Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Interfaz de control domótica basada en visión artificial

Autor: Ricardo Durán Viñuelas
Tutores: Jose María Maestre Torreblanca
Jesús Iván Maza Alcañiz

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Interfaz de control domótica basada en visión artificial

Autor:
Ricardo Durán Viñuelas

Tutores:
Jose María Maestre Torreblanca
Profesor titular
Jesús Iván Maza Alcañiz
Profesor titular

Dpto. de Ingeniría de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Interfaz de control domótica basada en visión artificial

Autor: Ricardo Durán Viñuelas

Tutore: Jose María Maestre Torreblanca
Jesús Iván Maza Alcañiz

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2013

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Agradecer a Iván y a Pepe el tiempo que han dedicado en darme las indicaciones a seguir, así como la ayuda que me han brindado a lo largo de estos meses de trabajo.

A mi familia y amigos, porque siempre me han acompañado y ayudado cuando los ánimos decaían. Y de igual forma, agradecer a todos los compañeros de la escuela y resto de profesores toda la dedicación aportada a lo largo de estos años de formación. Ha sido un placer ser parte de la escuela y participar en este trabajo.

Ricardo Durán Viñuelas

Sevilla, 2020

Resumen

El trabajo que se mostrará a continuación consistirá en una interfaz domótica basada en visión artificial con la cual se buscará facilitar la integración domótica a personas discapacitadas, pudiendo estas interactuar mediante gestos faciales con la vivienda.

Para llevar a cabo este proyecto, se desarrollarán los códigos pertinentes que detecten los gestos faciales para posteriormente implementarlos en una vivienda virtual. Para ello, se utilizará el lenguaje de programación Python junto al software HOME I/O.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

TFG	Trabajo Fin de Grado
Enter	Tecla del teclado para entrar o ejecutar
Bool	Dato tipo lógico o booleano
:	Tal que
=	Igual
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
⇔	Si y sólo si
fps	Frames por segundo

1 INTRODUCCIÓN

El ser humano, en relación a la tecnología, está continuamente buscando la manera de superarse con el objetivo de avanzar hasta el punto de que estas puedan ayudar en sus propias necesidades. Gracias a los avances tecnológicos de hoy en día se cuentan con numerosas facilidades en el día a día de las personas que años atrás eran impensables, y, probablemente, dentro de unos años habrá otras tecnologías o avances que en la actualidad no son concebidos.

1.1 Motivaciones

La domótica nos facilita ciertas acciones cotidianas dentro de nuestro hogar además de brindar a este de cierta inteligencia o seguridad. Encender las luces a distancia o que un motor se encargue de subir o bajar las persianas pueden hacer que la comodidad y productividad de los usuarios aumente. Pero no solo eso, y es que para personas discapacitadas o con dificultades en la movilidad estas acciones pueden ser de enorme importancia y la única forma de poder realizarlas por sí solas es gracias a estas aplicaciones de la domótica.

Es por ello por lo que en este trabajo fin de grado se desea automatizar los procesos cotidianos del día a día en la vivienda a través de gestos faciales, como puede ser abrir la boca o parpadear varias veces, pensando en aquellas personas que tienen movilidad únicamente en la cara. Se tratará de hacer una interfaz general y a grandes rasgos, siendo lo ideal que a cada usuario se le haga una adaptación a sus características físicas o preferencias.

1.2 Planteamiento

Para poder llevar a cabo este proyecto, se necesita hacer uso de la visión artificial. De acuerdo al blog de la empresa de automatización industrial Contaval, *"La visión artificial es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por una máquina. La principal finalidad de la visión artificial es dotar a la máquina de "ojos" para ver lo que ocurre en el mundo real, y así poder tomar decisiones para automatizar cualquier proceso."* («¿Qué es la visión artificial y para qué sirve?», 2016) [1]. Por lo tanto, se pretende que el ordenador y los dispositivos domóticos, y haciendo uso de una webcam o una cámara, sean capaces de detectar y reconocer en la cara del usuario distintos gestos, para luego poder automatizar ciertas acciones a través de los mismos.

Como lenguaje de programación se puede utilizar tanto C, como Matlab, Python o Java por poner algunos ejemplos. Pero, para llevar a cabo este proyecto, se ha tomado la decisión de utilizar Python como lenguaje principal de programación. Esta elección se debe a que en Python se pueden encontrar varias librerías dedicadas a la visión artificial que facilitarán la programación, además de ser un lenguaje cada vez más extendido en este y otros ámbitos. Debido a que no es del alcance del TFG, se tratarán de usar códigos de visión artificial ya creados por otras personas y de uso libre, o adaptaciones de los mismo a nuestras necesidades.

Para las pruebas de los códigos se usará Anaconda debido a la facilidad que ofrece a la hora de instalar librerías o uso del programa, aunque de igual manera, se podría usar cualquier otro entorno.

Una vez desarrollada la parte del reconocimiento de gestos, se implementarían dichos códigos en unos dispositivos domóticos. Sin embargo, y debido a la imposibilidad de disponer de estos, se usará un simulador de vivienda inteligente. Para ello, se hará uso del software HOME I/O, cuya licencia ha sido facilitada por los tutores de este trabajo.

1.3 Objetivos

De acuerdo con lo mencionado en esta introducción, el objetivo es desarrollar, a grandes rasgos y sin llegar a especificar en casos concretos, una interfaz que permita la automatización de acciones en el hogar a partir de gestos faciales, facilitándole a las personas discapacitadas o con dificultades estas tareas. Por ejemplo, encender una luz parpadeando tres veces o abrir la persiana abriendo la boca.

A su vez, se pretende abrir una vía de investigación para que posteriormente se pueda seguir mejorando y avanzando en esta tecnología, pudiéndose en un futuro usar de manera habitual y para cualquier persona en los hogares, facilitando así las acciones cotidianas.

2 LA DOMÓTICA

En un primer lugar, es conveniente hacer una introducción a la domótica, ya que debido a ella tiene sentido este proyecto. En los últimos años la domótica está cobrando especial interés gracias a los nuevos avances a los que se están llegando, haciendo posible la implementación de las nuevas tecnologías en cualquier hogar de forma más sencilla, además de encontrando una oferta mejor y de mayor calidad, acorde a acualquier usuario final.

2.1 ¿Qué es la domótica?

Accediendo a la RAE, se tiene que la palabra domótica está formada por la forma femenina del latín *domus* 'casa' y automá*tica*, y significa *"Conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda"* [2].

Otros autores definen la domótica como la tecnología implementada en los hogares y que automatiza las funciones o acciones que se realizan en estos con el objetivo de aumentar la comodidad, la seguridad y el ahorro de energía, mejorando así el nivel de vida de los ocupantes de la vivienda. Estas funciones se conseguirían a través de dispositivos que pueden funcionar de manera independiente o conjunta. [3]

En relación con esto último, en los primeros años estos dispositivos funcionaban de forma independiente, de modo que no era posible establecer interconexiones entre ellos. Hoy en día es posible automatizar el riego del jardín en función de si la luz está encendida o apagada, de forma que se riegue por la noche, por poner un ejemplo sencillo. Es por eso por lo que actualmente la domótica abarca un gran número de sistemas e interacciones entre ellos, siendo el propósito final el de tener todo el control de la vivienda únicamente haciendo uso de un dispositivo o aplicación. Sin embargo, actualmente, esto último sigue siendo difícil debido a la poca interoperabilidad entre distintas marcas o productos.

El término inmótica hace referencia a la tecnología que se implementa en inmuebles de mayor envergadura como pueden ser los edificios u hoteles. Los dispositivos que se requieren en este caso suelen ser de mayores dimensiones que los utilizados en la vivienda.

2.2 Propósitos de la domótica

De acuerdo al prunto anterior, la domótica debe de integrar las soluciones del hogar, proporcionando así una mayor calidad de vida, siendo este el objetivo del ser humano desde tiempos inmemoriables. Aún así, los propósitos de la domótica se pueden subdividir en:

- La domótica aporta comodidad al usuario, haciendo del hogar un lugar más confortable y facilita las tareas repetitivas que se realizan en él incrementando así el bienestar. Gestiona los dispositivos y actividades domésticas como pueden ser las acciones de abrir, cerrar, encender, apagar o regular cada uno de los sistemas de la casa.
- A su vez, ayuda a la gestión de la energía, favoreciendo un ahorro energético y económico. Esto se consigue regulando, programando y optimizando la energía y los dispositivos de nuestro hogar buscando así la eficiencia, por ejemplo, apagando los enchufes que no sean necesarios al salir de casa o programando el encendido de las luces solo cuando sea necesario.
- Aportar seguridad a los usuarios, tanto como la propia seguridad como la de los bienes materiales del hogar, favoreciendo así la tranquilidad de estos ante incidencias o averías que puedan aparecer. El uso de cámaras de seguridad o detectores de fuga de gas o de inundación son algunos de los ejemplos de este ámbito.

- Fomenta la accesibilidad, favoreciendo el manejo de los elementos del hogar a aquellas personas con dificultades, ya sea debido a las altas cargas de trabajo de la actividad en cuestión, o por motivos de lesiones o discapacidad de las personas, es decir, ayuda a las personas que así lo requieran.
- Garantiza las comunicaciones mediante el control y supervisión remoto de la vivienda a través de su teléfono, PC..., permitiendo la recepción de avisos o incidencias e información del funcionamiento de equipos e instalaciones. Además, la instalación domótica permite el intercambio de información a través de ella, haciendo posible la transmisión de datos o imágenes entre distintos dispositivos, por ejemplo. [4]

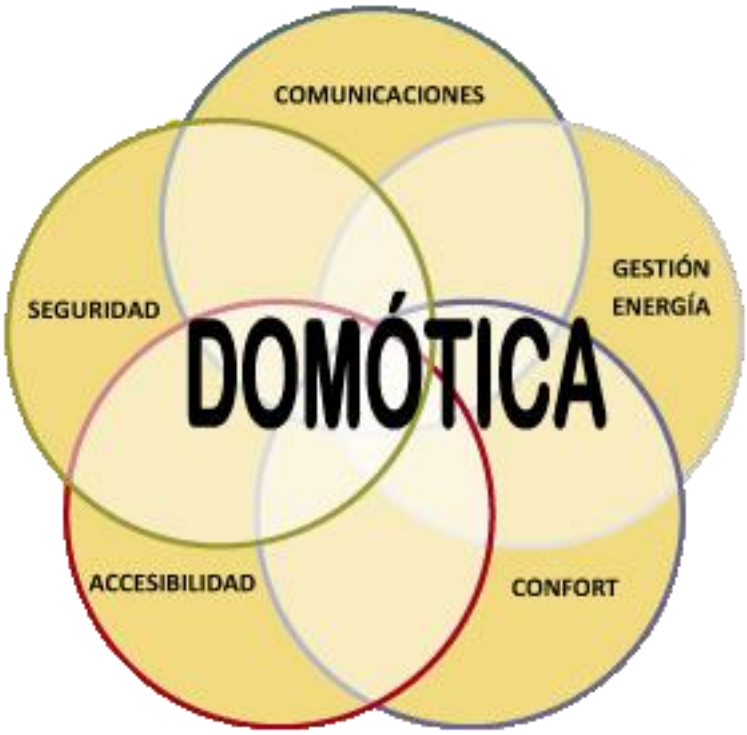

Figura 2-1. Propósitos de la domotica [4]

3 HOME I/O

Debido a la imposibilidad de disponer de una casa inteligente o de dispositivos domóticos reales, no queda más remedio que hacer uso del software HOME I/O el cual es capaz de simular una casa inteligente en tiempo real. Gracias a este software, podemos controlar cualquier sistema domótico de los implementados en la vivienda representada en dicho software.

3.1 ¿Qué es HOME I/O?

Home I/O es un software desarrollado por la empresa *Real Games* [5], dedicada al desarrollo de softwares de simulación. *Factory I/O* o *ITS PLC* completan algunos de los programas de esta empresa, la cual se encuentra extendida a lo largo de numerosos países y tiene la confianza de prestigiosas marcas a nivel mundial.

En la pagina web de Home I/O podemos encontrar toda la información al respecto. Tal y como dice su web, *'Home I/O es una simulación interactiva de una casa inteligente y el entorno circundante. Fue diseñado para cubrir una amplia gama de objetivos curriculares dentro de Ciencia, Tecnología, Ingeniería y Matemáticas (STEM). Con Home I/O, los estudiantes aprenderán sobre domótica, comportamiento térmico, eficiencia de consumo de energía y muchas otras materias que forman parte de la vida cotidiana.'* [6] siendo además *'El objetivo principal de Home I/O es introducir conceptos de automatización del hogar utilizando una casa interactiva inteligente. Equipado con los dispositivos de automatización más comunes, Home I/O desafía a los estudiantes a diseñar soluciones de control y comprender el impacto de su implementación.'* [6]. Es por tanto, un software con fines educativos o para que los usuarios puedan recrear o similar ciertos aspectos de su vivienda.

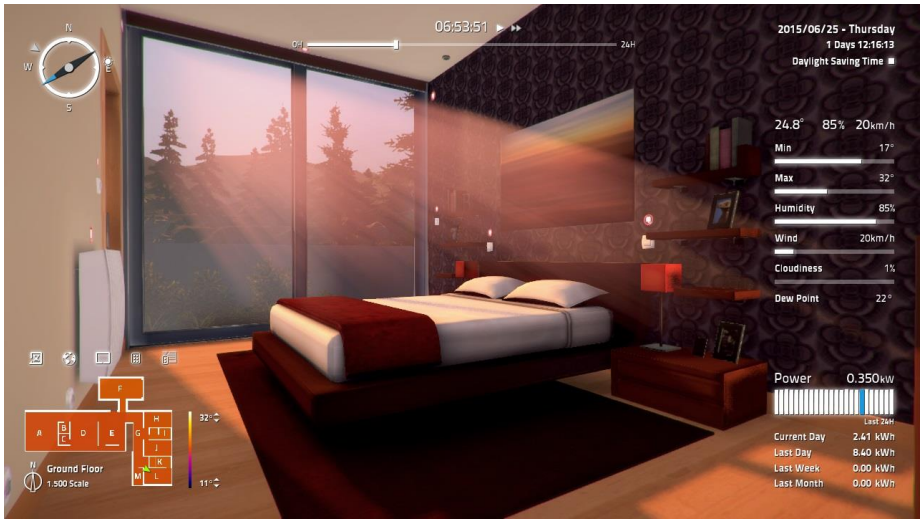

Figura 3-1. Interfaz Home I/O [6]

En esa misma web anteriormente señalada, es posible descargar el programa el cual está disponible en una version gratuita de prueba, que consta de 30 días en la que se pueden usar todas las funciones, o comprando una licencia del producto. Los tutores de este trabajo han facilitado una licencia del programa, sin embargo, no ha sido necesaria el uso de ella ya que con la version de prueba del software ha sido posible realizar este trabajo al completo.

3.2 Características disponibles en HOME I/O

En la web podemos encontrar toda la información del programa, la cual detalla que la vivienda a simular dispone de 174 dispositivos distintos que permiten la interacción con la iluminación o calefacción de la casa, dispone de una simulación en tiempo real del comportamiento térmico, dispone de entradas y salidas con valores digitales o analógicos, es posible cambiar las condiciones climáticas del entorno, contabiliza en tiempo real el consumo eléctrico y permite conectar o integrar otras tecnologías.

Debido a estas características, este software haría posible la necesidad de este trabajo de conectar los códigos creados con una vivienda inteligente, simulando en tiempo real lo que ocurriría en una vivienda domótica totalmente equipada. En el siguiente apratado se comprobará que, efectivamente, podemos usar este software a través de Python.

3.3 Conectividad con tecnologías externas

Home I/O ofrece una gran cantidad de opciones respecto a la interacción con los dispositivos. En el propio software, hay disponible una consola para automatizar la vivienda (una especie de tablet), la cual podemos configurar mediante distintas conexiones de los dispositivos o crear escenarios para así poder hacer uso de la simulación.

Además, *Real Games* incluye otro programa llamado *Connect I/O*, el cual aumenta esta posibilidad de interacción ya que puede usarse como una interfaz con tecnologías externas, permitiendo que se conecte fácilmente a PLC, microcontroladores o Modbus, entre muchas otras tecnologías. Adicionalmente, *Home I/O* se puede usar a través de Scratch 2, el cual es un lenguaje de programación.

Sin embargo, y lo interesante de *Home I/O* con relación a este trabajo es que también es posible su uso mediante Python. Más adelante se explicará al detalle como se puede realizar este proceso.

4 OBTENCIÓN DE CÓDIGOS

EL primer paso a realizar en este trabajo será la obtención de los código en el lenguaje de programación escogido para ello, Python, como se comentó en la introducción. De la misma forma, se procederá a buscar en internet los códigos relativos a la visión artificial que sean necesarios ya que la realización de los mismos no es objeto de este proyecto. Se realizará una adaptación de estos si fuera necesario o se crearán otros nuevos siguiendo los mismos patrones.

La visión artificial lleva varios años desarrollándose, por lo que se pueden encontrar distintos métodos, procedimientos o códigos para la obtención de un mismo objetivo. Se tratará de valorar las distintas opciones y escoger la más adecuada para el caso. De acuerdo con nuestro proyecto, serán necesarios códigos en tiempo real, es decir, que puedan analizar la información que se obtiene al momento de una webcam o cámara.

4.1 Módulos y librerías

Los módulos se definen como la porción de un programa, es decir, una pieza de código externa que se añade y aloja diferentes herramientas mientras que las librerías son un conjunto de implementaciones funcionales. [7]

En lo referente a la vision artificial, se hacen uso de distintos módulos y librerías, entre los cuales destacan los siguientes, no siendo estos los únicos a utilizar, pero sí los más importantes:

- Numpy: arrays y operaciones con matrices
- Scipy: librerías de alto nivel
- Matplotlib: visualización y gráficas
- OpenCV: librería para procesamiento de imágenes, vision y operaciones geométricas. Es una librería de código abierto
- Dlib: librería que contiene algoritmos de aprendizaje automático. Es también de código abierto

En el próximo apartado del trabajo se hablará sobre la instalación tanto del entorno que se ha utilizado en este caso, Anaconda, como de la instalación de estas librerías en dicho entorno, así como del uso de las mismas.

4.2 Códigos

En este apartado se comentará uno a uno los códigos necesarios, las dificultades que se han ido encontrando y de donde ha sido posible la obtención de estos. Se necesitarán códigos para el reconocimiento facial y de detección de gestos faciales.

4.2.1 Reconocimiento facial

Se pueden encontrar en la red numerosos códigos de detección facial, pudiéndose realizar mediante distintos métodos. Se recogen a continuación algunos de ellos.

4.2.1.1 Reconocimiento facial mediante haar cascade

Se trata de entrenar la función cascada a partir de muchas imágenes positivas o negativas para luego detectar esos objetos entrenados en la imagen a analizar. Es un método muy sencillo de aplicar y que apenas consume recursos en la máquina, ya que afortunadamente disponemos de una gran cantidad de archivos que implementan el entrenamiento de distintas regiones de la cara o la totalidad de la misma, sin embargo, no será el método a usar debido a que no es el que mejor funciona.

4.2.1.2 Reconocimiento facial usando la librería DLIB

La librería Dlib posee una función ya implementada que permite obtener los rostros de la imagen, *dlib.get_frontal_face_detector()*. Gracias a su facilidad de implementación junto con el resto de códigos, es el que será usado para ello. Funciona bien aunque no realiza el tracking de la cara.

4.2.1.3 Reconocimiento facial con tracking

Si finalmente se quisiera realizar el tracking del rostro, se podría usar el código de Adrian Rosebrock, el cual funciona a la perfección pero que requiere de un gran consumo de recursos del ordenador, es por ello por lo que no se utilizará en este proyecto.

4.2.2 Detector de marcas faciales

Es este uno de los principales códigos del proyecto, ya que en base a este se obtendrán los puntos de interés del rostro que servirán para el resto de códigos. Mediante la librería Dlib se puede obtener un total de 68 puntos de la cara, incluyendo así puntos en las cejas, ojos, nariz y labios.

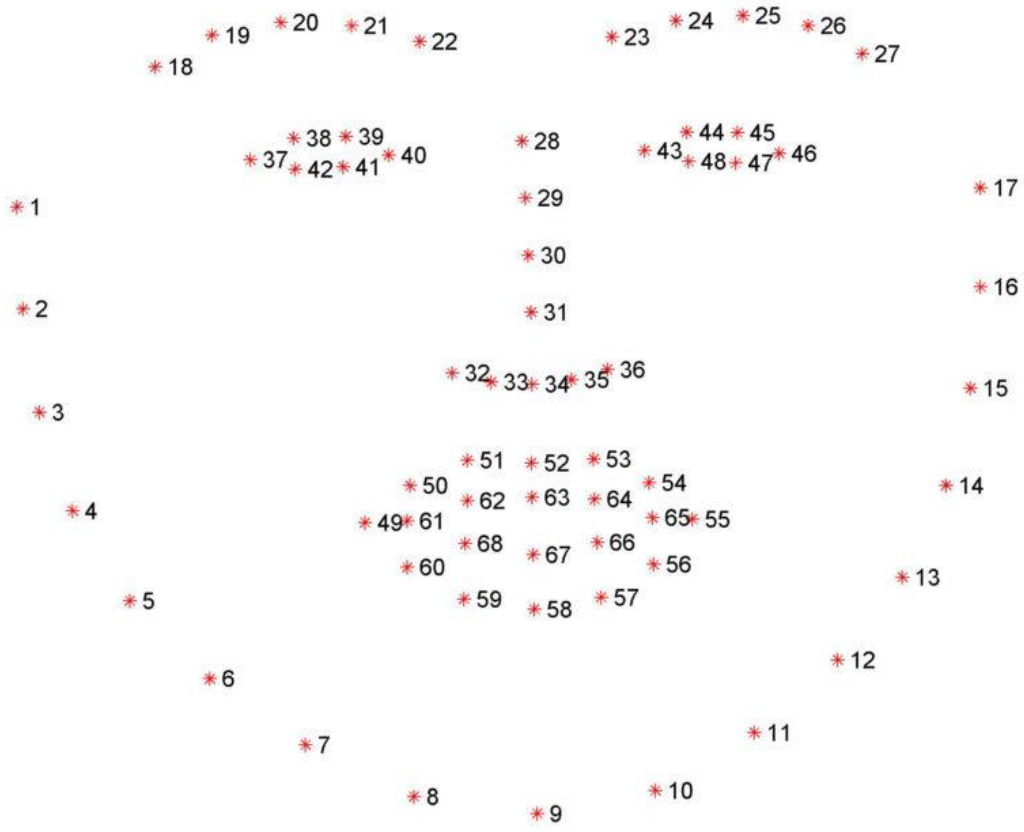

Figura 4-1. Puntos faciales

Para implementar esta detección de marcas faciales, se ha usado el código de Adrian Rosebrock, Facial landmarks with dlib, OpenCV, and Python [8]. Se han realizado unos pequeños cambios para adaptar el código a nuestro proyecto. Como se ha comentado en el punto anterior, para la detección facial se ha usado la función *dlib.get_frontal_face_detector()*. Con unas buenas condiciones de luz en la habitación, el código funciona perfectamente. No se realiza el tracking de los puntos ya que implementarlo requeriría de una gran carga de recursos en el ordenador.

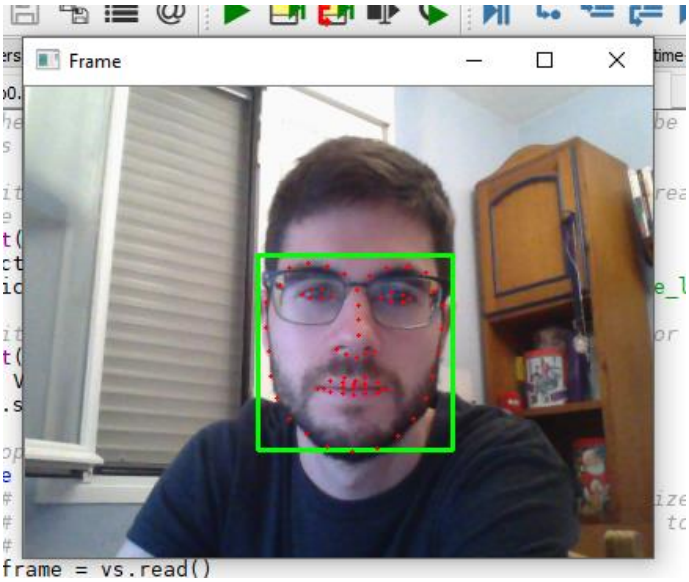

Figura 4-2. Reconocimiento de puntos faciales sobre el rostro

4.2.3 Detección parpadeo

Se usará un código creado por Adrian Rosebrock, Eye blink detection with OpenCV, Python, and dlib [9], el cual calcula la relación de aspecto de cada ojo, y cuando estos son menores de un cierto valor, detecta el parpadeo. Para obtener los puntos de los ojos ha sido necesario usar el código anterior que nos permitía calcular los puntos del rostro.

Se ha intentado modificar este código de forma que además de detectar el parpadeo, detecte cuando se guiñe el ojo derecho o el ojo izquierdo. Sin embargo, el código obtenido no es demasiado robusto, no detectando algunos de los guiños en algunas ocasiones. A la hora de la implementación con *Home I/O* se volverá a estudiar este código por si fuera posible la integración robusta del mismo.

4.2.4 Detección abrir la boca

Siguiendo el mismo método de la detección de parpadeo, se ha realizado este otro código que detecta cuando el usuario abre la boca. De la misma forma que en el código anterior, mediante el detector de marcas faciales se obtienen los distintos puntos de la boca, y cuando la relacion de aspecto se hace más grande que un determinado valor, el programa detecta que la boca se ha abierto. Esta relación de aspecto, al ser un número mucho mayor que la del ojo, hace que el programa sea mucho más robusto y funcione casi a la perfección. El usuario puede hablar delante de la cámara que el programa, hasta que no abra la boca de forma considerable, no va a detectar que esta se ha abierto.

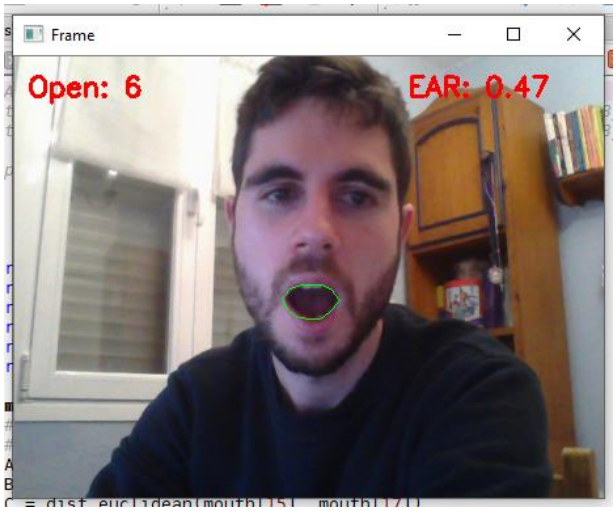

Figura 4-3. Detección abrir la boca

4.2.5 Levantar las cejas

Siguiendo el mismo procedimiento, se hace exactamente lo mismo pero con los puntos de la ceja y ojo, de forma que cuando se levantan las cejas la relación de aspecto aumenta y el código detecta que se han levantado las cejas. No es del todo robusto, fallando a veces cuando se cierran los ojos, es por ello por lo que se incluye en el código la condición de que los ojos se deben mantener abiertos, aumentando así la robustez del mismo.

Figura 4-4. Detección levantar las cejas

5 CONEXIÓN DE PYTHON A HOME I/O

En este capítulo vamos a describir el último paso del trabajo realizado, el cual, como su nombre indica, consiste en la conexión de Python de los códigos obtenidos con el software *Home I/O*. En definitiva, se pretende actuar sobre *Home I/O* realizando los distintos gestos faciales delante de la cámara. Se explicará detalladamente como llegar a realizar este proceso, desde la instalación del entorno Anaconda hasta el establecimiento de dicha conexión. Al final de este capítulo, cualquier persona que desee poner en práctica este trabajo podrá ser capaz de hacerlo sin ningún problema siguiendo las siguientes indicaciones.

5.1 Entorno Anaconda

A lo largo de este trabajo se ha mencionado que el entorno que se ha utilzado para llevar a cabo dicho proyecto ha sido *Anaconda* [10], pero no se ha hablado más sobre ello, es por eso que este subapartado va dedicado a *Anaconda*.

De acuerdo con *Wikipedia*, *'Anaconda es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático (machine learning). Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue y administración de los paquetes de software. Las diferentes versiones de los paquetes se administran mediante el sistema de gestión de paquetes conda, el cual lo hace bastante sencillo de instalar, correr, y actualizar software de ciencia de datos y aprendizaje automático como Scikit-learn, TensorFlow y SciPy'* [11].

La elección del entorno *Anaconda* se debe principalmente a lo comentado en las líneas anteriores, ya que se puede instalar de forma sencilla tanto el entorno como las librerías a utilizar gracias al sistema de gestión de paquetes conda o pip.

Antes de comenzar con la instalación de *Anaconda*, es conveniente hacer referencia a dos datos inportantes en el momento en el que se ha realizado este trabajo. En primer lugar, y unos pocos meses antes de la finalización del mismo, salió a la luz una última version del entorno, la cual, en ese momento, tenía algunos fallos en el sistema de gestión de paquetes anteriormente comentado. Además, la librería Dlib, que será de vital importancia para este trabajo, requiere de un proceso complicado de instalación en Python 3.7 (la versión de Python que incluye la última versión de Anaconda disponible en la finalización de este trabajo), ya que no es posible instalar dicha librería mediante conda ni pip y hay que hacerlo de manera manual.

Es por ello que usaremos una versión más antigua del entorno. Finalmente, y no menos importante, *Home I/O* es un software de 32 bits, por lo que tendremos que instalar la versión correspondiente de *Anaconda* de 32 bits. Teniendo en cuenta todo lo dicho, procedemos a instalar una versión adecuada del entorno. En este repositorio están todas las versiones disponibles, en mi caso, he decidido instalar la versión Anaconda3-5.2.0-Windows-x86.exe que se corresponde a la versión de 32 Bits y usa Python 3.6. La instalación del programa se realiza de forma similar a cualquier otro software y no tiene gran complejidad. Una vez instalado *Anaconda*, se usará *Spyder* para la escritura y ejecución de los códigos.

5.2 Instalando las librerías

Las librerías necesarias se instalarán a través de Anaconda Prompt mediante los comandos que se especificarán a continuación, haciendo uso de los gestores de paquetes conda y pip. Bastará simplemente con escribir los comandos mostrados a continuación y pulsar Enter para ejecutarlos e iniciar la instalación. En algunos de los procesos, además, se pedirá que se confirme la instalación.

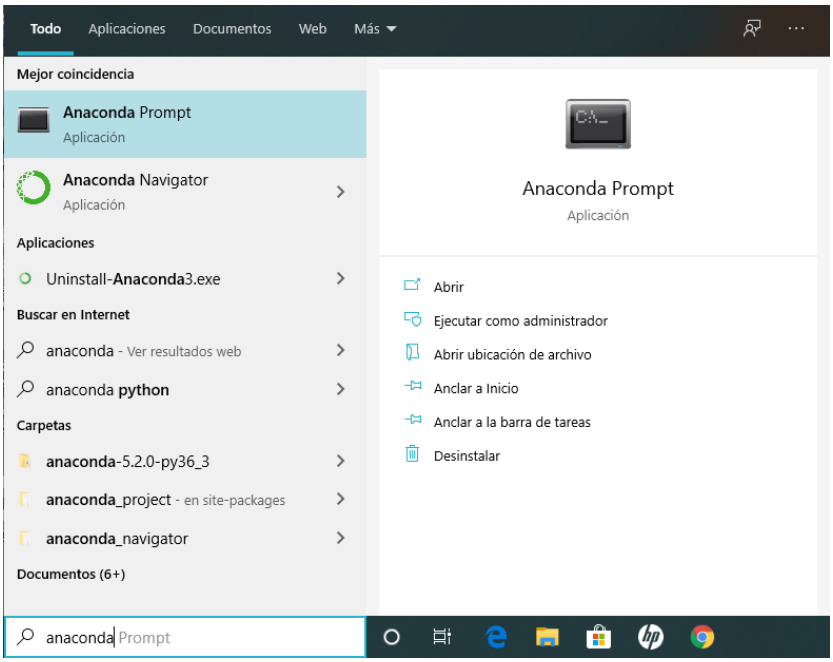

Figura 5-1. Anaconda Prompt

Una vez abierto Anaconda Prompt, se escriben lo siguientes comandos para instalar cada una de las librerías:

- LibreríaOpenCV: Se escribe `pip install opencv-python`. Automáticamente comienza la descarga e instalación de la librería.

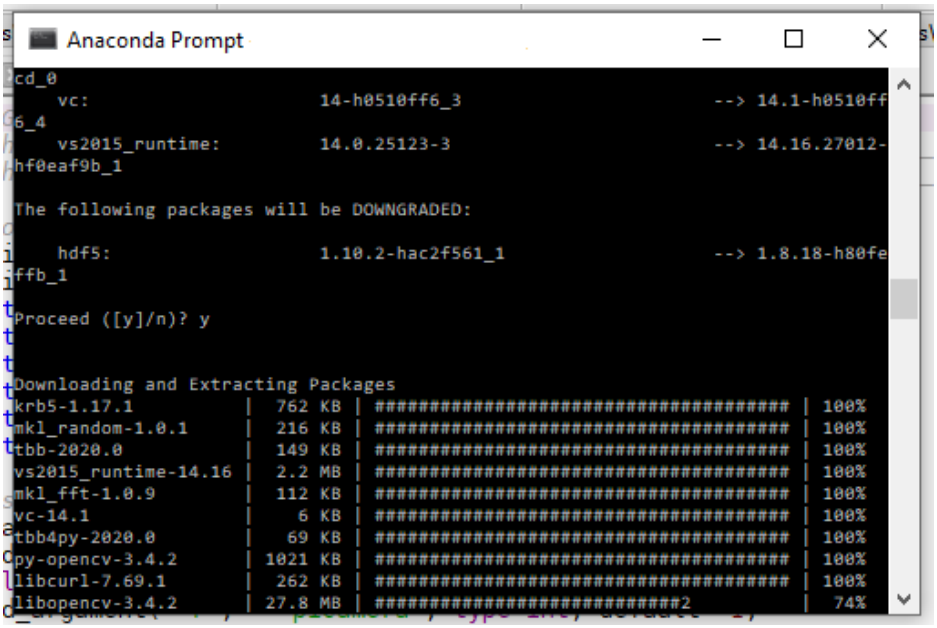

Figura 5-2. Instalación OpenCV

- Librería imutils: De igual forma, se escribe `pip install imutils`.
- Librería Dlib: Para instalar esta librería basta con escribir `conda install -c conda-forge dlib`. En el caso que tuviéramos Python 3.7, a fecha de creación de este trabajo, sería necesario instalar Dlib de forma manual. Es por ello el motivo de usar una versión anterior de Python, para así poder instalar Dlib más fácilmente.

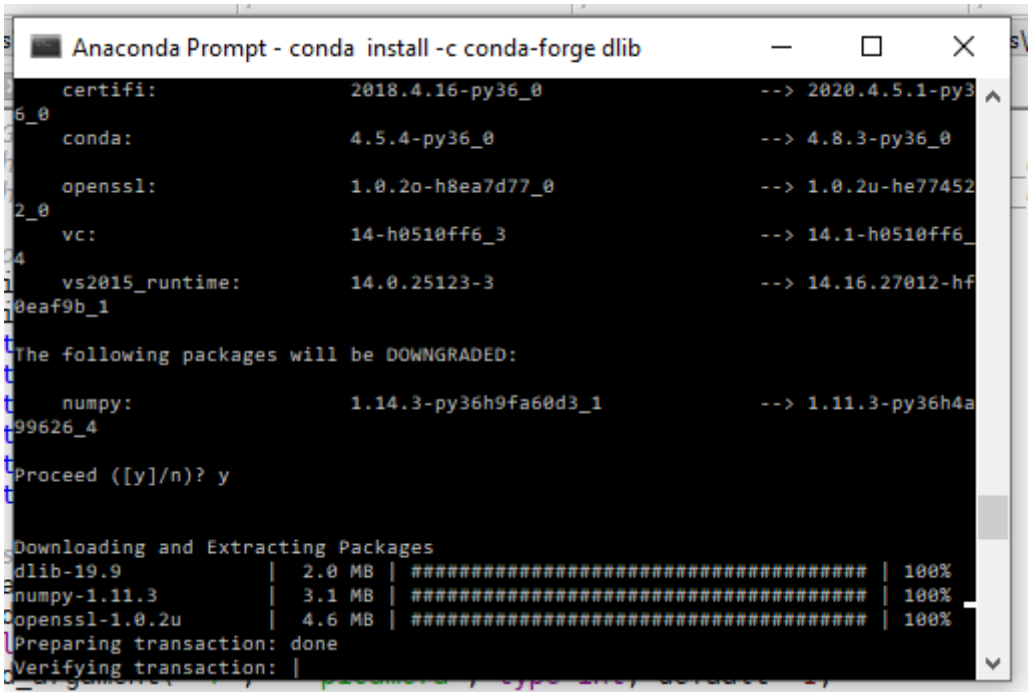

Figura 5-3. Instalación Dlib

- Librería Pythonnet: Esta es necesaria para ejecutar el código que conecta con *Home I/O*. Para instalarla, basta con escribir `pip install pythonnet`.
- Librería Engine IO: Por último, y también necesaria para usar *Home I/O*, se instala esta librería haciendo uso del comando `pip install python-engineio`.

De esta forma, y llegados a este punto, se tendrán el entorno *Anaconda* y las librerías necesarias totalmente instaladas y listas para ejecutar los códigos.

5.3 Integración de Python con Home I/O

De la página web de *Home I/O*, se tiene que *'Home I/O se puede usar junto con Python 3. La integración de Python le permite escribir programas de Python para interactuar con dispositivos de E / S domésticos. Esto significa que puede leer y escribir en dispositivos de E / S desde programas Python.'* [12].

Para llevar a cabo esto, en esa misma web se puede descargar un archivo que incluye lo necesario para realizar este procedimiento. A su vez, viene un código de ejemplo en Python en el que se apaga y enciende una luz del salón 5 veces. El archivo que se descarga dispone de distintas carpetas, en función de la versión de Python que se vaya a utilizar. En mi caso, usaré los archivos correspondientes a Python 3.6.

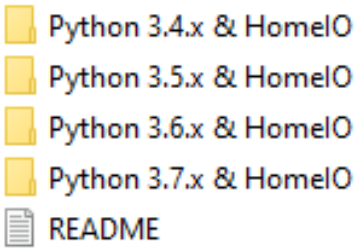

Figura 5-4. Contenido del archivo que se descarga de la web

Es conveniente señalar que, en este aspecto, *Real Games* y *Home I/O* facilitan mucho el trabajo ya que en su web explcan todo a la perfección además de facilitar los archivos necesarios. Es por ello por lo que se recomienda leer la ayuda que se ofrece en la web.

5.4 Dispositivos Home I/O

Antes de empezar a escribir el código sería conveniente presenter los distintos modos y dispositivos de los que consta la vivienda virtual para así poder interactuar con ellos correctamente. Como se viene diciendo a lo largo de este documento, toda la información se encuentra detallada en la web de ayuda de *Real Games* y *Home I/O*. [6]

5.4.1 Modos de los dispositivos

Todos los dispositivos se pueden usar en tres modos diferentes:

- Wired Mode: modo cableado (de color rojo). La casa no se encuentra automatizada, siendo la instalación eléctrica estándar.
- Wireless Mode: modo inalámbrico (de color verde). Se utiliza con la consola de automatización del hogar pudiendose así programar para automatizar la casa.
- External Mode: modo externo (de color azul). Permite acceder a los puntos de E / S de cada dispositivo con Connect I / O, a través del SDK o Python

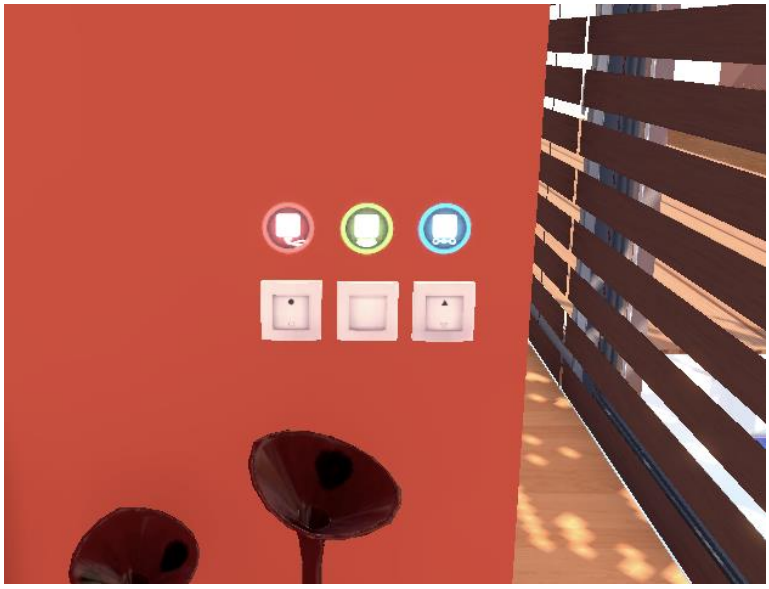

Figura 5-5. Modos de los dispositivos

Por tanto, para hacer funcionar los dispositivos domóticos de la vivienda a través de Python es necesario que los dispositivos de la vivienda virtual sean puestos en modo externo.

5.4.2 Dispositivos

En *Home I/O* hay disponibles un total de 174 dispositivos entre luces, interruptores de luz, persianas e interruptores de subida y bajada, calentadores y termostatos, distintos tipos de detectores y otros dispositivos como sirenas o la puerta del garaje. En este trabajo se tendrán en cuentas aquellos dispositivos que se consideran de uso imprescindible en una vivienda como pueden ser las luces o las persianas. Sin embargo, se podría realizar igualmente para automatizar la puerta del garaje, por ejemplo, sin ningún problema.

De igual forma, se ha decidido interactuar con los dispositivos de forma Sí o No, por ejemplo con las luces, o encendidas o apagadas, a pesar de que el software permite la regulación de la intensidad de las mismas. Sin embargo, no se ha decidido hacer uso de ello ya que se considera que por el tipo de problema a resolver, controlar la intensidad de las luces no es del todo necesario o imprescindible. Es por ello que todos los datos sobre los que se actuan en Python son de tipo Bool, es decir, 0 o 1.

5.4.2.1 Luces

A continuación, se muestra un mapa con la posición de las luces de la vivienda, así como la dirección de memoria sobre las que hay que actuar en Python para poder tener el control de las mismas.

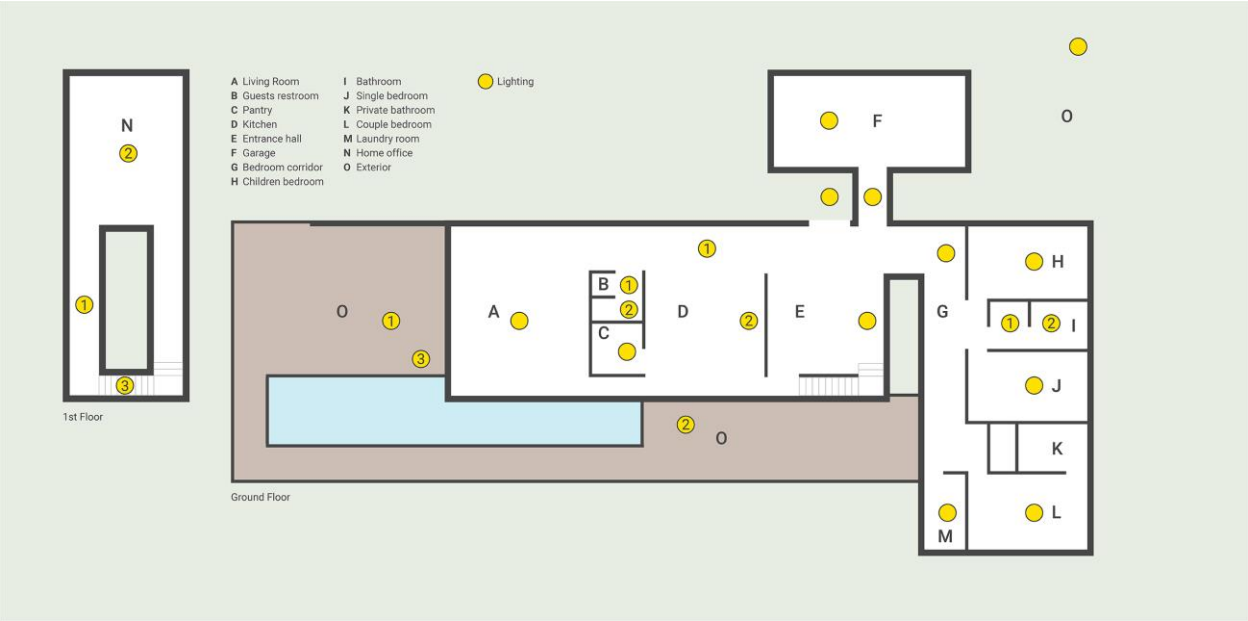

Figura 5-6. Mapa con las luces de la vivienda [13]

Siendo las direcciones de memoria de cada una de ellas las siguientes:

Tabla 5-1. Direcciones de memoria de las luces [14]

Nombre (Zona)	Dirección
Living Room (A)	0
Guests Restroom 1 (B)	19
Guests Restroom 2 (B)	20
Pantry (C)	30
Kitchen 1 (D)	40
Kitchen 2 (D)	41
Entrance Hall (E)	54
Garage 1 (F)	68
Garage 2 (F)	69
Bedroom Corridor (G)	83
Children Bedroom (H)	97
Bathroom 1 (I)	110
Bathroom 2 (I)	111
Single Bedroom (J)	122
Private Bathroom (K)	135
Couple Bedroom (L)	146
Laundry Room (M)	159
Home Office 1 (N)	172
Home Office 2 (N)	173

Home Office 1 (N)	174
Exterior Porch 1 (O)	187
Exterior Porch 2 (O)	188
Exterior Pool (O)	189
Exterior Garden (O)	190
Exterior Entrance (O)	191

Se han escrito todas las direcciones de memoria de las luces ya que se usarán todas en el código.

5.4.2.2 Persianas

De igual forma, se muestra el mapa de las persianas de la vivienda. En este caso, sin embargo, solo se harán uso de las persianas del salón ya que han sido las consideradas a tener en cuenta en el ejemplo que se está realizando. Si se quisieran utilizar otras persianas, bastaría con buscar las direcciones de memoria de cualquier otra persiana e incluirla al código.

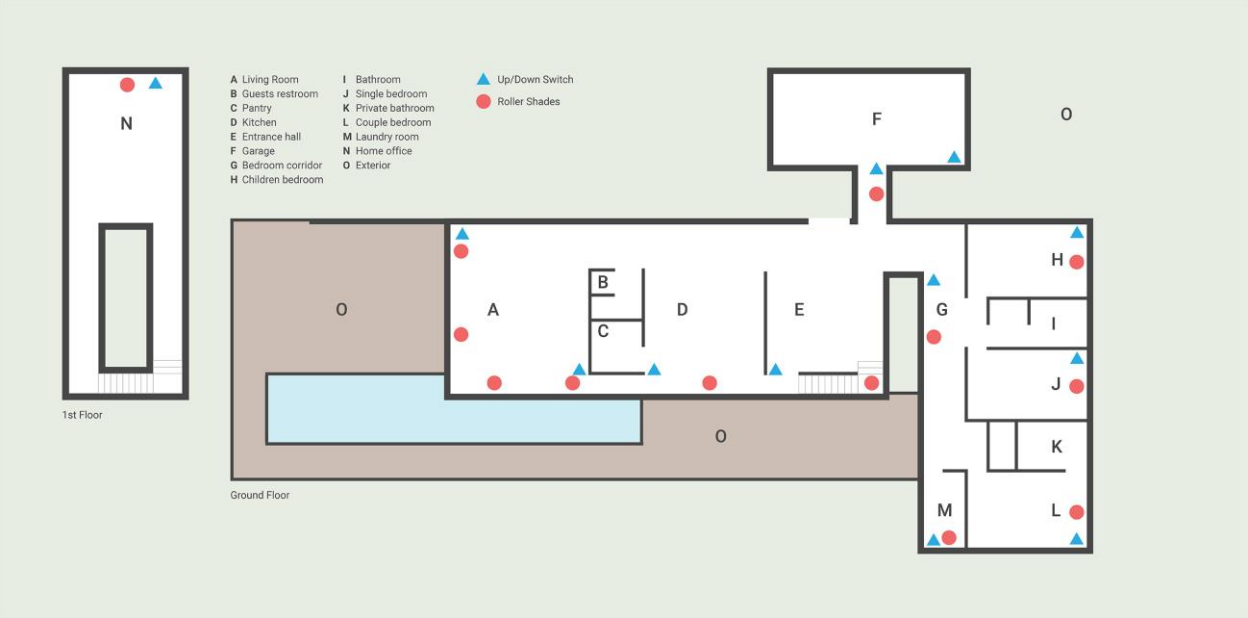

Figura 5-7. Mapa con las persianas de la vivienda [13]

Y, de igual forma que se hizo con las luces, las direcciones de memoria de las persianas a usar (zona A):

Tabla 5-2. Direcciones de memoria de las persianas [14]

Nombre (zona)	Dirección
Roller Shades 1 Up (A)	1
Roller Shades 1 Down (A)	2
Roller Shades 2 Up (A)	3
Roller Shades 2 Down (A)	4
Roller Shades 3 Up (A)	5
Roller Shades 3 Down (A)	6
Roller Shades 4 Up (A)	7

Roller Shades 4 Down (A)	8

En este caso, para cada persiana, hay dos direcciones, cada una de ellas correspondientes al motor de subida o de bajada de la persiana.

5.4.2.3 Otros dispositivos

No se han tenido en cuenta en este trabajo otros dispositivos, sin embargo se podrían usar de igual manera. Por ejemplo, se podría leer la información de los sensores de las puertas para que el usuario, mediante un gesto, pudiera comprobar si todas las puertas de la vivienda están cerradas. O que se pudiera activar la alarma mediante alguna combinación de gestos si el usuario se encuentra en una situación de peligro.

5.5 Escribir el código

Finalmente, y una vez introducido todo el proceso de instalación y la información acerca de los dispositivos a utilizar, bastaría con crear el código que controle los dispositivos domóticos de la vivienda virtual de *Home I/O*. Para ello, se utilizarán los códigos anteriormente presentados que detectan los gestos faciales. Sin embargo, estos códigos serán adaptados a las nuevas necesidades, como por ejemplo, que en vez de contar cuantas veces se parpadea, este se active cuando se parpadee 3 veces seguidas. Esto se debe a que anteriormente se presentaron los códigos de forma general y ahora se aprovecharán dichos códigos para nuestras necesidades.

Además, añadir que a continuación se propondrá una alternativa de código a modo de ejemplo, sin embargo, este puede ser totalmente modificado y personalizado al gusto del usuario.

5.5.1 Código principal

En este ejemplo, se propondrá un código con el que realizarán pequeñas tareas de la casa centradas en el salón, descritas a continuación:

- Encender la luz del salón parpadenado 3 veces seguidas en un rango de 2 segundos.
- Subir las cuatro persianas del salón abriendo la boca. Si mientras que se están subiendo las persianas abrimos de nuevo la boca, las persianas se quedarán paradas en el punto en el que se encuentren.
- Bajar las cuatro persianas del salón levantando las cejas. Si mientras que se están bajando las persianas levantamos de nuevo las cejas, las persianas se quedarán paradas en el punto en el que se encuentren.
- Apagar todas las luces de la vivienda manteniendo los ojos cerrados durante un segundo.

Como se puede comprobar, la idea de este código es la de que el usuario tenga un control sobre los dispositivos del salón. La elección de esta parte de la casa se debe a que se ha supuesto que el salón es la zona más utilizada por el usuario. En caso, por ejemplo, de que el usuario no pueda salir de la cama y se encuentre siempre en su habitación pues se podría usar exactamente este código pero cambiando las direcciones de los dispositivos utilizados por las que fueran necesarias.

El código completo se encuentra en el anexo de este documento, aunque aquí se explicarán las partes del mismo.

En primer lugar, debemos importar todas las librerías necesarias:

```
1.  import sys
2.  import time
3.  import clr
4.  from scipy.spatial import distance as dist
5.  from imutils.video import FileVideoStream
6.  from imutils.video import VideoStream
7.  from imutils import face_utils
8.  import numpy as np
9.  import imutils
10. import dlib
```

```
11. import cv2
12.
13. clr.AddReference('EngineIO')
14.
15. from EngineIO import *
```

Alguna de ellas no se llegan a usar pero se han incluido porque son habituales en este tipo de códigos y quizás en una nueva actualización del código sean necesarias.

Se definen las funciones que calculan la relación de aspecto, ya que gracias a este cálculo el código determinará cuando se ha producido un gesto:

```
16. def eye_aspect_ratio(eye):
17.     # compute the euclidean distances between the two sets of
18.     # vertical eye landmarks (x, y)-coordinates
19.     A = dist.euclidean(eye[1], eye[5])
20.     B = dist.euclidean(eye[2], eye[4])
21.
22.     # compute the euclidean distance between the horizontal
23.     # eye landmark (x, y)-coordinates
24.     C = dist.euclidean(eye[0], eye[3])
25.
26.     # compute the eye aspect ratio
27.     ear = (A + B) / (2.0 * C)
28.
29.     # return the eye aspect ratio
30.     return ear
```

Esta función calcula la relación que hay entre la distancia vertical de los puntos del ojo y la distancia horizontal. De esta forma, esta relación se hará muy pequeña cuando la distancia vertical sea muy pequeña (cuando se cierren los ojos). Este mismo procedimiento se hace para detectar cuando se abre la boca pero al contrario, es decir, el código detectará que se ha abierto la boca cuando esta relación sea más grande de lo normal. Para detectar cuando se levantan las cejas se usa el mismo procedimiento, pero en este caso las relaciones se establecen entre los puntos de la ceja y los ojos. En el código incluido en el anexo se encuentran las otras dos funciones, que por simplicidad, aquí se han omitido ya que son similares.

Posteriormente se deben definir las constantes que se deben usar en el código, las cuales corresponden a los valores de la relación de aspecto, así como los frames:

```
31. # define constants for the aspect ratio
32. EYE_AR_THRESH = 0.3
33. AR_CONSEC_FRAMES = 3
34. EYE_SECONDS_AR_CONSEC_FRAMES = 30
35. MOUTH_AR_THRESH = 0.6
36. EYEBROW_AR_THRESH = 0.6
```

Además de inicializar los contadores que serán usados:

```
37. # initialize the frame counters and the total number of open
38. COUNTER_EYE = 0
39. COUNTER_MOUTH = 0
40. COUNTER_EYEBROW = 0
41. TOTAL_EYE = 0
```

Se procede a inicializar el detector facial de Dlib, el cual detectará las caras en la imagen además de predecir las marcas faciales en las mismas. El archivo *'shape_predictor_68_face_landmarks.dat'* es el archivo facilitado por la librería Dlib y el cual es el resultado de los procedimientos de inteligencia artificial resueltos por esta librería para poder obtener dichos puntos. Este archivo debe encontrarse en el mismo directorio donde se guarde el código de Python:

```
42. # initialize dlib's face detector (HOG-based) and then create
```

```
43. # the facial landmark predictor
44. print("[INFO] loading facial landmark predictor...")
45. detector = dlib.get_frontal_face_detector()
46. predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

Obtenemos la direcciones de los dispositivos de *Home I/O*:

```
47. #Recibo la dirección que quiero modificar en HOME I/O
48. # Luces zona A
49. livingRoomLight = MemoryMap.Instance.GetBit(0, MemoryType.Output)
50. # Luces zona B
51. GuestsRestroom1Light = MemoryMap.Instance.GetBit(19, MemoryType.Output)
52. GuestsRestroom2Light = MemoryMap.Instance.GetBit(20, MemoryType.Output)
```

En el código del anexo aparece la obtención de todas las direcciones al completo, al igual que las direcciones de las persianas que se hacen de forma análoga.

Guardamos los puntos de referencia de las zonas de la cara que interesan:

```
53. # grab the indexes of the facial landmarks for eyes, the mouth
54. # and the eyebrowns
55. (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
56. (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
57. (MStart, MEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
58. (lebStart, lebEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eyebrow"]
59. (rebStart, rebEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eyebrow"]
```

La librería Dlib tiene descritos los puntos de cada parte de la cara, es por ello que escribiendo ‘mouth’, por ejemplos, obtenemos los puntos de la boca.

Se pone en marcha la webcam. En caso de usar una cámara externa, se debería usar el código comentado:

```
60. # start the video stream thread
61. print("[INFO] starting video stream thread...")
62. #vs = VideoStream().start()
63. #fileStream = True
64. vs = VideoStream(src=0).start()
65. # vs = VideoStream(usePiCamera=True).start()
66. # fileStream = False
```

Mientras que se esté usando la cámara, el código se mantendrá en este bucle. La webcam se lee frame a frame y se opera sobre cada uno de ellos:

```
67. # loop over frames from the video stream
68. while True:
69.     # if this is a file video stream, then we need to check if
70.     # there any more frames left in the buffer to process
71.     #if fileStream and not vs.more():
72.     #   break
73.
74.     # grab the frame from the threaded video file stream, resize
75.     # it, and convert it to grayscale
76.     # channels)
77.     frame = vs.read()
78.     frame = imutils.resize(frame, width=450)
79.     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

En cada uno de dichos frames, se detectan los rostros gracias al detector antes definido de la librería Dlib:

```
80.     # detect faces in the grayscale frame
81.     rects = detector(gray, 0)
```

Y para cada uno de los rostros detectados, se hacen las operaciones que se encuentren dentro del siguiente bucle for. Es por ello, que si en la imagen aparecen dos personas, el código puede detectar los gestos de ambas personas. Para empezar, lo primero que se hace en este bucle for es detectar las marcas faciales de cada rostro encontrado en la imagen:

```
82.    # loop over the face detections
83.     for rect in rects:
84.         # determine the facial landmarks for the face region, then
85.         # convert the facial landmark (x, y)-coordinates to a NumPy
86.         # array
87.         shape = predictor(gray, rect)
88.         shape = face_utils.shape_to_np(shape)
```

Se establecen las coordenadas de cada punto que se necesita y, mediante las funciones arriba definidas, se calculan las relaciones de aspecto:

```
89.         # extract the left and right eye coordinates, then use the
90.         # coordinates to compute the eye aspect ratio for both eyes
91.         leftEye = shape[lStart:lEnd]
92.         rightEye = shape[rStart:rEnd]
93.         leftEAR = eye_aspect_ratio(leftEye)
94.         rightEAR = eye_aspect_ratio(rightEye)
95.         # average the eye aspect ratio together for both eyes
96.         ear = (leftEAR + rightEAR) / 2.0
```

De igual forma que antes, por simplicidad, aquí solo se ha escrito la parte correspondiente a los ojos. Para la boca y las cejas se haría de forma análoga. Se recuerda que el código completo aparece en el anexo de este documento.

Se calcula el contorno de los ojos y se dibuja en cada frame del video que se mostrará por pantalla:

```
97.         # compute the convex hull for the left and right eye, then
98.         # visualize each of the eyes
99.         leftEyeHull = cv2.convexHull(leftEye)
100.                rightEyeHull = cv2.convexHull(rightEye)
101.                cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
102.                cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
```

De forma similar se realizaría para la boca y cejas.

Llegado a este punto, se comenzarían a calcular la detección de cada uno de los gestos. En primer lugar, se calculan los gestos de parpadear tres veces seguidas en un rango de dos segundos y mantener los ojos cerrados durante un segundo:

```
103.                # Detección parpadear
104.                # check to see if the eye aspect ratio is below the blink
105.                # threshold, and if so, increment the blink frame counter
106.                if ear < EYE_AR_THRESH:
107.                    COUNTER_EYE += 1
108.
109.                # otherwise, the eye aspect ratio is not below the blink
110.                # threshold
111.                else:
112.                    # if the eyes were closed for a sufficient number of
113.                    # then increment the total number of blinks
114.                    if COUNTER_EYE >= AR_CONSEC_FRAMES and COUNTER_EYE <= EYE_SECONDS_AR
    _CONSEC_FRAMES:
115.                        TOTAL_EYE += 1
116.                        if TOTAL_EYE == 1:
117.                            tiempo_EYE = time.time() + 2
118.                        if TOTAL_EYE >= 3 and time.time() <= tiempo_EYE:
119.                            livingRoomLight.Value = not livingRoomLight.Value
```

```
120.                            # When using a memory value before calling the Update method
     we are using a cached value.
121.                            print("Light is on? " + str(livingRoomLight.Value))
122.                            TOTAL_EYE = 0
123.                            # Calling the Update method will write the livingRoomLight.V
    alue to the memory map.
124.                            MemoryMap.Instance.Update()
125.                        elif time.time() >= tiempo_EYE:
126.                            TOTAL_EYE = 1
127.                            tiempo_EYE = time.time() + 2
128.                    elif COUNTER_EYE >= EYE_SECONDS_AR_CONSEC_FRAMES:
129.                        # Se apagan todas las luces
130.                        livingRoomLight.Value = 0
131.                        GuestsRestroom1Light.Value = 0
132.                        GuestsRestroom2Light.Value = 0
133.                         # Se incluyen el resto de variables
134.                        # When using a memory value before calling the Update method we
    are using a cached value.
135.                        print("Light is on? " + str(livingRoomLight.Value))

136.                        # Calling the Update method will write to the memory map.
137.                        MemoryMap.Instance.Update()
138.
139.
140.                    # reset the eye frame counter
141.                    COUNTER_EYE = 0
```

Cuando en un frame se detecta que la relación de aspecto es más pequeña que el valor de 0.3 propuesto, se cuenta un frame. Cuando este valor vuelve a aumentar, se comprueba durante cuantos frames ha ocurrido y si ha sido en al menos 3 frames consecutivos y durante menos de 30 frames (que se corresponde a un segundo, ya que la webcam graba a 30 fps), se detecta que el usuario ha parpadeado. Si en los próximos 2 segundos, se detecta que el usuario ha parpadeado 3 veces, el código detecta el gesto y cambia el valor que tenga la variable de la luz del salón. Si la luz estaba apagada, la enciende, y si estaba encendida, la apaga. Es necesario actualizar las variables para que *Home I/O* reconozca este cambio, acción que se realiza en la línea número 124. Por el contrario, si el código detecta que se ha tenido los ojos cerrados durante 30 frames o más, apaga todas las luces de la casa poniendo todas las variables a 0. En el código completo aparecen el resto de variables que aquí se han omitido.

La detección de los gestos de abrir la boca y levantar las cejas se realiza de forma similar y análoga a este procedimiento.

Ya llegando al final del código, se muestra por pantalla el video y se establece la forma de apagar la cámara y salir del bucle while:

```
142.            # show the frame
143.            cv2.imshow("Frame", frame)
144.            key = cv2.waitKey(1) & 0xFF
145.
146.            # if the `q` key was pressed, break from the loop
147.            if key == ord("q"):
148.                break
```

Y para concluir, se cierran corecctamente todos los procesos abiertos:

```
149.        # When we no longer need the MemoryMap we should call the Dispose method to rele
    ase all the allocated resources.
150.        MemoryMap.Instance.Dispose()
151.
152.        print("Bye!")
153.
154.        # do a bit of cleanup
155.        cv2.destroyAllWindows()
156.        vs.stop()
```

5.5.1.1 Conclusiones del código propuesto

El código funciona bastante bien y cumple con el cometido a la perfección. Sin embargo, los valores de las relaciones de aspecto podrían ser modificados en función del usuario para que el código se adapte mejor a cada persona. Puede ser una opción muy buena para tener un control de las luces y las persianas de una habitación y para que, por ejemplo, antes de dormir o salir de casa se puedan apagar todas las luces de la vivienda mediante un solo gesto.

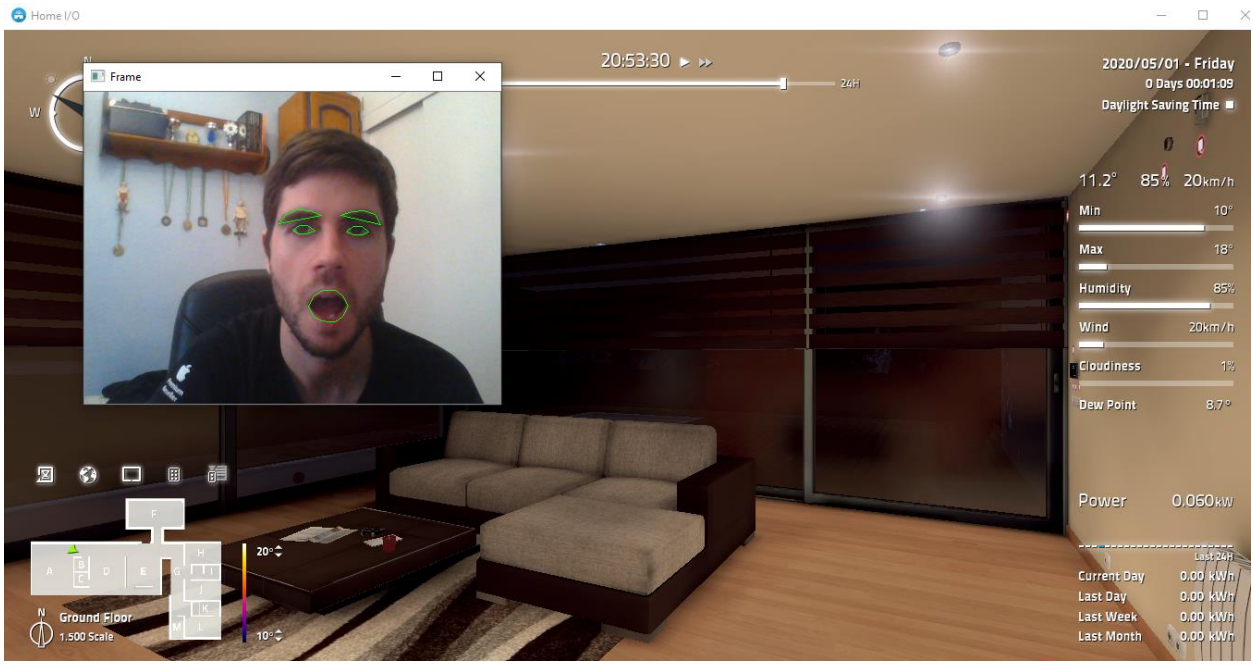

Figura 5-8. Código funcionando

Aún así, el control de los dispositivos de la vivienda sigue siendo muy simple debido a la única utilización de cuatro gestos. En este ejemplo no se ha llegado a utilizar la opción de guiñar los ojos debido a la poca robustez del mismo, sin embargo, si se pudiera pulir dicho código se podría implementar otras dos acciones. Aún así, si el usuario al que va destinado el programa no es capaz de guiñar los ojos correctamente, nos encontraríamos en la misma situación.

Como propuesta, se podría intentar implementar combinaciones de gestos, aumentando así las posibilidades de las acciones.

5.5.2 Condiciones en las que se ha probado el código

El código anterior ha sido probado y ejecutado en un ordenador portatil, usando la webcam integrada del mismo. Es por ello, y debido a la poca calidad de la misma, que para que el código funcione correctamente, la habitación en la que se pruebe debe tener buena luz además de evitar situaciones de contraluz o en las que el usuario salga con el rostro sobreexpuesto (demasiada luz). En cualquier condición que no sea buena, el código no detectará los gestos y en casos extremos, ni si quiera el rostro del usuario.

6 CONCLUSIONES

Finalmente, y para concluir este trabajo, sería conveniente dar algunas ideas generales obtenidas o formas de continuar con el mismo, el cual ha resultado ser muy interesante y una vía de estudio para futuros trabajos o, por que no, una primera idea para desarrollar un producto que satisfaga las necesidades de las personas.

No ha sido así durante todo este documento, pero si me lo permiten, me expresaré en este apartado en primera persona ya que trataré de exponer los problemas encontrados o mi opinión al respecto.

6.1 Complicaciones encontradas en el trabajo

En primer lugar, me encontré con la dificultad de apenas conocer el lenguaje Python y, mucho menos, la tecnología asociada con la inteligencia artificial o la vision artificial. Los primeros meses de dicho trabajo se centraron en aprender al respecto y empezar a orientar la forma de afrontarlo.

Una vez manos a la obra, me encontré con los quebraderos de cabeza de las librerías de Python. Es totalmente cierto que gracias a ellas nos evitamos muchas líneas de código, es más, gracias a las librerías he sido capaz de realizar este trabajo que sin ellas no hubiera sido posible ya que, por ejemplo, desconozco totalmente como crear un detector de marcas faciales, por poner un ejemplo. Pero también es cierto que la instalación de estas librerías se complica en algunas ocasiones, es por ello que he intentado centrar un apartado de este trabajo en eso mismo, para facilitarle en la medida de lo posible el trabajo a cualquier otro alumno o persona que desee realizar algo parecido. En mi caso, he destinado varios días al proceso de instalación, ya que ha sido complejo, además de encontrarme en la situación en mitad del trabajo de desinstalar y volver a instalar todo debido al problema comentado anteriormente de que *Home I/O* solo funciona mediante Python si este es de 32 bits.

Al igual que he comentado en el ultimo punto de la sección anterior, estos métodos son muy susceptibles a la calidad de las imágenes. Es por ello que es imprescindible disponer de una buena iluminación a la hora de realizar las pruebas. De igual forma, la librería Dlib que se ha usado, a pesar de estar muy extendida y ser usada en infinidad de aplicaciones, tiene ciertas limitaciones.

Además, debido a la limitación de puntos que se obtienen con dicha librería, es complicado obtener más cantidad de códigos que detecten otros gestos faciales.

6.2 Posibles líneas de mejora

Desde luego, se me ocurren infinidad de mejoras a implementar en este trabajo, ya que como he comentado, esto es un prototipo de interfaz y aún hay mucho por avanzar. Enumerándo algunas de las ideas:

- En primer lugar, la idea principal de este trabajo era implementar el código en unos equipos domóticos reales. Sin embargo, esto ha sido imposible realizarlo debido a no tener acceso a ellos. Por tanto, una primera línea podría ser completar este trabajo haciendo la adaptación a dispositivos Z-Wave, por ejemplo.
- De acuerdo a lo que he comentado en el apartado anterior, la librería Dlib tiene bastantes limitaciones. Es por ello que si en el futuro se desea seguir avanzando en este trabajo, una de las opciones sería probar algunas de esas librerías privadas o programas que obtienen los puntos faciales y que se encuentran por la red pero que son de pago. Bajo mi punto de vista, esto sería interesante en el caso de querer desarrollar un producto para ponerlo en el mercado o para usarlo de forma real.

- Como he comentado al inicio de las conclusiones, mis conocimientos en Python son limitados y es por ello que, seguramente, los códigos propuestos pueden mejorarse además de crearse otros nuevos que detecten nuevos gestos faciales.
- Se me ocurre una forma de tener un control total de los dispositivos de la vivienda únicamente usando los tres detectores de gestos faciales propuestos en este trabajo. Para ello, se debería crear una aplicación o interfaz que se pudiera recorrer entrando y saliendo de los menús mediante los gestos. De esa forma, el usuario tendría total acceso a cada una de las zonas y dispositivos de la vivienda.
- Como se ha venido comentando a lo largo del trabajo, esto ha sido solo un ejemplo, pero lo idóneo sería adaptarlo a cada usuario en concreto. Sería interesante hacer una adaptación real a un usuario que lo necesite y sobre todo, escuchar sus opiniones al respecto para poder mejorar el proyecto.

6.3 Opinión personal final

Para concluir, y como opinión personal, remarcar la importancia de este tipo de aplicaciones y trabajos ya que abren nuevas vías de estudio que ayudarán en un futuro a las personas que lo necesiten en mayor medida, como en el caso de este trabajo, las personas discapacitadas.

Considero que es una buena forma de empezar para desarrollos posteriores, y una vía interesante, ya que bajo mi opinion el programa ha funcionado correctamente y se ha conseguido el objetivo propuesto. Desde luego que se puede mejorar, pero es una muy buena forma de empezar. Invito a cualquier otro alumno de la escuela, o de cualquier otra Universidad, a seguir avanzando en este bonito proyecto y que se pueda mejorar para facilitarle la vida a aquellas personas que lo necesiten.

REFERENCIAS

[1] Contaval, «Contaval,» 18 febrero 2016. [En línea]. Available: https://www.contaval.es/que-es-la-vision-artificial-y-para-que-sirve/. [Último acceso: 27 marzo 2020].

[2] REAL ACADEMIA ESPAÑOLA, «Diccionario de la lengua española, 23.ª ed., [versión 23.3 en línea],» 2019. [En línea]. Available: https://www.rae.es/. [Último acceso: 27 marzo 2020].

[3] F. G. Navarro, «Domótica: gestión de la energía y gestión técnica de edificios,» RA-MA Editorial, 2015.

[4] ASOCIACIÓN ESPAÑOLA DE DOMÓTICA E INMÓTICA, «CEDOM,» [En línea]. Available: http://www.cedom.es/sobre-domotica/que-es-domotica. [Último acceso: 2020 marzo 27].

[5] Real Games, «Real Games,» [En línea]. Available: https://realgames.co/.

[6] Real Games, [En línea]. Available: https://docs.realgames.co/homeio/en/. [Último acceso: 28 Abril 2020].

[7] Pyromaniac, «Pythones,» [En línea]. Available: https://pythones.net/importar-modulos-en-python/. [Último acceso: 3 Abril 2020].

[8] A. Rosebrock, «PyImageSearch,» 3 Abril 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/. [Último acceso: 2020 Abril 3].

[9] A. Rosebrock, «PyImageSearch,» 24 Abril 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/. [Último acceso: 2020 Abril 4].

[10] Anaconda, [En línea]. Available: https://www.anaconda.com/.

[11] Wikipedia, «Wikipedia - Anaconda (distribución de Python),» 24 Octubre 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Anaconda_(distribuci%C3%B3n_de_Python). [Último acceso: 28 Abril 2020].

[12] Real Games, «Using with Python,» [En línea]. Available: https://docs.realgames.co/homeio/en/python/. [Último acceso: 2020 Abril 29].

[13] Real Games, «Devices Map,» [En línea]. Available: https://docs.realgames.co/homeio/en/devices-map/. [Último acceso: 1 Mayo 2020].

[14] Real Games, «Memory Addresses,» [En línea]. Available: https://docs.realgames.co/homeio/en/memory-addresses/#outputs. [Último acceso: 1 Mayo 2020].

7 ANEXOS

7.1 Código principal

```
1.  import sys
2.  import time
3.  import clr
4.  from scipy.spatial import distance as dist
5.  from imutils.video import FileVideoStream
6.  from imutils.video import VideoStream
7.  from imutils import face_utils
8.  import numpy as np
9.  import argparse
10. import imutils
11. import dlib
12. import cv2
13.
14. clr.AddReference('EngineIO')
15.
16. from EngineIO import *
17.
18. def eye_aspect_ratio(eye):
19.     # compute the euclidean distances between the two sets of
20.     # vertical eye landmarks (x, y)-coordinates
21.     A = dist.euclidean(eye[1], eye[5])
22.     B = dist.euclidean(eye[2], eye[4])
23.
24.     # compute the euclidean distance between the horizontal
25.     # eye landmark (x, y)-coordinates
26.     C = dist.euclidean(eye[0], eye[3])
27.
28.     # compute the eye aspect ratio
29.     ear = (A + B) / (2.0 * C)
30.
31.     # return the eye aspect ratio
32.     return ear
33.
34. def mouth_aspect_ratio(mouth):
35.     # compute the euclidean distances between the two sets of
36.     # vertical mouth landmarks (x, y)-coordinates
37.     A = dist.euclidean(mouth[13], mouth[19])
38.     B = dist.euclidean(mouth[14], mouth[18])
39.     C = dist.euclidean(mouth[15], mouth[17])
40.
41.     # compute the euclidean distance between the horizontal
42.     # mouth landmark (x, y)-coordinates
43.     D = dist.euclidean(mouth[12], mouth[16])
44.
45.     # compute the mouth aspect ratio
46.     mar = (A + B + C) / (2.0 * D)
47.
48.     # return the mouth aspect ratio
49.     return mar
50.
51. def eyebrow_aspect_ratio(eye,eyebrow):
52.     # compute the euclidean distances between the two sets of
53.     # vertical eyebrow and eyes landmarks (x, y)-coordinates
54.     A = dist.euclidean(eyebrow[1], eye[0])
55.     B = dist.euclidean(eyebrow[2], eye[3])
```

```
    C = dist.euclidean(eyebrow[3], eye[4])

    # compute the euclidean distance between the horizontal
    # eyebrow landmark (x, y)-coordinates
    D = dist.euclidean(eyebrow[0], eyebrow[4])

    # compute the eye aspect ratio
    ebar = (A + B + C) / (3.0 * D)

    # return the eyebrown aspect ratio
    return ebar

# define constants for the aspect ratio
EYE_AR_THRESH = 0.3
AR_CONSEC_FRAMES = 3
EYE_SECONDS_AR_CONSEC_FRAMES = 30
MOUTH_AR_THRESH = 0.6
EYEBROW_AR_THRESH = 0.6

# initialize the frame counters and the total number of open
COUNTER_EYE = 0
COUNTER_MOUTH = 0
COUNTER_EYEBROW = 0
TOTAL_EYE = 0

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

#Recibo la dirección que quiero modificar en HOME I/O
# Luces zona A
livingRoomLight = MemoryMap.Instance.GetBit(0, MemoryType.Output)
# Luces zona B
GuestsRestroom1Light = MemoryMap.Instance.GetBit(19, MemoryType.Output)
GuestsRestroom2Light = MemoryMap.Instance.GetBit(20, MemoryType.Output)
# Luces zona C
PantryLight = MemoryMap.Instance.GetBit(30, MemoryType.Output)
# Luces zona D
Kitchen1Light = MemoryMap.Instance.GetBit(40, MemoryType.Output)
Kitchen2Light = MemoryMap.Instance.GetBit(41, MemoryType.Output)
# Luces zona E
EntranceHallLight = MemoryMap.Instance.GetBit(54, MemoryType.Output)
        # Luces zona F
        Garage1Light = MemoryMap.Instance.GetBit(68, MemoryType.Output)
        Garage2Light = MemoryMap.Instance.GetBit(69, MemoryType.Output)
        # Luces zona G
        BedroomCorridorLight = MemoryMap.Instance.GetBit(83, MemoryType.Output)
        # Luces zona H
        ChildrenBedroomLight = MemoryMap.Instance.GetBit(97, MemoryType.Output)
        # Luces zona I
        Bathroom1Light = MemoryMap.Instance.GetBit(110, MemoryType.Output)
        Bathroom2Light = MemoryMap.Instance.GetBit(111, MemoryType.Output)
        # Luces zona J
        SingleBedroomLight = MemoryMap.Instance.GetBit(122, MemoryType.Output)
        # Luces zona K
        PrivateBathroomLight = MemoryMap.Instance.GetBit(135, MemoryType.Output)
        # Luces zona L
        CoupleBedroomLight = MemoryMap.Instance.GetBit(146, MemoryType.Output)
        # Luces zona M
        LaundryRoomLight = MemoryMap.Instance.GetBit(159, MemoryType.Output)
        # Luces zona N
        HomeOffice1Light = MemoryMap.Instance.GetBit(172, MemoryType.Output)
        HomeOffice2Light = MemoryMap.Instance.GetBit(173, MemoryType.Output)
        HomeOffice3Light = MemoryMap.Instance.GetBit(174, MemoryType.Output)
        # Luces zona O
```

```
123.        ExteriorPorch1Light = MemoryMap.Instance.GetBit(187, MemoryType.Output)
124.        ExteriorPorch2Light = MemoryMap.Instance.GetBit(188, MemoryType.Output)
125.        ExteriorPoolLight = MemoryMap.Instance.GetBit(189, MemoryType.Output)
126.        ExteriorGardenLight = MemoryMap.Instance.GetBit(190, MemoryType.Output)
127.        ExteriorEntranceLight = MemoryMap.Instance.GetBit(191, MemoryType.Output)
128.
129.        # Persianas zona A
130.        livingRoomRollerShades1Up = MemoryMap.Instance.GetBit(1, MemoryType.Output)
131.        livingRoomRollerShades1Down = MemoryMap.Instance.GetBit(2, MemoryType.Output)
132.        livingRoomRollerShades2Up = MemoryMap.Instance.GetBit(3, MemoryType.Output)
133.        livingRoomRollerShades2Down = MemoryMap.Instance.GetBit(4, MemoryType.Output)
134.        livingRoomRollerShades3Up = MemoryMap.Instance.GetBit(5, MemoryType.Output)
135.        livingRoomRollerShades3Down = MemoryMap.Instance.GetBit(6, MemoryType.Output)
136.        livingRoomRollerShades4Up = MemoryMap.Instance.GetBit(7, MemoryType.Output)
137.        livingRoomRollerShades4Down = MemoryMap.Instance.GetBit(8, MemoryType.Output)
138.
139.        # grab the indexes of the facial landmarks for eyes, the mouth
140.        # and the eyebrowns
141.        (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
142.        (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
143.        (MStart, MEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
144.        (lebStart, lebEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eyebrow"]
145.        (rebStart, rebEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eyebrow"]
146.
147.
148.        # start the video stream thread
149.        print("[INFO] starting video stream thread...")
150.        #vs = VideoStream().start()
151.        #fileStream = True
152.        vs = VideoStream(src=0).start()
153.        # vs = VideoStream(usePiCamera=True).start()
154.        # fileStream = False
155.        time.sleep(1.0)
156.
157.        # loop over frames from the video stream
158.        while True:
159.            # if this is a file video stream, then we need to check if
160.            # there any more frames left in the buffer to process
161.            #if fileStream and not vs.more():
162.            #   break
163.
164.            # grab the frame from the threaded video file stream, resize
165.            # it, and convert it to grayscale
166.            # channels)
167.            frame = vs.read()
168.            frame = imutils.resize(frame, width=450)
169.            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
170.
171.            # detect faces in the grayscale frame
172.            rects = detector(gray, 0)
173.
174.            # loop over the face detections
175.            for rect in rects:
176.                # determine the facial landmarks for the face region, then
177.                # convert the facial landmark (x, y)-coordinates to a NumPy
178.                # array
179.                shape = predictor(gray, rect)
180.                shape = face_utils.shape_to_np(shape)
181.
182.                # extract the left and right eye coordinates, then use the
183.                # coordinates to compute the eye aspect ratio for both eyes
184.                leftEye = shape[lStart:lEnd]
185.                rightEye = shape[rStart:rEnd]
186.                leftEAR = eye_aspect_ratio(leftEye)
187.                rightEAR = eye_aspect_ratio(rightEye)
188.
```

```
189.                # extract the mouth coordinates, then use the
190.                # coordinates to compute the mouth aspect ratio
191.                Mouth = shape[MStart:MEnd]
192.                MAR = mouth_aspect_ratio(Mouth)
193.
194.                # extract the left and right eye and eyebrow coordinates, then use the
195.                # coordinates to compute the eyebrow aspect ratio for both eyebrows
196.                leftEyebrow = shape[lebStart:lebEnd]
197.                rightEyebrow = shape[rebStart:rebEnd]
198.                leftEBAR = eyebrow_aspect_ratio(leftEye, leftEyebrow)
199.                rightEBAR = eyebrow_aspect_ratio(rightEye, rightEyebrow)
200.
201.                # average the eye aspect ratio together for both eyes
202.                ear = (leftEAR + rightEAR) / 2.0
203.
204.                # average the mouth aspect ratio
205.                mar = MAR
206.
207.                # average the eyebrow aspect ratio together for both eyebrows
208.                ebar = (leftEBAR + rightEBAR) / 2.0
209.
210.                # compute the convex hull for the left and right eye, then
211.                # visualize each of the eyes
212.                leftEyeHull = cv2.convexHull(leftEye)
213.                rightEyeHull = cv2.convexHull(rightEye)
214.                cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
215.                cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
216.
217.                # compute the convex hull for the mouth, then
218.                # visualize
219.                MouthHull = cv2.convexHull(Mouth)
220.                cv2.drawContours(frame, [MouthHull], -1, (0, 255, 0), 1)
221.
222.                # compute the convex hull for the left and right eyebrow, then
223.                # visualize each of the eyebrows
224.                leftEyebrowHull = cv2.convexHull(leftEyebrow)
225.                rightEyebrowHull = cv2.convexHull(rightEyebrow)
226.                cv2.drawContours(frame, [leftEyebrowHull], -1, (0, 255, 0), 1)
227.                cv2.drawContours(frame, [rightEyebrowHull], -1, (0, 255, 0), 1)
228.
229.                # Detección parpadear
230.                # check to see if the eye aspect ratio is below the blink
231.                # threshold, and if so, increment the blink frame counter
232.                if ear < EYE_AR_THRESH:
233.                    COUNTER_EYE += 1
234.
235.                # otherwise, the eye aspect ratio is not below the blink
236.                # threshold
237.                else:
238.                    # if the eyes were closed for a sufficient number of
239.                    # then increment the total number of blinks
240.                    if COUNTER_EYE >= AR_CONSEC_FRAMES and COUNTER_EYE <= EYE_SECONDS_AR
    _CONSEC_FRAMES:
241.                        TOTAL_EYE += 1
242.                        if TOTAL_EYE == 1:
243.                            tiempo_EYE = time.time() + 2
244.                        if TOTAL_EYE >= 3 and time.time() <= tiempo_EYE:
245.                            livingRoomLight.Value = not livingRoomLight.Value
246.                            # When using a memory value before calling the Update method
     we are using a cached value.
247.                            print("Light is on? " + str(livingRoomLight.Value))
248.                            TOTAL_EYE = 0
249.                            # Calling the Update method will write the livingRoomLight.V
    alue to the memory map.
250.                            MemoryMap.Instance.Update()
251.                        elif time.time() >= tiempo_EYE:
252.                            TOTAL_EYE = 1
```

```
                            tiempo_EYE = time.time() + 2
                    elif COUNTER_EYE >= EYE_SECONDS_AR_CONSEC_FRAMES:
                        # Se apagan todas las luces
                        livingRoomLight.Value = 0
                        GuestsRestroom1Light.Value = 0
                        GuestsRestroom2Light.Value = 0
                        PantryLight.Value = 0
                        Kitchen1Light.Value = 0
                        Kitchen2Light.Value = 0
                        EntranceHallLight.Value = 0
                        Garage1Light.Value = 0
                        Garage2Light.Value = 0
                        BedroomCorridorLight.Value = 0
                        ChildrenBedroomLight.Value = 0
                        Bathroom1Light.Value = 0
                        Bathroom2Light.Value = 0
                        SingleBedroomLight.Value = 0
                        PrivateBathroomLight.Value = 0
                        CoupleBedroomLight.Value = 0
                        LaundryRoomLight.Value = 0
                        HomeOffice1Light.Value = 0
                        HomeOffice2Light.Value = 0
                        HomeOffice3Light.Value = 0
                        ExteriorPorch1Light.Value = 0
                        ExteriorPorch2Light.Value = 0
                        ExteriorPoolLight.Value = 0
                        ExteriorGardenLight.Value = 0
                        ExteriorEntranceLight.Value = 0

                        # When using a memory value before calling the Update method we
    are using a cached value.
                        print("Light is on? " + str(livingRoomLight.Value))

                        # Calling the Update method will write the livingRoomLight.Value
     to the memory map.
                        MemoryMap.Instance.Update()


                    # reset the eye frame counter
                    COUNTER_EYE = 0

                # check to see if the mouth aspect ratio is below the open
                # threshold, and if so, increment the open frame counter
                if mar > MOUTH_AR_THRESH:
                    COUNTER_MOUTH += 1

                # otherwise, the mouth aspect ratio is not below the open
                # threshold
                else:
                    # if the mouth were opened for a sufficient number of
                    # then increment the total number of open
                    if COUNTER_MOUTH >= AR_CONSEC_FRAMES:
                        livingRoomRollerShades1Up.Value = not livingRoomRollerShades1Up.
    Value
                        livingRoomRollerShades2Up.Value = not livingRoomRollerShades2Up.
    Value
                        livingRoomRollerShades3Up.Value = not livingRoomRollerShades3Up.
    Value
                        livingRoomRollerShades4Up.Value = not livingRoomRollerShades4Up.
    Value
                        livingRoomRollerShades1Down.Value = 0
                        livingRoomRollerShades2Down.Value = 0
                        livingRoomRollerShades3Down.Value = 0
                        livingRoomRollerShades4Down.Value = 0

                        # Calling the Update method will write to the memory map.
```

```
                        MemoryMap.Instance.Update()

                    # reset the mouth frame counter
                    COUNTER_MOUTH = 0

                # check to see if the eyebrow aspect ratio is below the open
                # threshold, and if so, increment the open frame counter
                if ebar > EYEBROW_AR_THRESH and ear > EYE_AR_THRESH:
                    COUNTER_EYEBROW += 1

                # otherwise, the eyebrow aspect ratio is not below the open
                # threshold
                else:
                    # if the eyebrow were uped for a sufficient number of
                    # then increment the total number of open
                    if COUNTER_EYEBROW >= AR_CONSEC_FRAMES:
                        livingRoomRollerShades1Down.Value = not livingRoomRollerShades1D
    own.Value
                        livingRoomRollerShades2Down.Value = not livingRoomRollerShades2D
    own.Value
                        livingRoomRollerShades3Down.Value = not livingRoomRollerShades3D
    own.Value
                        livingRoomRollerShades4Down.Value = not livingRoomRollerShades4D
    own.Value
                        livingRoomRollerShades1Up.Value = 0
                        livingRoomRollerShades2Up.Value = 0
                        livingRoomRollerShades3Up.Value = 0
                        livingRoomRollerShades4Up.Value = 0

                        # Calling the Update method will write to the memory map.
                        MemoryMap.Instance.Update()

                    # reset the eyebrow frame counter
                    COUNTER_EYEBROW = 0


            # show the frame
            cv2.imshow("Frame", frame)
            key = cv2.waitKey(1) & 0xFF

            # if the `q` key was pressed, break from the loop
            if key == ord("q"):
                break

        # When we no longer need the MemoryMap we should call the Dispose method to rele
    ase all the allocated resources.
        MemoryMap.Instance.Dispose()

        print("Bye!")

        # do a bit of cleanup
        cv2.destroyAllWindows()
        vs.stop()
```

GLOSARIO

Python: lenguaje de programación 1

Anaconda: Python: lenguaje de programación 1

Home I/O: software de im simulador de vivienda inteligente 1

Real Games: empresa desarrolladora del software Home I/O *5*

STEM: Science, Technology, Engineering and Mathematics *5*

PLC: controlador lógico programable 6

Modbus: protocolo de comunicaciones 6

Tracking: seguimiento 8

Conda: gestor de paquetes 11

Pip: gestor de paquetes 11