu propia clave de API)
    api_key = '05da2c81113e9befba021f553e31263b'

    # URL para la consulta de geolocalización
    geo_url = f'http://api.ipstack.com/{ip_address}?access_key={api_key}'

    # Realiza la solicitud a la API de geolocalización
    response = requests.get(geo_url)
    data = response.json()

    # Obtiene la latitud y longitud de la ubicación
    latitud = data['latitude']
    longitud = data['longitude']
```

Figura 34. Obtener geolocalización

9.3.2.3. Datetime

Similar a la biblioteca Time, proporciona clases y funciones para trabajar con fechas, horas y combinaciones de ambas. Es una biblioteca estándar de Python que ofrece una amplia gama de funcionalidades relacionadas con el manejo de fechas y tiempos.

9.3.2.4. Firebase_admin

Es una biblioteca proporcionada por Firebase que facilita la integración y el uso de servicios de Firebase en aplicaciones Python. Esta biblioteca permite interactuar con diversas funcionalidades y servicios de Firebase, como la autenticación de usuarios, el almacenamiento en la nube, la base de datos en tiempo real y la mensajería en la nube, entre otros.

Para utilizar esta biblioteca es necesaria la instalación manual de la misma, para ello se debe ejecutar el comando que aparece en la siguiente figura (figura 35).

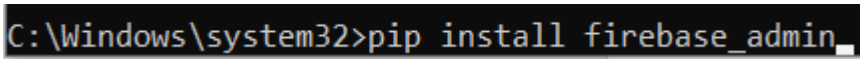

Figura 35. Instalar firebase_admin

9.3.3. Protocolo de envío de mensajes

Para realizar el envío de mensajes a la base de datos en tiempo real de Firebase, se puede hacer uso de dos protocolos, FCM (Firebase Cloud Messaging) o HTTP, en este caso, se ha hecho uso del protocolo HTTP, concretamente del método push() para realizar el envío.

El protocolo HTTP es ampliamente utilizado en aplicaciones web y servicios en línea para la comunicación entre clientes y servidores. Permite la transferencia de datos estructurados, como el cuerpo de la solicitud y las respuestas, utilizando diferentes métodos, como POST, GET, PUT, DELETE.[35]

HTTP se fundamenta en simples intercambios de solicitud y respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. A su vez, el servidor responde con un mensaje similar que incluye el estado de la operación y el posible resultado. Cada una de estas operaciones puede involucrar un objeto o recurso al que se aplican. [35]

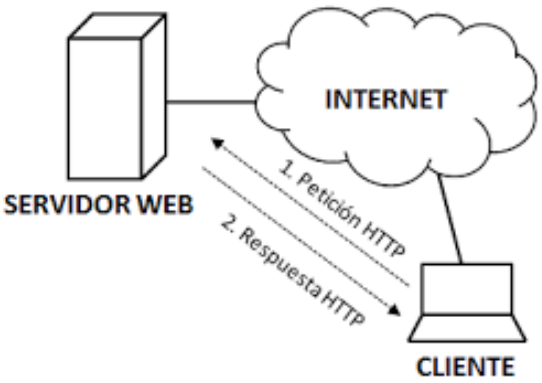

Figura 36. Protocolo HTTP

Fuente: https://acortar.link/lLupAi

9.3.4. Funcionamiento

Este bloque entra en funcionamiento cuando la persona no ha respondido a la alarma de emergencia y por consiguiente se pasa a un estado de riesgo absoluto, donde cabe la posibilidad de que el conductor haya podido sufrir un desvanecimiento o llegar a un estado de somnolencia profundo.

El primer paso, en un entorno ideal, donde la conducción autónoma de vehículos estuviera asentada, sería activar dicho modo con objeto de salvaguardar la seguridad vial poniendo fuera de peligro el vehículo, ya sea realizando una parada de emergencia en el arcén o tratando de llegar al área de descanso más cercana. El siguiente paso, sería la activación del aviso de emergencia, éste consiste en enviar un mensaje a la nube, concretamente a una base de datos en tiempo real gestionada por el 112, la idea de funcionamiento sería la siguiente:

1. En primer lugar, se obtiene la geolocalización del vehículo basada en la dirección IP del dispositivo. Por tanto, lo primero sería obtener la dirección IP del dispositivo, haciendo uso de la API Ipify, una vez que se ha obtenido la dirección IP, obtenemos la geolocalización a través de una solicitud HTTP a la API de IpStack, nos devolverá los datos de geolocalización, longitud y latitud en formato JSON.
2. En segundo lugar, se establece el montaje de los datos, para ello, se obtiene tanto la fecha actual como la matrícula del vehículo y estos datos son montados junto con los de geolocalización en una estructura definida en JSON, que posteriormente usaremos para mandar.

3. Posteriormente, conectamos con la base de datos en tiempo real de Firebase y una vez que se dispone de la conexión como cliente a la base de datos, se realiza la solicitud de envío de datos a través de HTTP mediante el método push().
4. Por último, los datos quedan registrados de manera instantánea en la base de datos y los servicios de emergencia pueden actuar de forma inmediata personándose en lugar puesto que disponen de la geolocalización del vehículo.

Para tener una visión más tangible del funcionamiento se presenta el siguiente flujograma con objeto de clarificar, el funcionamiento del bloque.

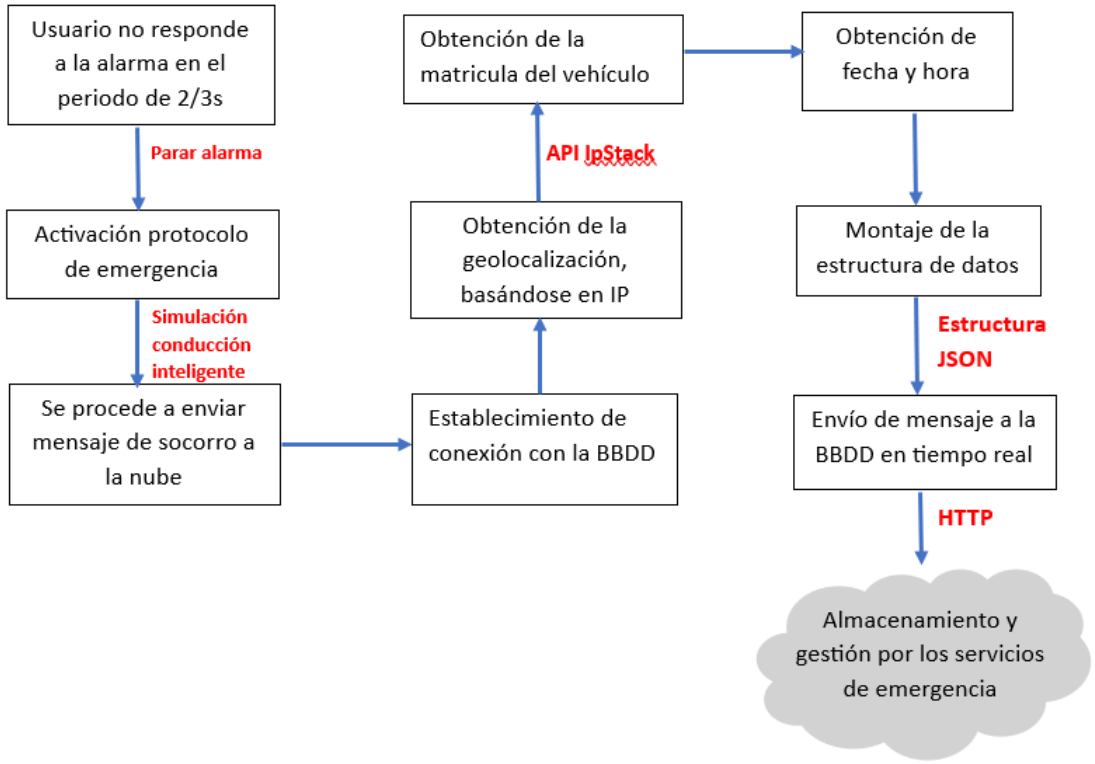

Figura 37. Diagrama de bloques del protocolo de alto riesgo

10. Prototipo físico

En los siguientes subapartados se va a proceder a detallar, los elementos necesarios para la creación de un prototipo de detección de somnolencia real. Como se puede ver en la siguiente imagen, el resultado final sería el siguiente.

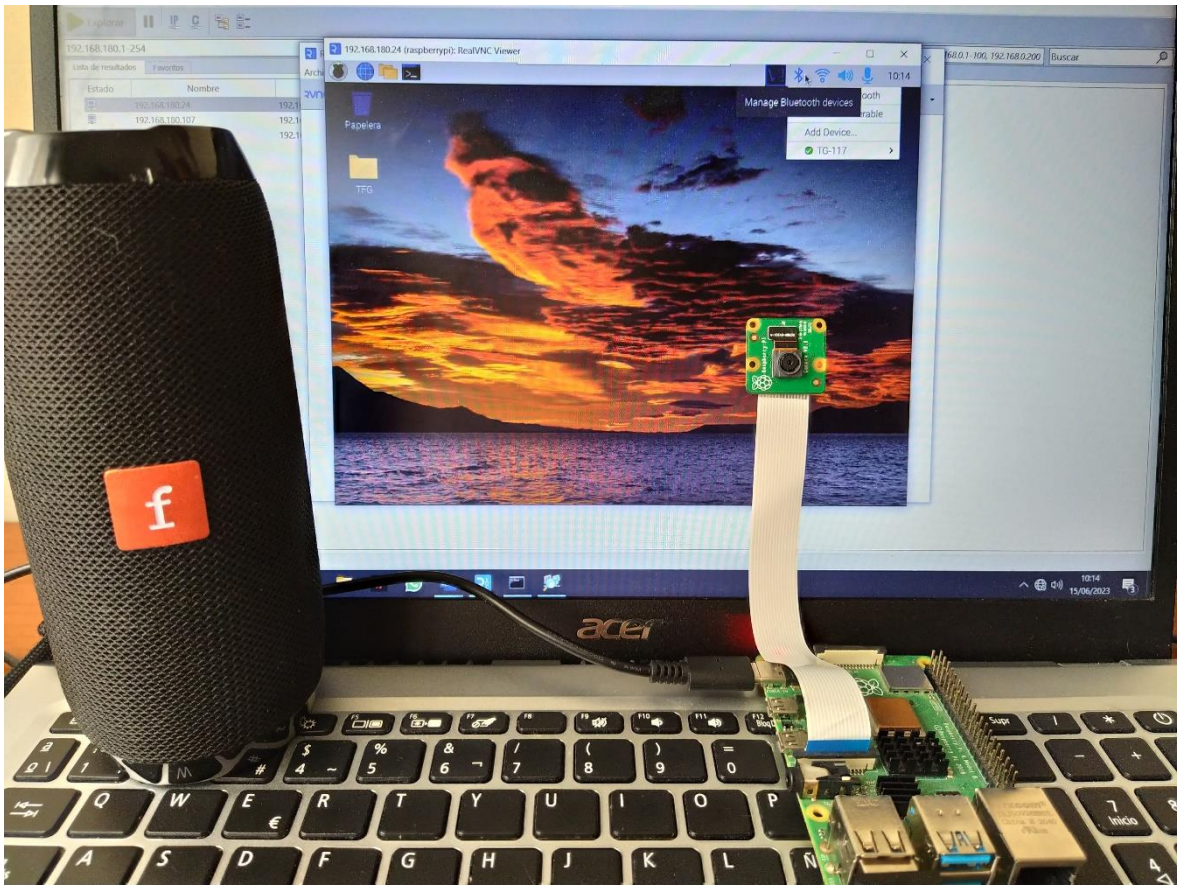

Figura 38. Prototipo físico

10.1. Dispositivos y accesorios

10.1.1. Raspberry PI

Una Raspberry Pi es una computadora de placa única (SBC, por sus siglas en inglés) de bajo costo y tamaño compacto y altamente versátil. Está diseñada para promover la enseñanza de la informática y la programación, así como para permitir la creación de una amplia variedad de proyectos electrónicos. Aunque es pequeña, ofrece un rendimiento suficiente para realizar diversas tareas y ejecutar diferentes sistemas operativos basados en Linux.

La Raspberry Pi que se ha escogido para la realización de este proyecto se corresponde con el último modelo de 2018, es decir, la Raspberry Pi 4 Model B. Como se puede observar en la figura inferior, cuenta con numerosos elementos integrados que pueden ser realmente útiles para el desarrollo del proyecto.

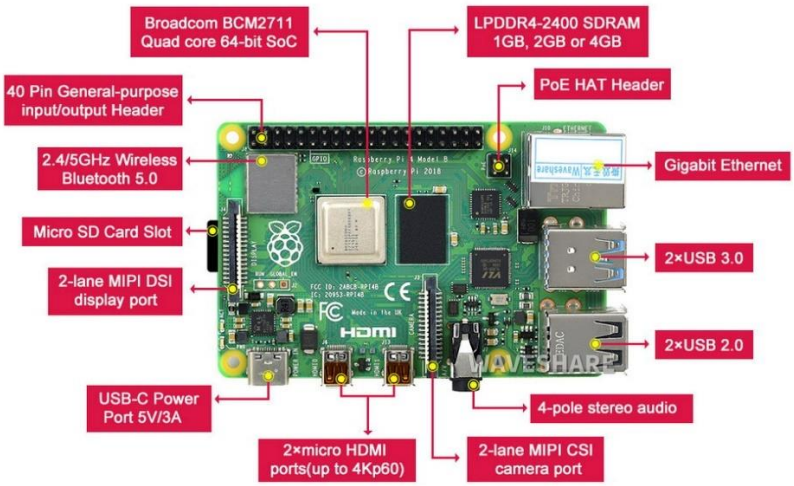

Figura 39. Partes de la Raspberry Pi 4B

Fuente: http://msrobotics.net/index.php/clases-de-raspberry-pi/247-raspberry-pi-4

10.1.2. Cámara Pi

Es una cámara diseñada específicamente para ser utilizada con una placa Raspberry Pi. Estas cámaras se conectan directamente a los puertos de la placa y permiten capturar imágenes y videos de alta calidad. La cámara Raspberry Pi se controla mediante software y se puede programar para realizar diversas funciones y operaciones. Se utiliza

en combinación con la potencia de procesamiento de la Raspberry Pi para crear soluciones personalizadas de visión por computadora y proyectos relacionados con la imagen.

En este proyecto es de vital importancia ya que será la encargada de monitorear el parpadeo de los ojos y de capturar los frames que determinaran si se encuentran cerrados o abiertos.

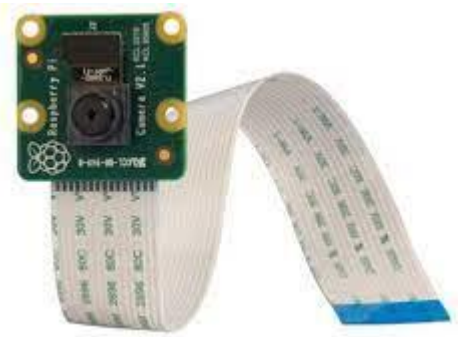

Figura 40. Cámara nativa de Raspberry

Fuente: https://tienda.bricogeek.com/accesorios-raspberry-pi/822-camara-raspberry-pi-v2-8-megapixels.html

10.1.3. Tarjeta SD

La tarjeta SD en la Raspberry Pi tiene un papel fundamental, ya que se utiliza como medio de almacenamiento principal para el sistema operativo y los datos en la placa. La Raspberry Pi no cuenta con un almacenamiento interno incorporado como una unidad de disco duro. En su lugar, se utiliza una tarjeta SD (Secure Digital) o una tarjeta microSD (en modelos más recientes) para alojar el sistema operativo y los archivos.

La tarjeta SD alberga el sistema operativo de la Raspberry Pi, que generalmente es una distribución de Linux diseñada específicamente para esta plataforma, como Raspbian (ahora conocido como Raspberry Pi OS). El sistema operativo se instala en la tarjeta SD y se inicia desde allí cuando se enciende la Raspberry Pi.

Además del sistema operativo, también se pueden almacenar programas, archivos y datos en la tarjeta SD. Esto permite que la Raspberry Pi ejecute aplicaciones, almacene archivos de configuración y acceda a datos de manera similar a una computadora convencional.

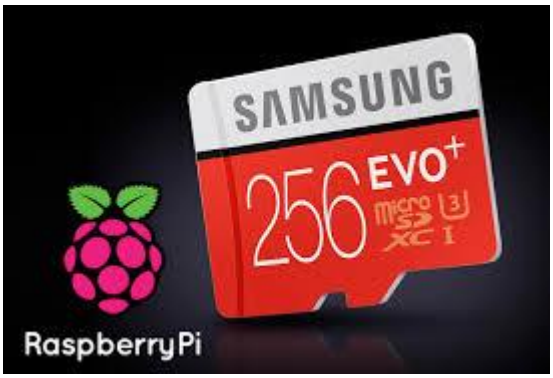

Figura 41. Tarjeta SD para Raspberry

Fuente: https://tarjetasdememoria.online/tarjetas-micro-sd-raspberry-pi/

10.1.4. Adaptador de corriente

Un adaptador de corriente, también conocido como fuente de alimentación o cargador, es un dispositivo utilizado para proporcionar energía eléctrica a otros dispositivos. Su función principal es transformar la corriente eléctrica de una fuente de alimentación, como una toma de corriente de pared, en una corriente y voltaje adecuados para el dispositivo que se va a alimentar.

Figura 42. Adaptador de corriente para Raspberry

Fuente: https://raspberryparanovatos.com/tienda/adaptadores-de-corriente.html

10.1.5. Micrófono USB

Es un dispositivo externo que se conecta a la Raspberry Pi a través de un puerto USB y se utiliza para capturar audio, son una gran opción en relación calidad precio y debido a sus grandes capacidades para capturar audio. De modo que, se encargará de capturar las respuestas que el conductor le proporcione de la manera más limpia posible.

Figura 43. Micrófono USB

Fuente: https://www.amazon.es/SunFounder-Microphone-Raspberry-Recognition-Software/dp/B01KLRBHGM

10.1.6. Altavoz

Un altavoz es un dispositivo que convierte señales eléctricas en ondas sonoras audibles, permitiendo la reproducción de sonido en diferentes dispositivos y entornos.

En este proyecto será usado para reproducir el sonido de la alarma, que emitirá una señal acústica aguda con objeto de que el sujeto se despierte del micro sueño y también servirá para reproducir la voz del asistente virtual. En este caso se ha hecho uso de un altavoz que disponía de Bluetooth, por lo cual se ha conectado a la Raspberry por medio de dicha tecnología permitiéndonos realizar el proceso de emisión acústica vía inalámbrica.

Figura 44. Altavoz Bluetooth

Fuente: https://www.amazon.es/JBL-Inal%C3%A1mbrico-Resistente-Partyboost-Reproducci%C3%B3n/dp/B08VDNCZT9

10.1.7. Punto de acceso

En este proyecto es necesaria la intervención de un punto de acceso, el proyecto se engloba dentro del campo de Internet de las Cosas puesto que el sistema se encuentra conectado a Internet. Luego, es necesario para que se pueda llevar a cabo la ejecución del bloque tres, que consiste en el envío de mensajes a una base de datos en tiempo real cuando se entra en una situación de extremo peligro. Por ello, la conexión deberá ser estable.

En este caso se ha usado como punto de acceso un smartphone y un repetidor WiFi. Primeramente, se conectó a un router para que éste le asignara una dirección IP por DHCP.

10.2. Inserción software en Raspberry

Como bien se ha comentado anteriormente en el apartado 10.1.1, la Raspberry es una mini computadora, capaz de realizar procesos igual que un ordenador, por tanto, dispone de una interfaz gráfica y de diversos procesos que se pueden realizar en un ordenador convencional como por ejemplo acceder a internet a través de un navegador, en el caso de este proyecto al ser para uso personal, se ha descargado en Raspberry a través de Google Drive, pero en caso de quedar el proyecto a nivel público, y estar este en GitHub, por ejemplo. Se podría acceder sin problemas desde la interfaz gráfica.

Una vez que está el proyecto descargado, es necesario crear un entorno virtual para realizar la ejecución del proyecto, esto se debe a la necesidad de instalar todas las bibliotecas mencionadas anteriormente en el proyecto. Los pasos a realizar son los siguientes: [36]

1. Creación de una carpeta donde se establecerá el entorno virtual.
2. Acceder a través de la línea de comandos a la carpeta creada.
3. Ejecutar los siguientes pasos, para la creación del entorno virtual:
    a. Instalar virtualenv, mediante el comando: **pip install virtualenv**
    b. Creación del entorno virtual haciendo uso del siguiente comando: **python3 -m virtualenv nombre_de_tu_entorno**
    c. Para activarlo, se debe acceder a la carpeta que se ha creado, que será el entorno virtual y ejecutar el siguiente comando: **source bin/actívate**

d. Posteriormente es recomendable crear una carpeta llamada src, dentro del entorno virtual, y allí almacenar todo el proyecto.
e. Proceder a la instalación de todas las librerías necesarias, dentro de entorno virtual, teniendo el mismo activado.
f. Para desactivar el entorno sería ejecutar el comando: **deactivate**

Una vez que ya está el entorno virtual activado y todas las dependencias instaladas, se puede ejecutar el proyecto desde la ventana de comandos como se muestra en la siguiente figura (figura 45).

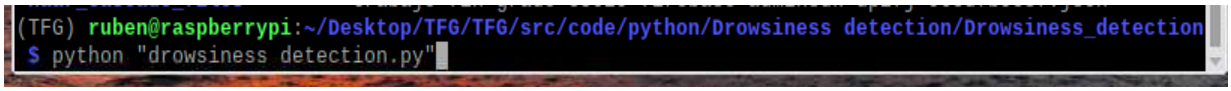

Figura 45. Comando ejecución del proyecto en Raspberry

11. Resultados finales

Como resultado final del proyecto se ha obtenido un prototipo software y un prototipo físico haciendo uso de una Raspberry PI 4B. En ambos entornos el funcionamiento es idéntico, aunque cabe destacar que en el ordenador portátil, el procesamiento de las imágenes y la fluidez del sistema de detección de somnolencia es superlativamente superior, esto se debe a que Raspberry Pi solamente cuenta con 4GB de memoria RAM, mientras que el ordenador donde se han realizado las pruebas cuenta con 8GB, además Raspberry Pi 4B dispone de un único procesador de cuatro núcleos a 1.5GHz, mientras que el ordenador cuenta con 8 procesadores de 4 núcleos a 2.8 GHz.

En las siguientes imágenes se podrá observar la puesta en marcha de ambos entornos, por un lado, el código software ejecutándose en el equipo:

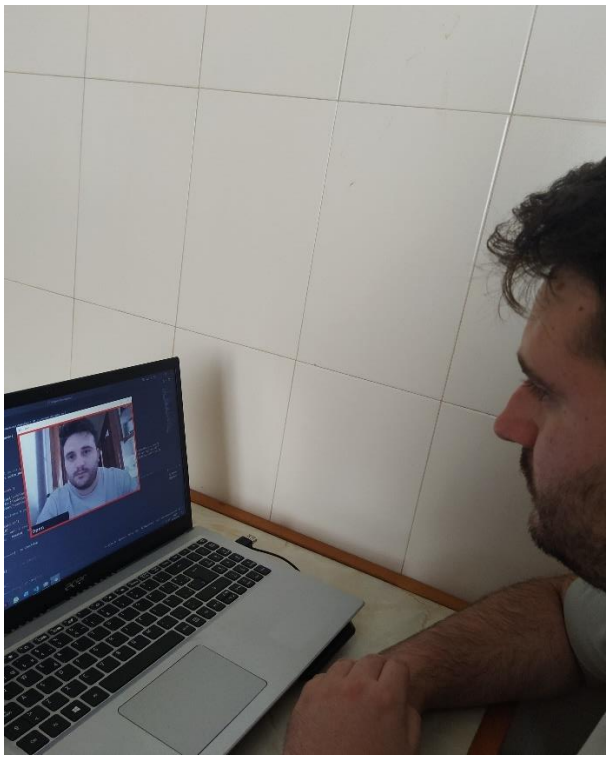

Figura 46. Compilación software en equipo.

Por otro lado, el prototipo físico haciendo uso de la interfaz gráfica de la conexión remota:

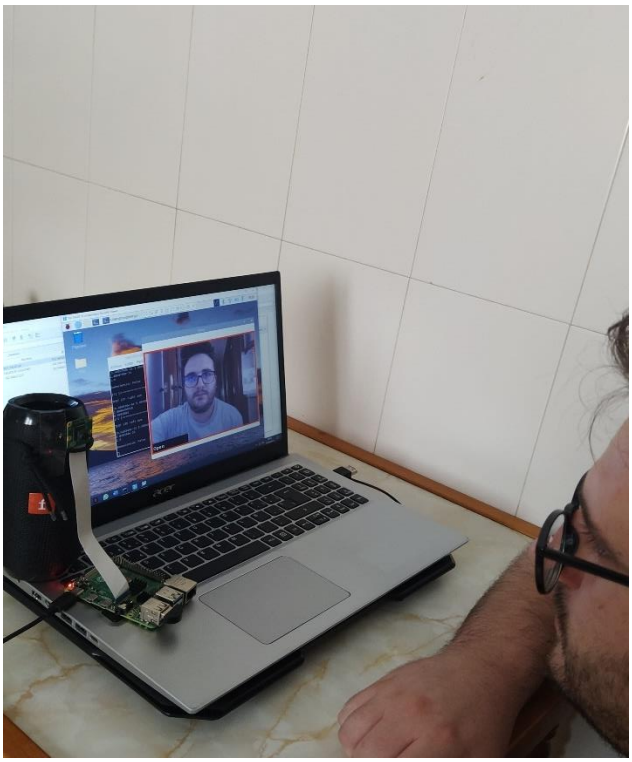

Figura 47. Prototipo físico en funcionamiento

A continuación, se mostrarán una serie de imágenes en las cuales se podrá observar lo que podría ser el funcionamiento lógico del programa. Se diferenciará entre:

1. Estado de los ojos, tanto abiertos como cerrados.
2. Detección de somnolencia.
3. Envío de mensajes a la nube.

En primer lugar, se muestra como el programa es capaz de determinar el estado de los ojos, después de realizar el procesamiento de la imagen capturada, y procesar y analizar cada ojo por separado mediante la predicción de la red neuronal convolucional, el resultado se puede apreciar en las siguientes imágenes.

Ojos abiertos

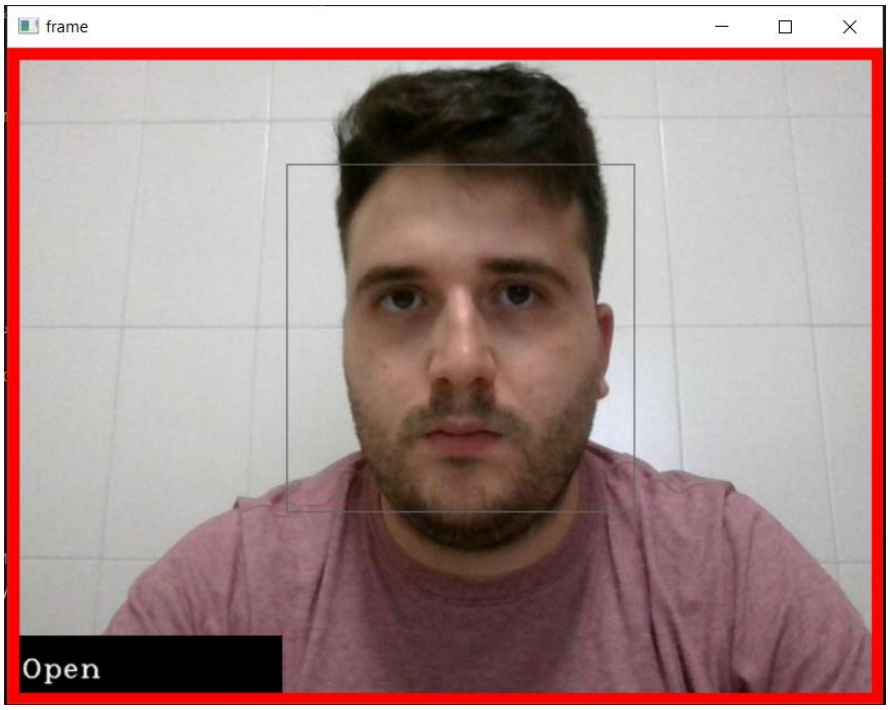

Figura 48. Detección ojos abiertos

Ojos cerrados

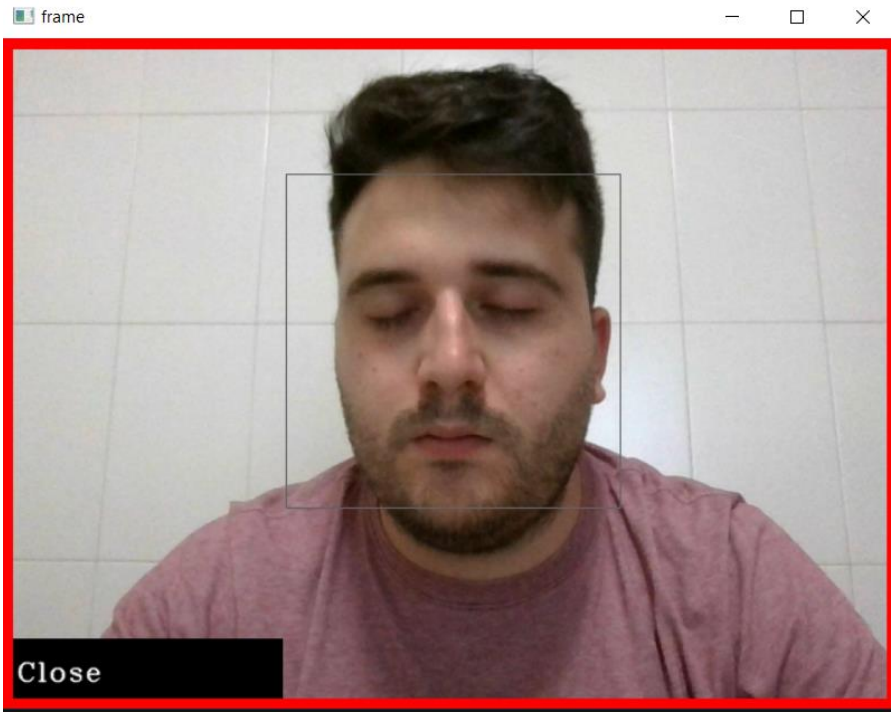

Figura 49. Detección ojos cerrados

A continuación, se mostrará el proceso cuando el usuario experimenta somnolencia, pero este reacciona y se activa el asistente virtual, que le preguntará por su estado anímico

en ese momento, el asistente procesará la respuesta del usuario y obtendrá una respuesta acorde a la frase que el usuario le ha indicado, además esa frase, será una frase tipo que se encuentra dentro de la familia en la que se ha acotado la entrada de datos acorde al patrón del usuario, de modo que podemos observar el resultado en las siguientes imágenes.

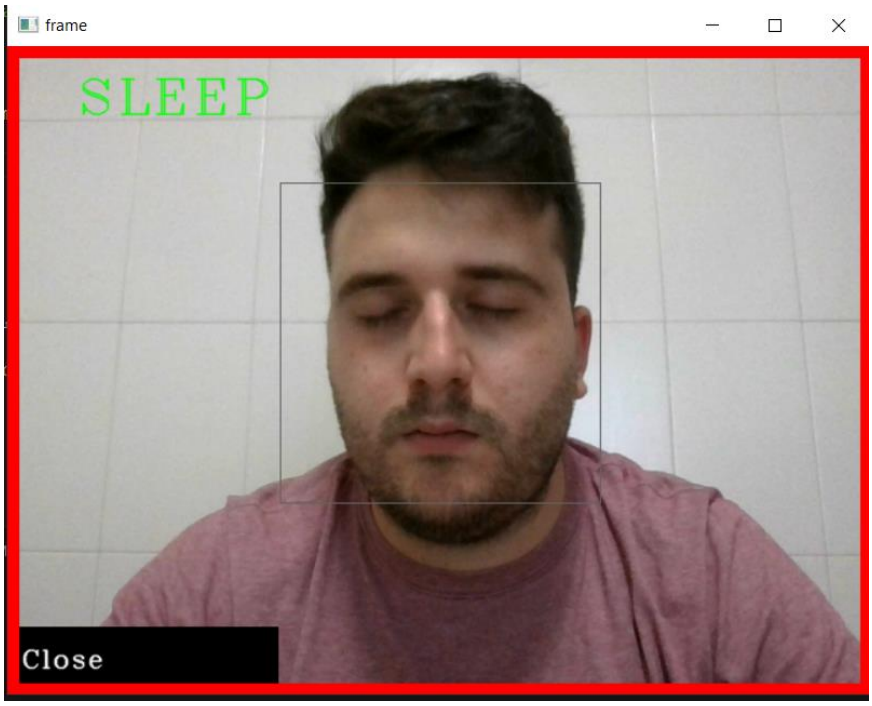

Figura 50. Detección de somnolencia

En este punto la alarma se encontraría sonando a la espera de que el usuario sea capaz de reaccionar en dos segundos, una vez que reacciona se produciría el proceso comentado en el párrafo anterior y el resultado sería el siguiente.

```
Di algo...

mensaje:  me encuentro bastante cansado tengo mucho sueño

1/1 [==============================] - 0s 15ms/step
Intent: somnolencia, Probability: 0.9936879

tag: somnolencia

result: Le recomiendo que haga una parada en un lugar seguro y descanse durante unos minutos para recuperarse
```

Figura 51. Entrada de datos y respuesta del asistente virtual.

Con respecto al paso anterior podemos ver como se ha detectado somnolencia, justo a los dos segundos y ha comenzado el protocolo de alarma. Para ello observamos la siguiente imagen tomada en el momento exacto que se detecta el micro sueño.

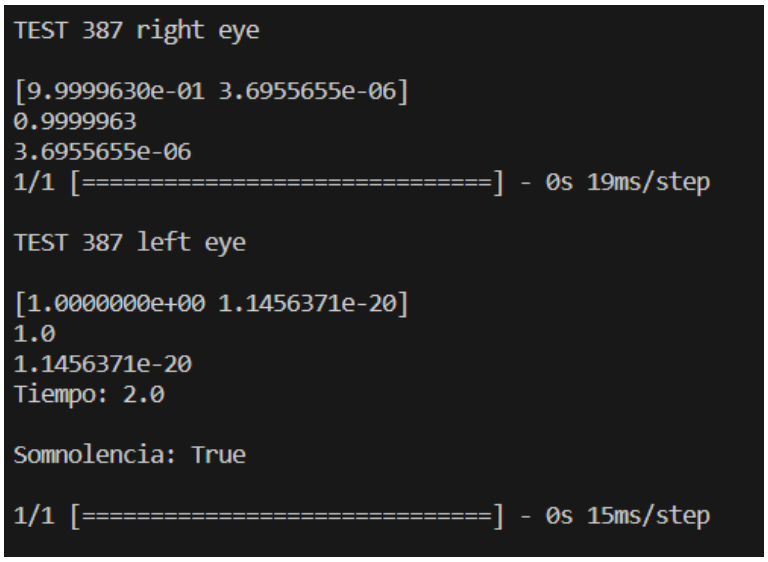

Figura 52. Momento exacto de la detección de somnolencia

Por último, en caso de que el usuario no responda al aviso de la alarma, se procede al inicio del protocolo de peligro máximo, por el cuál se activaría la conducción inteligente, con objeto de poner a salvo al usuario y se enviaría un mensaje a la nube con la matrícula del vehículo, las coordenadas geográficas, fecha y hora del suceso.

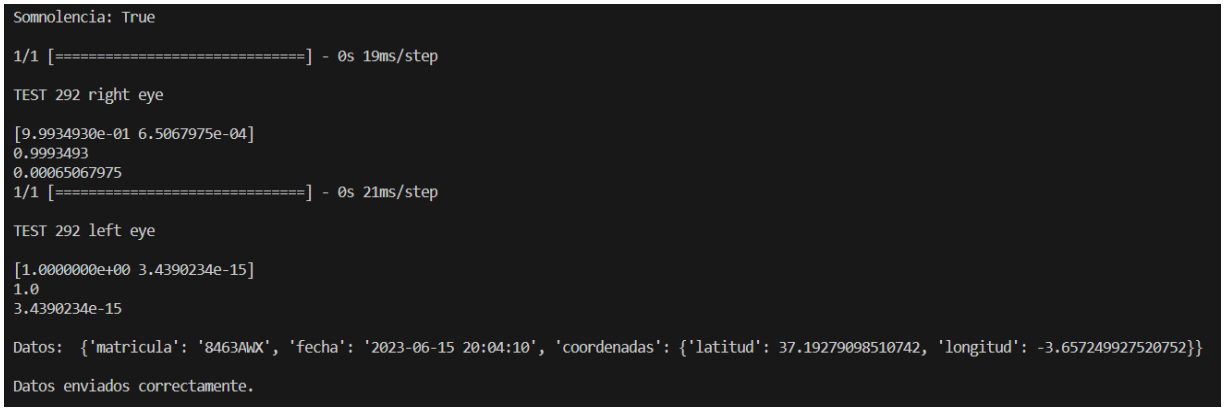

Figura 53. Protocolo de alto riesgo

Una vez enviados los datos, los servicios de emergencia dispondrían de los datos que han sido puestos en su base de datos, que se trata de una base de datos NoSQL que controla información en tiempo real. Los datos quedarían expuestos en la plataforma de la siguiente manera.

Figura 54. Exposición de los datos enviados en Firebase

12. Estudio económico

En este apartado se detallará el coste de la realización del proyecto, tanto de los materiales implementados como de la mano de obra de realización.

12.1. Materiales

Se realizará una descripción de los materiales implementados y su coste, dicha descripción aparecerá reflejada en la siguiente tabla:

Cantidad	Ud.	Descripción	Precio	Subtotal
1	U	Raspberry Pi 4 Model B	85.70 €	85.70 €
1	U	Micrófono USB	7.99 €	7.99 €
1	U	Altavoz Bluetooth Speaker TG117	12,00 €	12,00 €
1	U	Adaptador MICRO-HDMI tipo D a HDMI	3.95 €	3.95 €
1	U	Adaptador de corriente para Raspberry Pi	11.14 €	11.14 €
1	U	Cámara Raspberry Pi v2.1	48.33 €	48.33 €
1	U	Tarjeta SD 32 GB Samsung EVO+	24.99 €	24.99 €
			Total	**194.10 €**

Tabla 4. Presupuesto de materiales

12.2. Mano de obra

En este apartado se detallarán los costes relacionados con la mano de obra realizada por el alumno, entrarán elementos como la formación en las diferentes áreas del proyecto, realización de los diferentes bloques, pruebas realizadas, etcétera. De modo que, el desglose de la mano de obra llevada a cabo en el proyecto se puede ver representada en la siguiente tabla (tabla 5).

Cantidad	Ud	Descripción	Precio	Subtotal
20	h	FORMACIÓN EN PYTHON	25,00 €	500,00 €
10	h	ESTUDIO DE BIBLIOTECAS	25,00 €	250,00 €
10	h	ANALISIS DEL PROBLEMA DE SOMNOLENCIA	25,00 €	250,00 €
30	h	FORMACIÓN EN IA, MACHINE LEARNING Y DEEP LEARNING	25,00 €	750,00 €
4	h	ESTUDIO PREVIO DE THINGSPEAK	25,00 €	100,00 €
80	h	DISEÑO Y DESARROLLO DE UN PROTOTIPO SOFTWARE	25,00 €	2000,00 €
10	h	PRUEBAS DE SOFTWARE	25,00 €	250,00 €
50	h	ADAPTACIÓN PROTOTIPO A RASPBERRY PI	25,00 €	1.250,00 €
70	h	ELABORACIÓN DE LA MEMORIA	25,00 €	1.750,00 €
6	h	OTROS **(*)**	25,00 €	150,00 €
			TOTAL	7.250,00 €

Tabla 5. Presupuesto de mano de obra

(*) OTROS; se refiere al tiempo empleado en la instalación de dependencias externas a las bibliotecas principales del programa, así como la puesta a punto de los entornos donde se ha desarrollado el prototipo tanto físico como a nivel software.

12.3. Resumen

A continuación, en la siguiente tabla se mostrará un resumen del coste total del proyecto realizado, en ella se hará uso de los dos costes totales anteriores y se le aplicará un impuesto del 21% (I.V.A), luego, el coste total del proyecto asciende a **NUEVE MIL SIETE EUROS CON TREINTA Y SEIS CÉNTIMOS.**

Apartado	Descripción	Importe
10.1	MATERIALES	194,10 €
10.2	MANO DE OBRA	7.250,00 €
	PRESUPUESTO (SIN I.V.A) ……	7.444,10 €
	21% I.V.A …….	1.563,26 €
	**PRESUPUESTO TOTAL …….. **	**9.007,36 €**

Tabla 6. Presupuesto final

13. Líneas futuras

Con respecto a líneas futuras, encontramos un proyecto ambicioso, cuyo desarrollo lleva estudiándose años y que, a día de hoy, es una realidad que muchos vehículos están implantando. De cara al futuro se podría decir que esto solo ha sido un pequeño comienzo, ya que las mejoras que se pueden realizar son variadas.

En primer lugar, se podría completar el sistema, es decir, que no solo sea detección de somnolencia, si no, que sea también detector de fatiga, ya que ésta en multitud de ocasiones deja ver una alta probabilidad de aparición de somnolencia. Algunas ideas para realizar serían por ejemplo el desarrollo de un sistema que detecte los bostezos, así como el grado de inclinación de la cabeza y la monitorización de la posición del cuerpo en el asiento, para detectar si el conductor se mueve más de lo normal (lo que sería un indicador de fatiga).

Por otro lado, el sistema también se podría completar mediante la monitorización del ritmo cardiaco y la tensión arterial, para saber si se está produciendo por ejemplo un episodio de estrés agudo, típico de un atasco en hora punta, lo que debilitaría la concentración llegando a producirse un agotamiento físico desmesurado.

Por último, algo que le daría una completa funcionalidad, sería agregar un sistema de alerta que detecte cuando el conductor cambia de carril de forma involuntaria debido a un excesivo estado de agotamiento, a una distracción o por falta de atención.

Como se puede observar, las ideas para aumentar la seguridad vial con ayuda de la tecnología son infinitas, de modo que, para este proyecto las líneas de futuro son prometedoras ya que se podría innovar de manera continuada.

14. Conclusiones

Este proyecto se creó con el objetivo principal de diseñar y desarrollar un sistema de somnolencia en tiempo real, a través del monitoreo del parpadeo de los ojos. Finalmente, se puede concluir que el objetivo se ha conseguido. Además, se ha conseguido superando objetivos adicionales, ya que también se ha realizado el desarrollo de un asistente virtual inteligente al que se le ha integrado un sistema de voz, con el que el usuario se puede comunicar para que le ayude a valorar la situación y además se ha conseguido desarrollar por partida doble, puesto que además del software, se dispone de un prototipo físico real.

Por tanto, en líneas generales se puede estar satisfecho del trabajo realizado, ya que tanto el desarrollo del sistema, como montaje y redacción entran dentro de los cánones esperados. Además, después de estudiar las inmensas posibilidades que tiene el proyecto de desarrollarse en diferentes vertientes, no se descarta tomar el proyecto de fin de grado como un proyecto de desarrollo personal en el mundo de la ingeniería y seguir desarrollándolo y mejorándolo hasta realizar un sistema ADAS tremendamente completo.

Por otro lado, cabe destacar que se han afianzado conceptos que quizás hasta ahora no estaban tan claros como parecían, es decir, se ha aprendido sobre Inteligencia Artificial, Machine Learning y Deep Learning. Principalmente, se ha aprendido a diferenciarlos y a encajar cada uno de ellos en su correspondiente lugar, ya que están relacionados entre sí, pero no son lo mismo.

También, gracias a este proyecto se ha tenido familiarización con la gama de dispositivos Raspberry, dispositivo puntero en el campo del aprendizaje de la ingeniería ya que nos brinda una minicomputadora para poder realizar infinidad de trabajos, por lo que ha sido una experiencia gratificante a la par que educativa.

Por último, queda la satisfacción de haber realizado un proyecto original, con el que se ha aprendido mucho y además se ha producido una familiarización con uno de los lenguajes de programación que se encuentra en auge, y por el momento es el más puntero, debido a sus prestaciones.

15. Anexos

Anexo I. Configuración inicial de Raspberry

Para realizar la instalación de Raspbian en el dispositivo Raspberry Pi, primero hay que dirigirse al siguiente enlace, https://www.raspberrypi.com/software/ . Una vez dentro, se procederá a descargar la imagen acorde al sistema operativo en el que se va a ejecutar el instalador de imagen, en este caso, Windows.

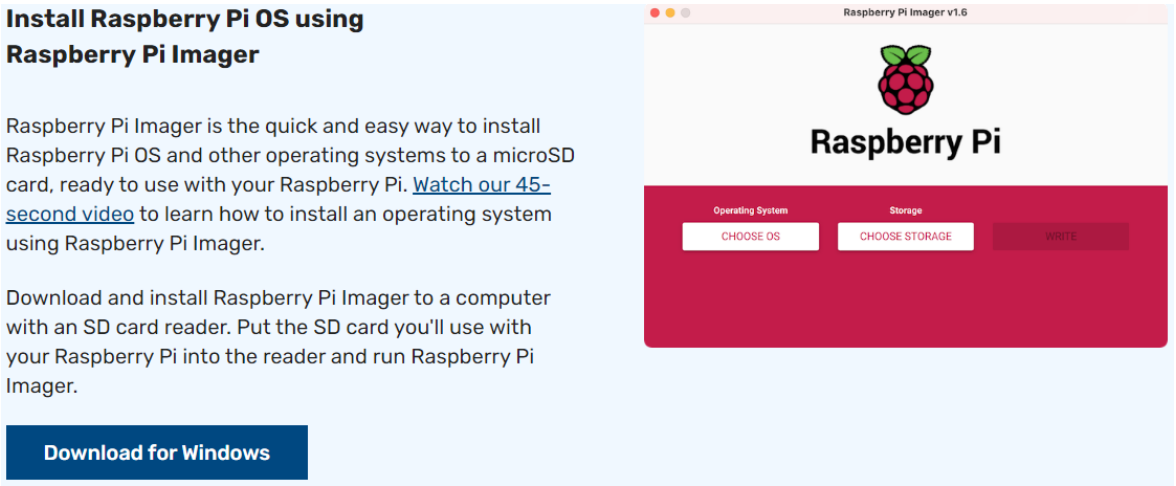

Figura 55. Raspberry PI Imager

Fuente: https://www.raspberrypi.com/software/

Posteriormente, se procederá a la instalación de la imagen. Una vez instalada se introducirá en el equipo la tarjeta SD con su correspondiente adaptador, para que el equipo pueda leerla, no hará falta realizar un formateo de la misma puesto que la imagen descargada anteriormente se encargará de realizar las particiones pertinentes y de realizar su formateo correspondiente.

Al abrir la imagen descargada de Raspberry aparecerá la siguiente pantalla:

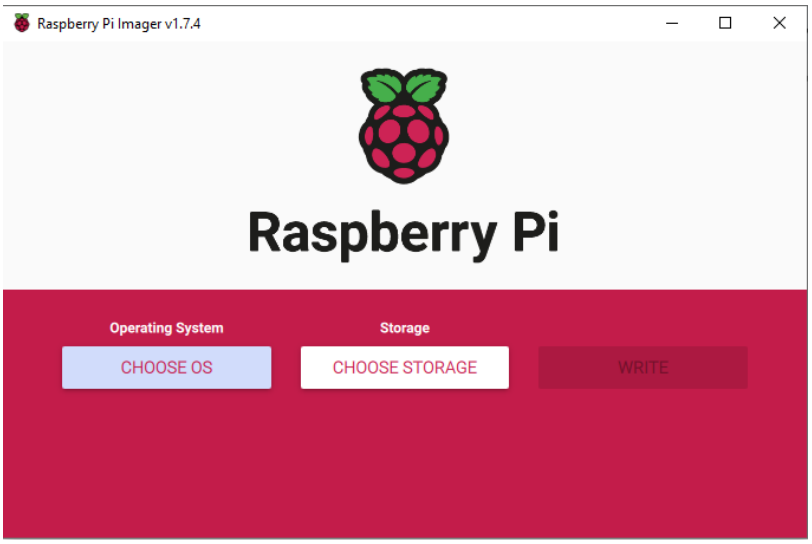

Figura 56. Entorno Imager

En ella se deberá escoger la tarjeta SD que se ha introducido en el equipo que posteriormente irá dentro de la Raspberry PI y será la encargada de correr el sistema operativo, por otro lado, a nivel de sistema operativo son multitud de opciones de las que se disponen, se cogerá la más conveniente a las necesidades, en este caso, para este proyecto se ha escogido la siguiente.

Figura 57. Selección OS para Raspberry

Una vez seleccionado, se debe prestar atención a los siguientes pasos. Raspberry 4 ya no tiene una configuración por defecto de nombre y contraseña predeterminada, por motivos de seguridad, por lo que se deberá realizar la configuración previamente antes de escribir sobre la tarjeta SD.

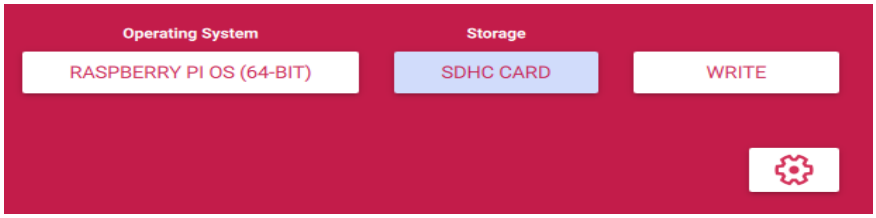

Figura 58. OS y Almacenamiento

Al pulsar sobre la imagen de settings, se abrirá un menú donde se pueden realizar múltiples configuraciones, algunas de las cuales se realizarán son las siguientes.

1. Habilitar ssh.
2. Establecer un nombre para raspberry.
3. Establecer una contraseña.
4. Configurar una red WiFi.
5. Establece la zona de huso horario.

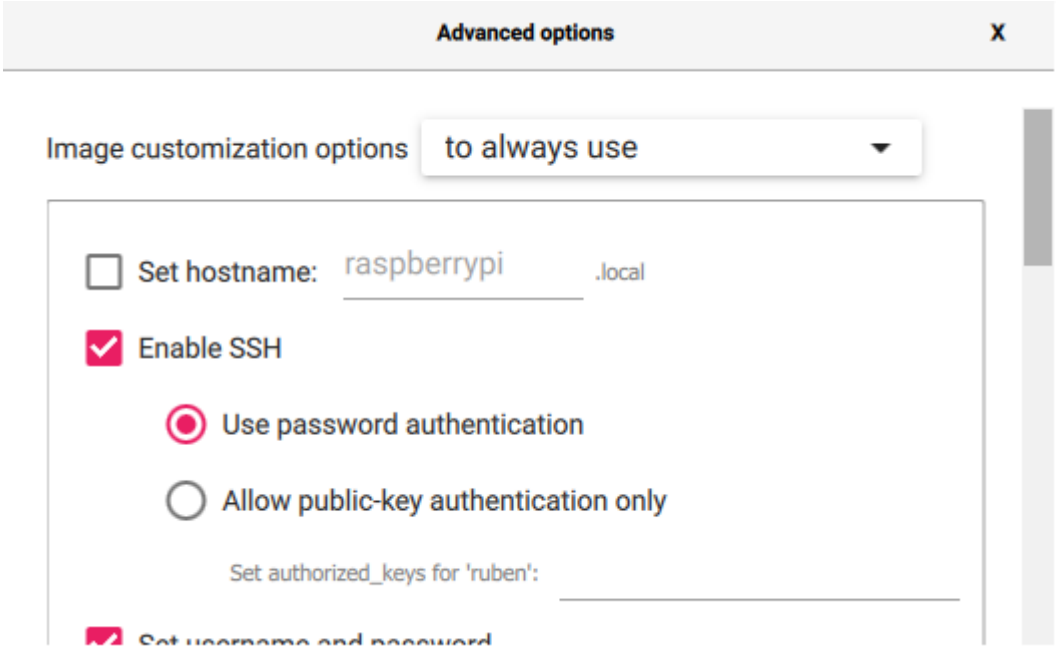

Figura 59. Opciones de configuración 1

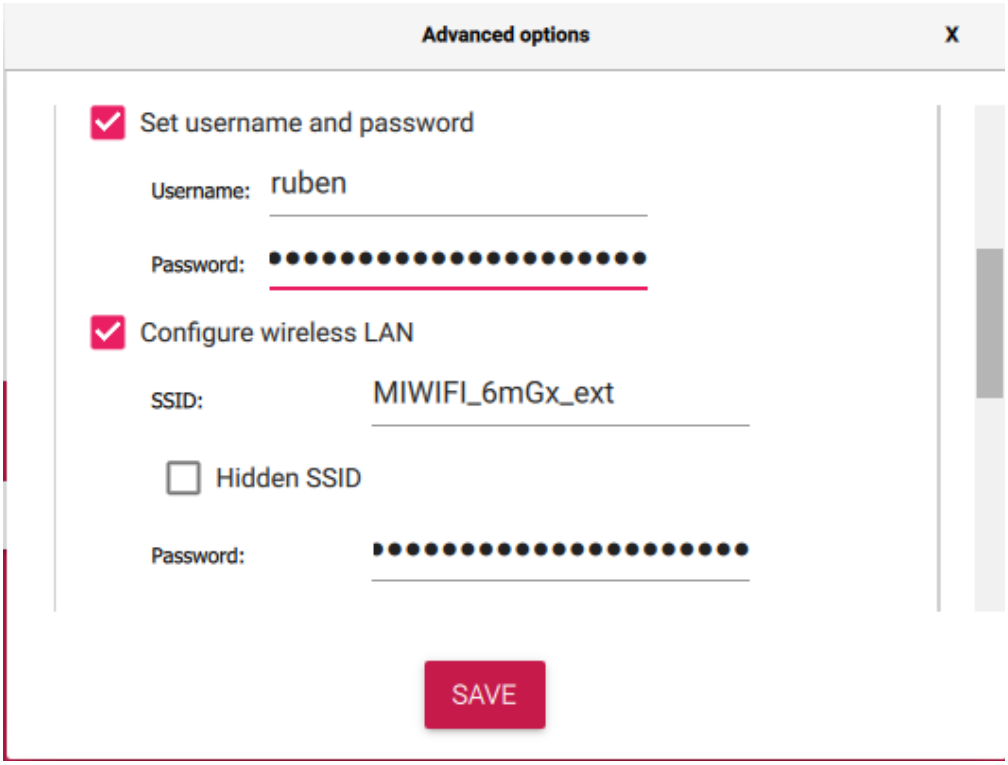

Figura 60. Opciones de configuración 2

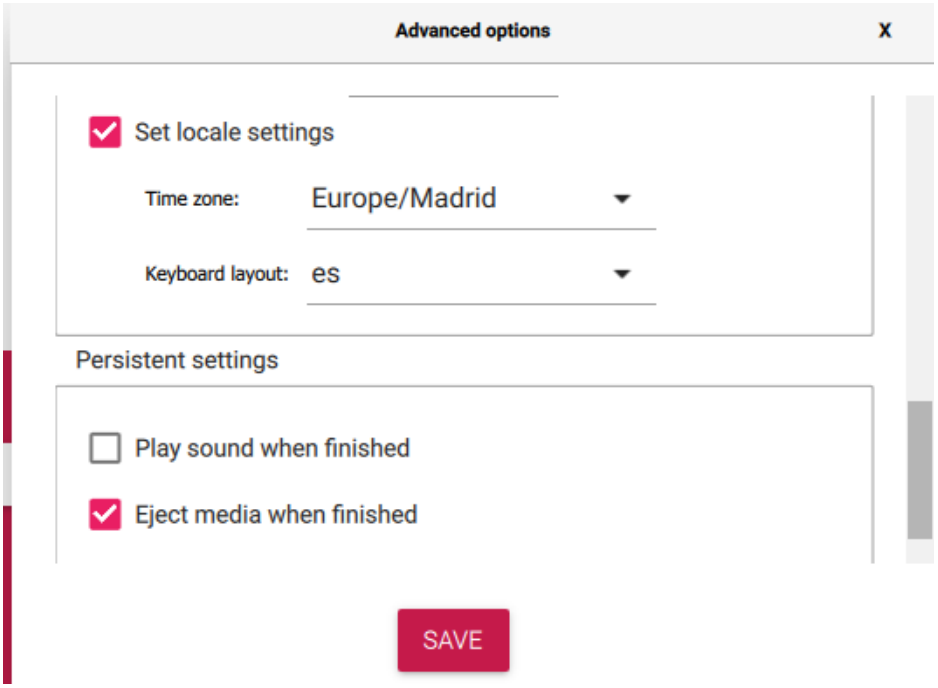

Figura 61. Opciones de configuración 3

Una vez que se haya finalizado este proceso, se hará clic en la pantalla principal del imager de Raspberry sobre escribir (write), este proceso tardará unos minutos.

Una vez que el proceso haya finalizado, se retirará la tarjeta SD y se volverá a insertar de manera instantánea. Cuando aparezca, se deberá crear un archivo de texto (que no tiene por qué contener nada necesariamente) que tenga como nombre SSH, esto se debe a que anteriormente se ha habilitado, pero éste no se encuentra corriendo. De modo que cuando la raspberry arranque y lea ese fichero con el nombre de ssh, iniciará el servicio.

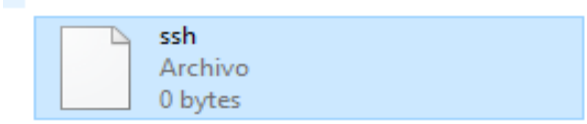

Figura 62. Opciones de configuración 4

Una vez finalizado este proceso, se insertará la tarjeta SD en la Raspberry Pi como se muestra en la siguiente figura (figura 61).

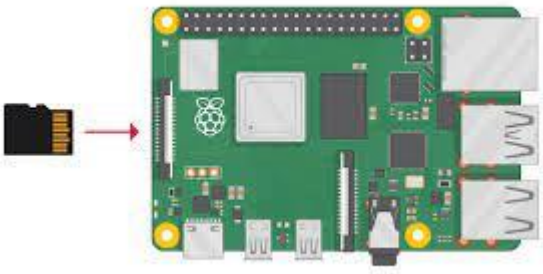

Figura 63. Insertar SD en Raspberry

Finalmente, ya se podrá conectar a la corriente la Raspberry Pi para hacer uso de ella. En el siguiente anexo **(Anexo II. Acceso remoto a Raspberry)**, se pueden ver los pasos a seguir para realizar con éxito dicho proceso.

Anexo II. Acceso remoto a Raspberry

El acceso remoto a la Raspberry va a permitir libertad para poder controlarla desde un equipo externo en cualquier lugar, en el momento que se desee. Otra de las principales ventajas que aporta este tipo de conexión es que no existe la necesidad de usar un monitor, teclado y ratón específico para tener que conectarlos de forma manual a la Raspberry, por tanto, si se realizan los siguientes pasos se ganará mucha flexibilidad a la hora de gestionar el dispositivo.

1. Encontrar dispositivo Raspberry

La Raspberry se puede haber configurado de varias maneras, una de ellas puede ser realizando un archivo de configuración en la SD, en el cuál se establecerá la red WiFi a la que se quiera que se conecte, indicando el SSID de la misma. O bien, conectándola por primera vez a un router para que le asigne una dirección IP por DHCP. Es importante saber en qué rango de direccionamiento IP se va a mover la Raspberry. Por tanto, es altamente recomendable que la primera vez que se quiera encontrar el dispositivo en la red, el ordenador con el que se vaya a producir la conexión remota se encuentre dentro del mismo rango de direccionamiento IP, de esta forma se agilizará el proceso.

En este caso, se ha conectado la Raspberry directamente al router, haciendo uso de un cable Ethernet. Como se puede apreciar en la siguiente figura (figura 62).

Figura 64. Conexión Raspberry al router

Una vez completado el paso anterior, es necesario disponer de un software que sea capaz de monitorear la red en busca de dispositivos que se encuentran en la misma con el objetivo de obtener su dirección IP. En este caso se ha hecho uso del software IpScanner, ya que se puede ejecutar sin necesidad de realizar ninguna instalación.

Una vez que el programa se encuentra abierto, en la barra de búsqueda, se escribe el rango de direccionamiento IP en el que el software debe buscar dispositivos, en este caso, como se puede ver en la siguiente figura, el programa detecta la Raspberry PI.

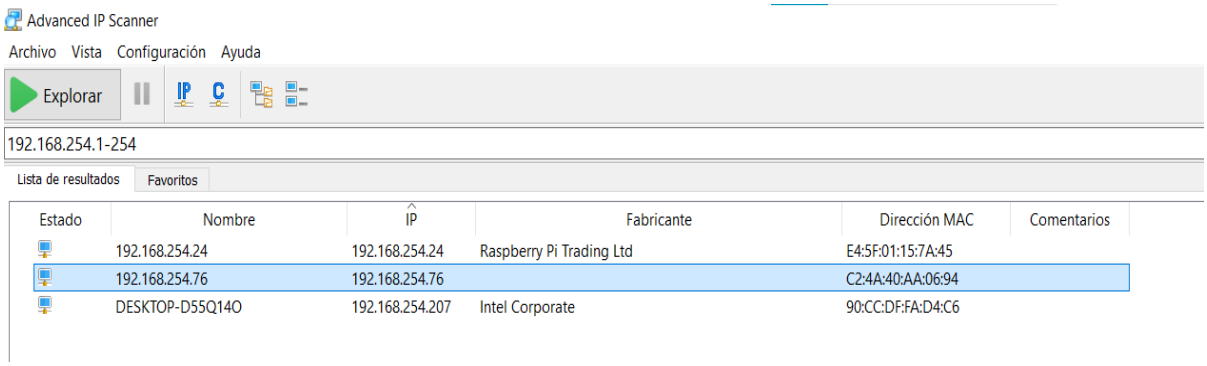

Figura 65. Escaneo de red

En caso de no conocer el rango de direccionamiento IP en el que se mueve el router al que estás conectado, podemos hacer uso de la ventana de comandos del equipo para ver el direccionamiento, basta simplemente con poner el comando ipconfig y se podrá observar información relacionada con cada uno de los adaptadores disponibles en el equipo, en este caso en la siguiente figura se muestra la información relaciona con el adaptador que se estaba usando en ese momento.

```
Adaptador de LAN inalámbrica Wi-Fi:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::8c85:429:5715:81ff%4
   Dirección IPv4. . . . . . . . . . . . : 192.168.254.207
   Máscara de subred . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.254.76
```

Figura 66. Especificaciones adaptador inalámbrico

2. Establecer conexión SSH

Lo primero que se debe realizar es una conexión SSH a la Raspberry. Esto se debe a que se pretende modificar la configuración de la misma para habilitar el servidor VNC que trae integrado, del que después se hará uso para la conexión remota. Para realizar la conexión se deberá hacer uso de la línea de comandos del equipo, abriéndola con la siguiente combinación de teclas, Wnd + r, y poniendo posteriormente en la ventana emergente que aparece cmd.

Una vez en la línea de comandos, se procede a realizar la conexión SSH, recordar que para poder llevarla a cabo, se debe de haber creado un fichero con el nombre de ssh en la SD en la que se encuentra el sistema Raspbian de la Raspberry, para establecer la conexión se debe poner lo siguiente (ver la figura 65) ssh nombre_raspberry@dirrección_ip, posteriormente, se deberá de poner la contraseña que se le haya establecido, en este caso es raspberry y por último se accederá al menú de configuración haciendo uso del comando “sudo raspi-config”.

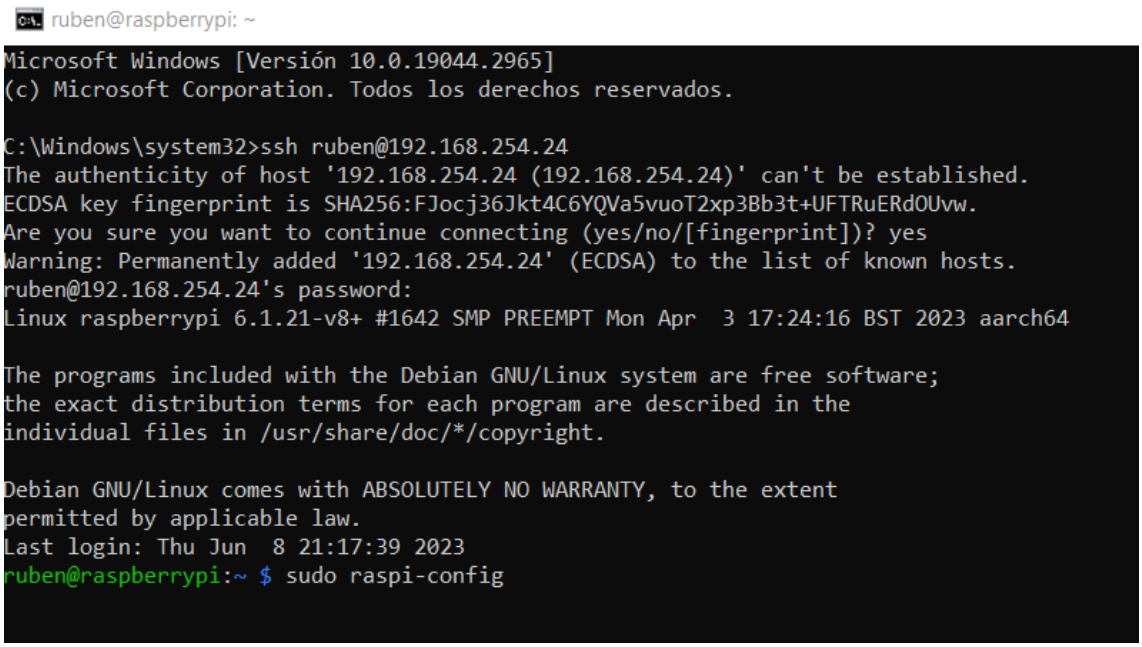

Figura 67. Acceso a raspberry por SSH

3. Activación del servidor VNC

Una vez dentro del menú de configuración de Raspberry, habrá que dirigirse mediante las flechas del teclado hacia la opción de “Opciones de interfaz”

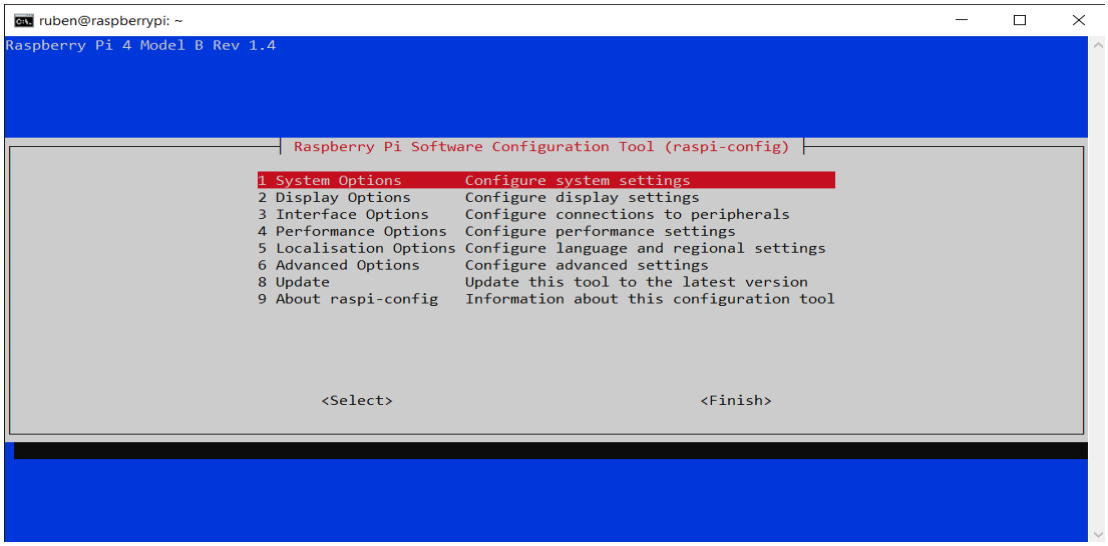

Figura 68. Menú de configuración de Raspberry

Encontrada dicha opción, en este caso es la tercera, se pulsará enter y se accederá a otro menú, allí se encontrarán las diferentes formas de conexión a la Raspberry a través de sus distintas interfaces y por tanto, habrá que habilitar la opción que pone VNC.

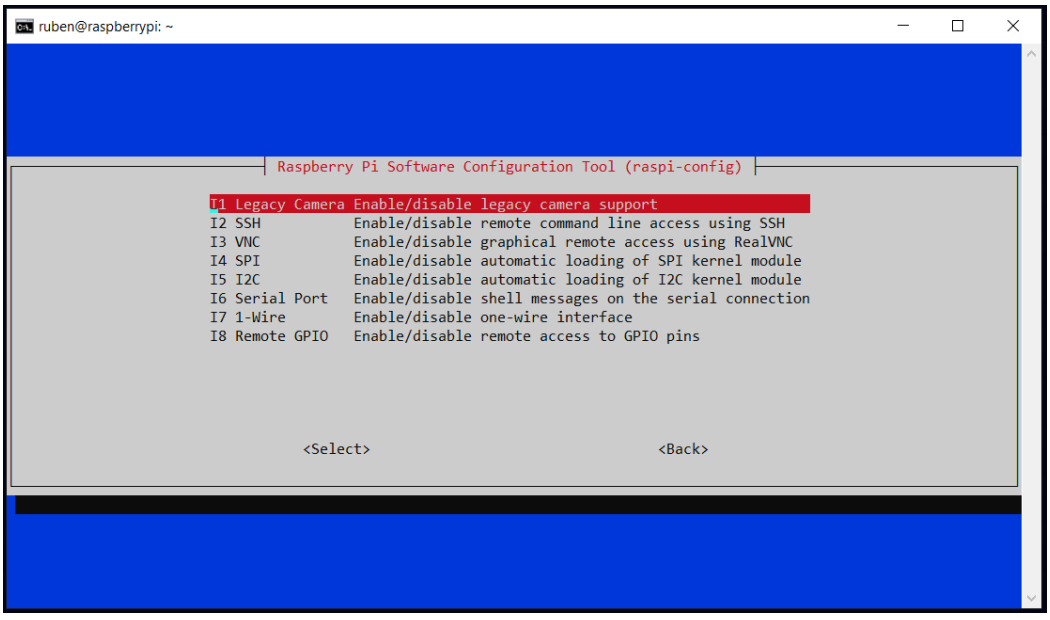

Figura 69. Menú de configuración de Raspberry 2

Una vez que se pulsa sobre VNC aparecerá una ventana, donde dice si se desea habilitar el servicio, habrá que colocarse sobre la opción afirmativa y dar enter, y ya estaría el servidor VNC activo para poder realizar la conexión de forma remota.

Figura 70. Habilitar SSH

4. Descargar la aplicación RealVNC

Para realizar la conexión de forma remota es necesario disponer de un software que permita hacer uso del servicio VNC, esta aplicación se puede encontrar en el siguiente enlace https://www.realvnc.com/es/ . Una vez descargada, se establecerá la conexión remota a través del servidor VNC, para ello se accederá a la aplicación RealVNC, (posteriormente se debe haber habilitado la conexión VNC en la Raspberry habiendo hecho uso de una conexión SSH, ver puntos anteriores), en el menú de arriba, se podrá establecer la conexión de dos maneras, bien desde la pestaña de archivo como se muestra en la siguiente figura (figura 69).

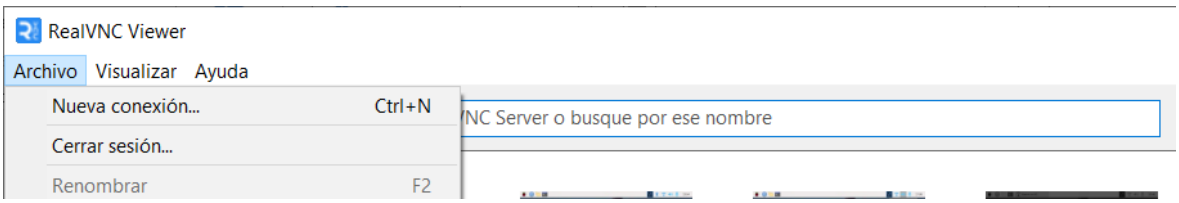

Figura 71. Establecimiento de conexión VNC.

La siguiente ventana que aparecerá, pedirá rellenar algunos datos, como por ejemplo el nombre indicativo que se quiere que tenga la conexión, así como la dirección IP del servidor VNC, que en este caso será la dirección IP de la Raspberry, ya que se le ha activado el servicio anteriormente de servidor VNC. Las demás opciones se pueden dejar por defecto.

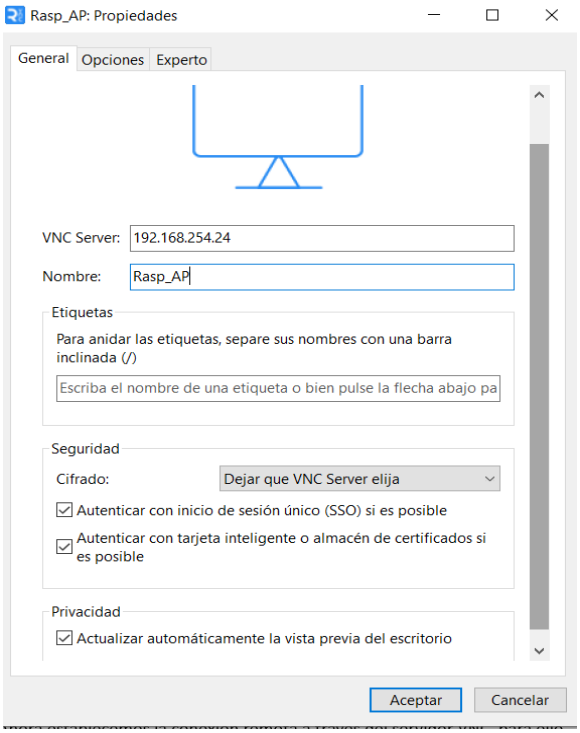

Figura 72. Establecer conexión 1

O bien, indicando la dirección IP encontrada posteriormente con IpScanner, en la barra de búsqueda principal del programa RealVNC, como se muestra en la siguiente figura (figura 71).

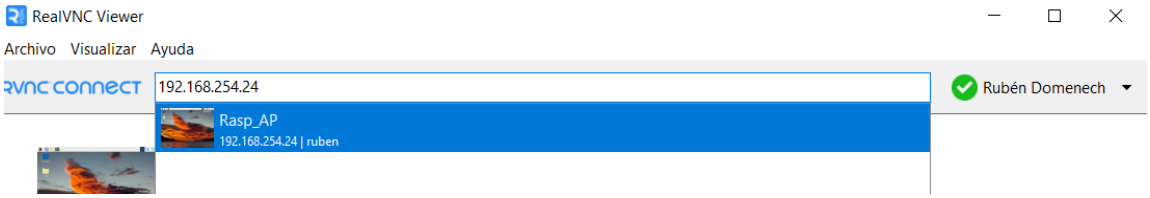

Figura 73. Establecer conexión 1

Una vez que se haya completado uno de los dos pasos anteriores, el que se prefiera. Aparecerá una ventana de autenticación, donde habrá que poner el nombre

asignado a la Raspberry en el inicio de la configuración, cuando se montó la imagen **(ver Anexo I. Configuración inicial de Raspberry)**, y la contraseña, que en este caso es Raspberry.

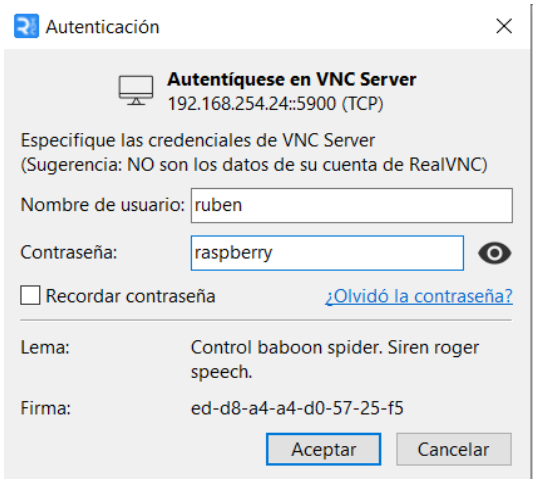

Figura 74. Credenciales

Una vez realizados los pasos anteriores, se establecerá conexión con la Raspberry y podremos visualizar, su entorno gráfico, el cual ya se podrá manipular como si fuera un equipo de casa.

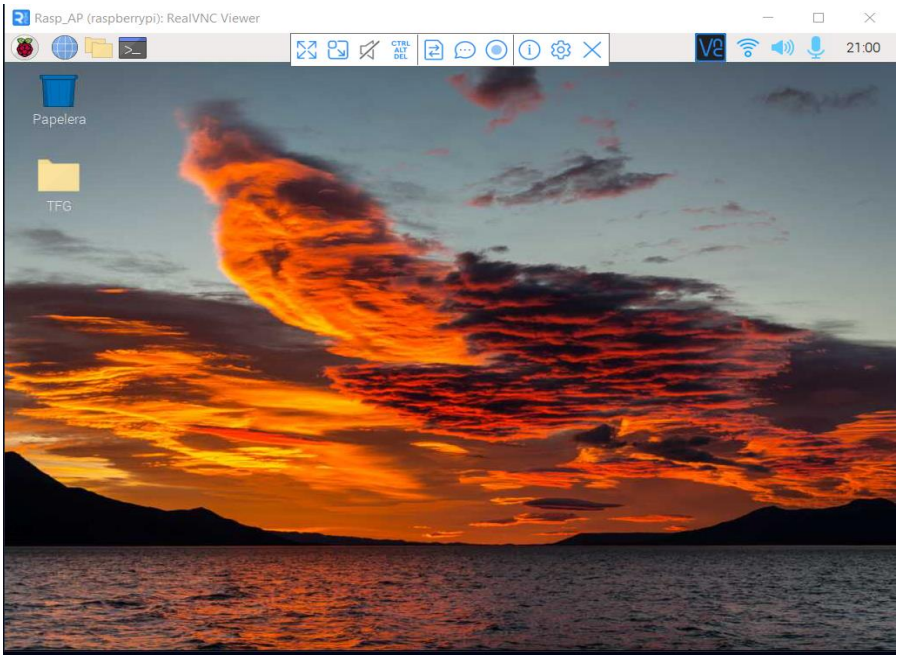

Figura 75. Escritorio remoto de Raspberry

Anexo III. Aspectos clave para correcto funcionamiento

En este anexo se hará referencia a unos detalles clave para el correcto funcionamiento de ambos resultados finales, además se dará una guía de buenas prácticas de cómo se debería probar para obtener una experiencia óptima.

En primer lugar, el código se ha desarrollado en un entorno Windows, las rutas existentes para hacer uso de los archivos necesarios dentro del proyecto son relativas, el problema reside en la forma de escribirse las rutas en los distintos sistemas operativos. Luego, recordar que en caso de descargar el software para Raspberry y que ésta monte un SO Linux, en este proyecto Raspbian, se deberán modificar todas las rutas existentes.

En segundo lugar, el asistente de voz se ha realizado en torno a las prestaciones que ofrecían los dispositivos utilizados, por tanto, comprobar primero que las especificaciones programadas se ajusten a las del equipo en donde se va a hacer uso del programa. En los equipos, se pueden tener distintos tipos de voces y en diferentes idiomas y sobre todo en distinto orden, por lo que habrá que asegurarse de cuál es la correcta antes de hacer uso del mismo. Esto es aplicable tanto para equipo convencional como para Raspberry.

Recalcar la necesidad de instalación de todas las dependencias en los apartados correspondientes a bibliotecas. Además, es necesario disponer de una API Key para hacer uso del servicio de geolocalización, para ello se deberá estar registrado en: https://ipstack.com/?utm_source=google&utm_medium=cpc&utm_campaign=ipstack_search_eu&gclid=CjwKCAjw-b-kBhB-EiwA4fvKrJebTQQXMQcswcn0OY-gDHfgmhs0uxNO2clgfyChL2Go_Pclv1eR2hoC5C4QAvD_BwE

Una vez que se tenga, esa API Key, se deberá cambiar en el fichero cloud.py, como se muestra en la siguiente figura:

```
# API key para ipstack (reemplaza con tu propia clave de API)
api_key = '05da2c81113e9befba021f553e31263b'
```

Figura 76. API Key IPStack

Otro aspecto a tener en cuenta es la diferencia de fluidez en cuanto a funcionamiento, como se comentó en el punto de Resultados finales, en la Raspberry el

sistema de detección de somnolencia será mucho más lento, esto se debe a las características que brinda el dispositivo Hardware. Si se desea una mejora en el funcionamiento del mismo se podría realizar usando una memoria SSD en ver de una tarjeta SD, corriendo un SO sin interfaz gráfica, puesto que ello conlleva un alto consumo de recursos, etcétera.

Es importante que, si se produce la descarga del proyecto, y se hace una reestructuración de los archivos modificando la ruta de estos, el proyecto puede quedar inservible, ya que hay muchos ficheros Python que se relaciona entre sí. Por ello, para una buena experiencia de uso, se recomienda seguir todos los pasos detallados en los diferentes puntos de la memoria, evitando así que se corrompa el proyecto.

Como guía de buenas prácticas y para tener una satisfacción plena del uso y disfrute del proyecto se recomienda seguir los siguientes pasos una vez que se ha ejecutado el proyecto en un dispositivo:

1. Realizar en varias ocasiones y de forma continuada apertura y cierre de los ojos, tanto de manera rápida, como lenta.
2. Una vez realizado el paso anterior durante un determinado tiempo, cerrar los ojos hasta que se detecte somnolencia, una vez detectada se activará la alarma. Reaccionar ante esta alarma abriendo los ojos.
3. Cuando el sistema detecte que el usuario ha reaccionado ante el estimulo de la alarma, preguntará por el estado del sujeto. Una vez que se haga la pregunta, contestar al asistente dejando un segundo de margen para que se active la escucha.
4. El audio debe ser relativamente corto y conciso, ya que se dispone de un tiempo concreto de captación de sonido, concretamente 8 segundos.
5. Una vez realizado los pasos anteriores y habiendo obtenido la respuesta del asistente virtual, realizar el paso número 1, para simular un viaje largo.
6. Posteriormente, ya se podrá poner a prueba el bloque número tres, situación de alto riesgo. Para ello, se cerrarán los ojos y no se abrirán bajo ninguna circunstancia, aunque suene la alarma. Transcurridos unos segundos se activará el protocolo de peligro máximo, donde se simulará la activación de conducción inteligente del vehículo y se procederá a enviar un mensaje a la nube (La matricula se puede cambiar en el fichero cloud.py).

Si la prueba de funcionamiento se ha realizado en la Raspberry haciendo uso del entorno virtual que se hace casi obligatorio su uso, o por el contrario en un equipo convencional del que también se ha hecho uso de un entorno virtual, es altamente recomendable, desactivar el entorno virtual una vez finalizada la ejecución del programa. Para ello en la ventana de comandos donde se este ejecutando el mismo, dentro del entorno virtual, habrá que ejecutar el siguiente comando: **deactivate .**

16. Bibliografía

[1] Qué es la Inteligencia Artificial. [en línea], [19 abril 2023]. [consulta: 26 abril 2023]. Disponible en: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr.

[2] CORPORATIVA, I. Descubre los principales beneficios del Machine Learning. *Iberdrola* [en línea]. [consulta: 26 abril 2023]. Disponible en: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico.

[3] ¿Qué es IoT (Internet Of Things)? *Deloitte Spain* [en línea]. [consulta: 26 abril 2023]. Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html.

[4] Influencia de la somnolencia en los accidentes de tráfico en España (2011-2015). Fundación Línea Directa. [en linea], [17 Julio 2017]. [consulta: 27 abril]. Disponible en: https://acortar.link/bTzvd7

[5] El sueño y la fatiga en la conducción. ¿Cuáles son los hábitos de los conductores españoles? Informe sobre la influencia de la fatiga y el sueño en la conducción. Fundación CEA. [en línea]. [8 de julio, 2015]. [Consulta en: 2 mayo, 2023] Disponible en: https://www.fundacioncea.es/np/pdf/estudio-somnolencia-al-volante.pdf

[6] Bosch llevará un nuevo detector de fatiga basado en inteligencia artificial al CES 2020. [en línea], [sin fecha]. [consulta: 4 mayo 2023]. Disponible en: https://www.motor.es/noticias/bosch-detector-fatiga-inteligencia-artificial-ces-2020-201963373.html.

[7] Python: qué es y por qué deberías aprender a utilizarlo. [en línea], [sin fecha]. [consulta: 5 mayo 2023]. Disponible en: https://www.becas-santander.com/es/blog/python-que-es.html.

[8] Ventajas y Desventajas de Python | KeepCoding Bootcamps. [en línea]. [consulta: 5 mayo 2023]. Disponible en: https://keepcoding.io/blog/ventajas-y-desventajas-de-python/.

[9] Qué es Visual Studio Code y qué ventajas ofrece. *OpenWebinars.net* [en línea], 2022. [consulta: 5 mayo 2023]. Disponible en: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/.

[10] ¿Qué es IoT (Internet Of Things)? *Deloitte Spain* [en línea]. [consulta: 5 mayo 2023]. Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html.

[11] What is IoT (Internet of Things) and How Does it Work? - Definition from TechTarget.com. [en línea], [sin fecha]. [consulta: 6 mayo 2023]. Disponible en: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT.

[12] ¿Qué es la inteligencia artificial o IA? | Google Cloud | Google Cloud. [en línea], [sin fecha]. [consulta: 8 mayo 2023]. Disponible en: https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419.

[13] Deep Learning qué es y por qué es clave para la inteligencia artificial - Iberdrola. [en línea], [sin fecha]. [consulta: 10 mayo 2023]. Disponible en: https://www.iberdrola.com/innovacion/deep-learning.

[14] Inteligencia Artificial | Machine Learning | Deep Learning | Redes Neuronales. Relaciones y USOS - YouTube. [en línea], [sin fecha]. [consulta: 12 mayo 2023]. Disponible en: https://www.youtube.com/watch?v=DFUGEe5F5WA.

[15] ¿Qué es OpenCV y para qué sirve? 👁 [2021]. *https://www.crehana.com* [en línea], [28 abril 2021]. [consulta: 13 mayo 2023]. Disponible en: https://www.crehana.com/blog/transformacion-digital/que-es-opencv/.

[16] ¿Qué es Keras en Deep Learning? | KeepCoding Bootcamps. [en línea]. [consulta: 14 mayo 2023]. Disponible en: https://keepcoding.io/blog/keras-en-deep-learning/.

[17] GONZALEZ, L., 2020. Librería NumPy. *Aprende IA* [en línea]. [consulta: 14 mayo 2023]. Disponible en: https://aprendeia.com/libreria-de-python-numpy-machine-learning/.

[18] math — Funciones matemáticas. *Python documentation* [en línea], [sin fecha]. [consulta: 16 mayo 2023]. Disponible en: https://docs.python.org/3/library/math.html.

[19] PHERKAD, 2017. Python 3 para impacientes: El módulo time. *Python 3 para impacientes* [en línea]. [consulta: 16 mayo 2023]. Disponible en: https://python-para-impacientes.blogspot.com/2017/03/el-modulo-time.html.

[20] HTTPS://WWW.FACEBOOK.COM/GROKKEEPCODING, 2020. ¿Qué son los Datasets? [4 sitios donde encontrarlos]. [en línea]. [consulta: 16 mayo 2023]. Disponible en: https://keepcoding.io/blog/que-son-datasets/.

[21] MRL Eye Dataset | MRL. [en línea], [sin fecha]. [consulta: 16 mayo 2023]. Disponible en: http://mrl.cs.vsb.cz/eyedataset.

[22] Python | Cascadas de Haar para la detección de objetos – Barcelona Geeks. [en línea], [sin fecha]. [consulta: 16 mayo 2023]. Disponible en: https://barcelonageeks.com/python-cascadas-de-haar-para-la-deteccion-de-objetos/.

[23] random —Generar números pseudoaleatorios. *Python documentation* [en línea], [sin fecha]. [consulta: 17 mayo 2023]. Disponible en: https://docs.python.org/3/library/random.html.

[24] Cómo trabajar con datos JSON utilizando Python. *Code Envato Tuts+* [en línea], 2016. [consulta: 17 mayo 2023]. Disponible en: https://code.tutsplus.com/es/tutorials/how-to-work-with-json-data-using-python--cms-25758.

[25] io — Herramientas principales para trabajar con streams — documentación de Python - 3.9.16. [en línea], [sin fecha]. [consulta: 17 mayo 2023]. Disponible en: https://docs.python.org/es/3.9/library/io.html.

[26] os — Interfaces misceláneas del sistema operativo — documentación de Python - 3.10.11. [en línea], [sin fecha]. [consulta: 18 mayo 2023]. Disponible en: https://docs.python.org/es/3.10/library/os.html.

[27] *dill: serialize all of python* [en línea], [sin fecha]. Python. S.l.: s.n. [consulta: 19 mayo 2023]. Disponible en: https://github.com/uqfoundation/dill.

[28] LEÓN, E., 2020. Procesamiento del lenguaje natural (PLN) con Python. *BAOSS* [en línea]. [consulta: 19 mayo 2023]. Disponible en: https://www.baoss.es/procesamiento-del-lenguaje-natural-pln-con-python/.

[29] PROGRAMACIONPYTHON80889555, 2020. MANIPULANDO AUDIOS EN PYTHON, CON «pydub». *El Programador Chapuzas* [en línea]. [consulta: 20 mayo 2023]. Disponible en:

https://programacionpython80889555.wordpress.com/2020/02/25/manipulando-audios-en-python-con-pydub/.

[30] Un tutorial esencial de texto a voz de Python utilizando la biblioteca pyttsx3 | HackerNoon. [en línea], [sin fecha]. [consulta: 22 mayo 2023]. Disponible en: https://hackernoon.com/an-essential-python-text-to-speech-tutorial-using-the-pyttsx3-library.

[31] Introducing Whisper. [en línea], [sin fecha]. [consulta: 22 mayo 2023]. Disponible en: https://openai.com/research/whisper.

[32] Whisper: usa gratis esta AI para el reconocimiento de voz. Platzi [en línea], [sin fecha]. [consulta: 22 mayo 2023]. Disponible en: https://platzi.com/blog/whisper-usa-gratis-esta-ai-para-el-reconocimiento-de-voz/.

[33] openai-whisper: Robust Speech Recognition via Large-Scale Weak Supervision [en línea], [15 marzo 2023]. S.l.: s.n. [consulta: 22 mayo 2023]. Disponible en: https://github.com/openai/whisper.

[34] SCHOOL, T., 2022. ¿Qué es y para qué sirve la librería Python Request? | Tokio. *Tokio School* [en línea]. [consulta: 22 mayo 2023]. Disponible en: https://www.tokioschool.com/noticias/python-request/.

[35] Mensajes HTTP - HTTP | MDN. [en línea], 2022. [consulta: 23 mayo 2023]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Messages.

[36] CANIZALES, M.M., 2022. Configurar entorno virtual Python. *Medium* [en línea]. [consulta: 23 mayo 2023]. Disponible en: https://m-monroyc22.medium.com/configurar-entorno-virtual-python-a860e820aace.