UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA BIOMÉDICA

DESARROLLO DE UN SISTEMA DE AYUDA AMBIENTAL BASADO EN RASPBERRY PI PARA PERSONAS CON DIVERSIDAD FUNCIONAL MOTORA

AUTORA: ANA RODRÍGUEZ DÍAZ

TUTOR: CARLOS ROLDÁN PORTA

COTUTOR: CARLOS ROLDÁN BLAY

Curso Académico: 2020-21

AGRADECIMIENTOS

A pesar de tratarse de un trabajo individual, el desarrollo de este TFG ha sido fruto de la colaboración de muchas personas que me han acompañado en este último tiempo de poner fin a un camino de cuatro años en los que me he estado formando para ser ingeniera biomédica. Por eso, me gustaría agradecer, en primer lugar, a los profesores que han dedicado su tiempo a enseñarme, dirigirme y aconsejarme para que pueda crecer como profesional. Especialmente agradezco a mi tutor, Carlos Roldán Porta, por la paciencia, la dedicación y la entrega con la que me ha guiado durante estos meses.

Por supuesto, estoy enormemente agradecida a mi familia y a mis amigos, quienes, con su alegría, su cariño y su ayuda siempre desinteresada me han dado los empujones necesarios para que perseverara en el trabajo, animándome a sacar lo mejor de mi y enseñándome a disfrutar de cada momento y a valorar las cosas realmente importantes. No tengo palabras de agradecimiento suficientes. Echando la vista atrás, entiendo que es un verdadero privilegio haber recibido esta formación, y espero poder poner todos mis conocimientos al servicio de los demás, contando siempre con la ayuda adecuada.

RESUMEN

Las personas con diversidad funcional motora encuentran con frecuencia dificultades a la hora de realizar las tareas domésticas más sencillas [1]. Esto provoca una situación de dependencia que empeora su calidad de vida y la de sus cuidadores. Las llamadas tecnologías de apoyo buscan proporcionar una ayuda adecuada para estas personas, que les permita ganar más autonomía en su día a día. Sin embargo, hoy en día, solamente una de cada diez personas que requieren algún producto de asistencia tiene acceso al mismo [2]. Esto es debido, en gran medida, al elevado coste de las soluciones diseñadas, que se debe, a su vez, a la necesidad de fabricar productos personalizados que cubran los requerimientos concretos de cada usuario.

En el presente trabajo final de grado se pretende desarrollar un sistema de ayuda para personas con diversidad funcional motora de bajo coste que les permita actuar sobre elementos de su entorno (alumbrado, persianas, calefacción, dispositivos, etc.) de manera fácil y adecuada a su condición. Así, se ha diseñado un prototipo basado en el miniordenador monoplaca Raspberry Pi. Se trata de un dispositivo móvil que ofrece una interfaz de usuario donde muestra una serie de opciones de un menú gráfico, tales como encender la luz, subir la persiana o estirar de la cadena. Cuando se selecciona una acción, el dispositivo emitirá una orden por un sistema de comunicación inalámbrico y un receptor especial detectará la orden y activará el accionamiento adecuado.

La interacción con la persona se basa, principalmente, en técnicas de visión artificial, de forma que, con una cámara, se pueden captar distintos gestos y movimientos que sean reconocidos por el sistema y transformados en órdenes. Estos se adaptarán a las posibilidades de cada usuario para ofrecer una solución versátil y económica.

Palabras clave: Raspberry Pi; Sistema de ayuda ambiental; diversidad funcional; tecnologías de apoyo; visión artificial.

RESUM

Les persones amb diversitat funcional motora troben amb freqüència dificultats a l'hora de realitzar les tasques domèstiques més senzilles [1]. Això provoca una situació de dependència que empitjora la seua qualitat de vida i la dels seus cuidadors. Les anomenades tecnologies de suport busquen proporcionar una ajuda adequada per a aquestes persones, que els permeta guanyar més autonomia en el seu dia a dia. No obstant això, hui dia solament una de cada deu persones que requereixen algun producte d'assistència té accés a aquest [2]. Això és degut principalment a l'elevat cost de les solucions dissenyades, que es deu, al seu torn, a la necessitat de fabricar productes personalitzats que cobrisquen els requeriments concrets de cada usuari.

En el present treball de fi de grau es pretén desenvolupar un sistema d'ajuda per a persones amb diversitat funcional motora de baix cost que els permeta actuar sobre elements del seu entorn (enllumenat, persianes, calefacció, dispositius, etc.) de manera fàcil i adequada a la seua condició. Així, s'ha dissenyat un prototip basat en el miniordinador monoplaca Raspberry Pi. Es tracta d'un dispositiu mòbil que ofereix una interfície d'usuari on mostra una sèrie d'opcions d'un menú gràfic, com ara encendre la llum, pujar la persiana o estirar de la cadena. Quan es seleccione una acció, el dispositiu emetrà una ordre per un sistema de comunicació sense fil i un receptor especial detectarà l'ordre i activarà l'accionament adequat.

La interacció amb la persona es basa, principalment, en tècniques de visió artificial, de manera que, amb una càmera, es poden captar diferents gestos i moviments que siguen reconeguts pel sistema i transformats en ordres. Aquests s'adaptaran a les possibilitats de cada usuari per a oferir una solució versàtil i econòmica.

Paraules clau: Raspberry Pi; Sistema d'ajuda ambiental; diversitat funcional; tecnologies de suport; visió artificial.

ABSTRACT

People with motor functional diversity often find it difficult to carry out the simplest domestic tasks [1]. This leads to a situation of dependency that worsens their quality of life and that of their caregivers. So-called assistive technologies aim to provide adequate support for these people, enabling them to gain more autonomy in their daily lives. However, today, only one in ten people who require an assistive product have access to it [2]. This is largely due to the high cost of the solutions designed, which is due, in turn, to the need to manufacture customised products that meet the specific requirements of each user.

The aim of this final degree project is to develop a low-cost assistance system for people with motor functional diversity that allows them to act on elements of their environment (lighting, blinds, heating, devices, etc.) in an easy and appropriate way for their condition. Thus, a prototype has been designed based on the Raspberry Pi single-board mini-computer. It is a mobile device that offers a user interface where it displays a series of options from a graphical menu, such as turning on the light, raising the blind or pulling the chain. When an action is selected, the device will issue a command via a wireless communication system and a special receiver will detect the command and activate the appropriate actuation.

The interaction with the person is mainly based on artificial vision techniques, so that, with a camera, different gestures and movements can be captured, recognised by the system and transformed into commands. These will be adapted to the possibilities of each user to offer a versatile and economical solution.

Keywords: Raspberry Pi; environmental assistance system; functional diversity; assistive technologies; artificial vision.

ÍNDICE GENERAL

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Presupuesto
- Anexos

ÍNDICE DE LA MEMORIA

CAPÍTULO 1. INTRODUCCIÓN 11

1.1. MOTIVACIÓN 11
1.2. ANTECEDENTES 11
1.3. JUSTIFICACIÓN 13

CAPÍTULO 2. MARCO TEÓRICO 14

2.1. DIVERSIDAD FUNCIONAL MOTORA 14
2.2. SISTEMAS AUMENTATIVOS Y ALTERNATIVOS DE COMUNICACIÓN (SAAC) 14
2.3. RASPBERRY PI 15
2.4. DIGISPARK 17
2.5. ARDUINO IDE 18
2.6. VISIÓN ARTIFICIAL 18
2.6.1. OpenCV 18
2.6.2. Dlib 19
2.6.3. imutils 20
2.7. KIVY 20
2.8. ENTORNO VIRTUAL 20
2.9. TRANSMISIÓN INALÁMBRICA 21
2.10. COMUNICACIÓN SERIAL. PROTOCOLO I2C 21
2.11. DOMÓTICA 22

CAPÍTULO 3. OBJETIVOS 23

CAPÍTULO 4. DESARROLLO DEL PROTOTIPO 24

4.1. FUNCIONAMIENTO GENERAL 24
4.2. ENTRADA. ELEMENTOS DE ACTIVACIÓN 25
4.2.1. Detección de elementos de activación mediante visión artificial 27
A. Función de detección de movimiento 28
B. Detección de gestos faciales 34
4.2.2. Pulsador como elemento de activación. 37
4.3. SALIDA. PROCESO DE ENVÍO DE LA SEÑAL 37
4. SISTEMA DE TRIAJE, PERSONALIZACIÓN DE LA APP Y CALIBRACIÓN 39

CAPÍTULO 5. RESULTADOS Y CONCLUSIÓN 40

5.1. RESULTADOS 40
5.2. CONCLUSIONES 43

BIBLIOGRAFÍA 44

ÍNDICE DEL PRESUPUESTO

PRESUPUESTO 49

1. INTRODUCCIÓN 49
2. PRESUPUESTO DEL DESARROLLO DEL PROTOTIPO 49
3. COMERCIALIZACIÓN DEL PRODUCTO 52

ÍNDICE DE LOS ANEXOS

ANEXO 1. CÓDIGO SCRIPTS DE LA RASPBERRY 56

1. INTRODUCCIÓN 56
2. SCRIPT DETECTA_GESTOFACIAL.PY 56
3. SCRIPT DETECTA_MOVIMIENTO.PY 58
3. SCRIPT MAINAPP.PY 60

ANEXO 2. CÓDIGO SKETCHS DE LAS PLACAS DIGISPARK 65

1. INTRODUCCIÓN 65
2. SKETCH DE LA PLACA DIGISPARK EMISORA 65
3. SKETCH PLACA RECEPTORA PERSIANA 66
4. SKETCH PLACA RECEPTORA LUZ 66
5. SKETCH PLACA RECEPTORA CADENA 67

ÍNDICE DE FIGURAS

Figura 1: a) Big Lib Switch (42€): diámetro de 125 mm, es resistente y permite la colocación de pictogramas. b) Candy Corn (149€): es un pulsador que se activa por proximidad, sin contacto físico. c) PowerLink 4 (250€): diseñado para proporcionar acceso por conmutación/control de electrodomésticos con funcionalidad básica de encendido-apagado. Fuente: TecnoAccesible. 12

Figura 2: Ejemplos de pictogramas descargables. Fuente: ARASAAC 15

Figura 3: Descripción de los pines GPIO de la Raspberry Pi modelo 2B. Fuente [24]. 15

Figura 4: Esquema de las partes de la monoplaca Raspberry Pi 4B. Principales características 16

Figura 5: Raspberry Pi 2B con la cámara PiNoIR conectada. Fuente: página web de la fundación Raspberry Pi. 17

Figura 6: Pantalla oficial Raspberry Pi. Fuente: página web de la fundación Raspberry Pi. 17

Figura 7: Digispark Pinout. Fuente: [26] 18

Figura 8: Resultado de la aplicación del algoritmo de detección de marcas faciales de dlib en una imagen de la base de datos HELEN. Fuente: [31] 19

Figura 9: Visualización de las 68 coordenadas de marcas faciales que devuelve la base de datos iBUG 300-W. Fuente: [34] 20

Figura 10: Módulos de transmisión inalámbrica por RF de 433MHz, emisor (derecha) y receptor (izquierda). Fuente: naylampmechatronics 21

Figura 11: Diagrama de flujo del funcionamiento general del sistema. Fuente: elaboración propia mediante la aplicación web lucid (https://lucid.co/es). 24

Figura 12: Flujo de funcionamiento de la programación con hilos teórico (izquierda) y real (derecha) 26

Figura 13: Diagrama de flujo del funcionamiento de la App mediante la programación con hilos. . 27

Figura 14: Esquema del funcionamiento del algoritmo de sustracción de fondo. Fuente: [48] 28

Figura 15: Resultados de la aplicación de los algoritmos de sustracción de fondo implementados en OpenCV a) imagen original. b) MOG. c) MOG2. d) GMG. 29

Figura 16: Resultados de la implementación del algoritmo MOG a) tras mover ligeramente la cabeza hacia la izquierda. b) errores de detección debidos a cambios en la iluminación al pasar la mano por delante de la cabeza. 30

Figura 17: Gráfica del umbral simple aplicado. Fuente: [54]. 31

Figura 18: Ejemplo de dilatación de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras la dilatación. Fuente: [56] 31

Figura 19: Ejemplo de erosión de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras la erosión. Fuente: [56] 32

Figura 20: Ejemplo de apertura de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras la apertura. Fuente: [56] 32

Figura 21: Ejemplo de cierre de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras el cierre. Fuente: [56] 32

Figura 22: Resultado de la aplicación del algoritmo de detección de movimiento desarrollado a) tras mover ligeramente la cabeza, aunque detecta cambios, no lo considera como movimiento por no ser significativos. b) tras mover la cabeza intencionadamente, el área de píxeles grandes es lo bastante grande como para considerarse movimiento. c) tras mover la mano por delante de la cabeza, detecta el movimiento correctamente. 33

Figura 23: Las 6 marcas faciales que definen el ojo. Fuente: [63] 35

Figura 24: Marcas faciales en el ojo abierto y cerrado. La gráfica muestra el EAR (ecuación 8) de varios fotogramas de un vídeo donde ha ocurrido un pestañeo. Fuente [64] 36

Figura 25: Selección de los 8 puntos, de entre los 20 que definen el contorno de la boca, utilizados para detectar la apertura de la misma. Fuente: elaboración propia. 36

Figura 26: Las 8 marcas faciales que definirán el movimiento de la boca. La gráfica muestra el MAR (ecuación 9) de varios fotogramas de un vídeo donde se ha abierto la boca una vez. Fuente: [65]. 37

Figura 27: Esquema de las relaciones entre los diferentes elementos del sistema. Fuente: elaboración propia. 38

Figura 28: Esquema del funcionamiento del sistema de ayuda. Fuente: elaboración propia. 38

Figura 29: Esquema de conexiones del prototipo desarrollado. Fuente: elaboración propia. 40

Figura 30: Ventanas de la aplicación diseñadas para el prototipo. a) Ventana 1: persiana. b) Ventana 2: luz c) Ventana 3: cadena. Fuente: elaboración propia. 40

Figura 30: Imágenes del prototipo diseñado. a) Protoboard con los diferentes dispositivos externos conectados siguiendo el esquema de la figura 29 b) pantalla de la Raspberry Pi con la aplicación en marcha y la cámara colocada para la captación de las imágenes c) capturas de pantalla de la aplicación en marcha y el output de las funciones de detección de movimientos mediante visión artificial: (1) detecta_pestañeo() (2) dir_derecha() (3) detecta_BocaAbierta(). Fuente: elaboración propia. 41

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

MEMORIA

CAPÍTULO 1. INTRODUCCIÓN

1.1. MOTIVACIÓN

Las personas con diversidad funcional motora presentan a menudo numerosas dificultades a la hora de realizar las tareas más sencillas en sus hogares. Según el Instituto Nacional de Estadística (INE), el 62,2% de las personas con discapacidad presentan limitaciones en la movilidad, y el 55,3% tienen problemas relacionados con las tareas domésticas [1]. Como consecuencia, su nivel de dependencia suele ser muy elevado y disponen de poca autonomía, reduciéndose de esta manera su calidad de vida y la de las personas que están a su cuidado. De la mano de las tecnologías, es posible simplificar la realización de ciertas tareas, tales como subir o bajar las persianas, encender y apagar las luces, estirar de la cadena del wáter o controlar la temperatura del radiador, entre muchas otras.

En relación con esta problemática, el microordenador Raspberry Pi (R-PI), sacado al mercado en 2012, ha mostrado en los últimos años tener un gran potencial en diferentes áreas a un coste muy asequible. Dispone de una serie de pines que le permiten conectar diversos dispositivos de entrada y salida, siendo capaz, por tanto, de interactuar con el entorno. Concretamente, este miniordenador dispone de un sistema operativo capacitado para procesar las imágenes adquiridas por una cámara, de forma que, distintos gestos pueden ser captados y traducidos en órdenes.

Según la Organización Mundial de la Salud (OMS), solamente una de cada diez personas que requieren algún producto de asistencia tiene acceso al mismo [2]. Así pues, la motivación principal de este proyecto es poder desarrollar y evaluar una solución tecnológica de bajo coste para aumentar la autonomía y la calidad de vida de las personas con diversidad funcional motora en el interior de sus viviendas.

Finalmente, el presente trabajo permitirá aplicar los conocimientos y las habilidades aprendidas durante los cuatro años del grado. Se trata de un proyecto altamente interesante, puesto que supone una profundización en diversas áreas relevantes, como son el tratamiento de imágenes, la electrónica, la programación con Python y la aplicación de las TIC para tratar de proponer una solución viable a un problema del mundo real. Todo ello, contribuye a mejorar la formación como ingeniera biomédica.

1.2. ANTECEDENTES

Diversos dispositivos tecnológicos han sido diseñados específicamente para facilitar el uso de las nuevas tecnologías a las personas con discapacidad. Encontramos en el mercado una amplia variedad de soluciones adaptadas, tales como diferentes tipos de pulsadores o módulos y sistemas que permiten al usuario controlar diversos elementos funcionales del hogar a distancia. Una buena recopilación de estos productos queda recogida en el portal de internet TecnoAccesible[1] donde se puede consultar el catálogo ordenado en función de la categoría del producto. Aunque, dentro de la gran variedad de soluciones que reúne la página, el principal inconveniente es su elevado coste.

[1] Se puede consultar su sitio web: https://www.tecnoaccesible.net

a) *b)* *c)*

Figura 1: *a) Big Lib Switch (42€): diámetro de 125 mm, es resistente y permite la colocación de pictogramas. b) Candy Corn (149€): es un pulsador que se activa por proximidad, sin contacto físico. c) PowerLink 4 (250€): diseñado para proporcionar acceso por conmutación/control de electrodomésticos con funcionalidad básica de encendido-apagado. Fuente: TecnoAccesible.*

De las soluciones que encontramos en el mercado resultan especialmente interesantes dos de ellas. En primer lugar, la microempresa de emprendimiento social Mouse4all desarrolló un producto que permite utilizar la pantalla táctil de una tableta o teléfono Android sin tocarla, brindando a los usuarios una mayor autonomía y privacidad, y contribuyendo así a su desarrollo personal y la mejora de su calidad de vida [3]. Esta misma empresa, en colaboración con el Centro Dato, una asociación dedicada a la atención a personas con discapacidad física gravemente afectadas, comenzaron un proyecto de Transformación Digital ("Domótica Accesible") enmarcado en la Comunidad "Conectados por la Accesibilidad", de la Fundación Vodafone con la finalidad de desarrollar una App ("Domótica Fácil") manejar distintos elementos del hogar desde la tableta o el teléfono móvil [4].

La utilización de una cámara para captar movimientos que permitan el manejo de una pantalla es una solución adoptada por empresas e instituciones para posibilitar la utilización de dispositivos electrónicos a personas con diversidad funcional motora. A continuación, se mencionarán algunos ejemplos destacables de la puesta en práctica de la visión artificial con este propósito que encontramos en el mercado.

La empresa BJ Adaptaciones, dedicada a la creación y distribución de tecnología de asistencia para personas con discapacidad, ha desarrollado TrackerPro 2, un dispositivo que permite el control del ratón del ordenador, tableta o Smartphone con la cabeza. Es compatible con diversos sistemas operativos (Windows y MacOS), y el precio actual es de 1056,00 € [5].

Como alternativa gratuita, el Grupo de Investigación en Robótica de la Universitat de Lleida desarrolló en 2008 HeadMouse, un programa capaz de controlar el desplazamiento del cursor del PC con ligeros movimientos de la cabeza, y hacer clics mediante gestos faciales, a elegir entre: abrir la boca, guiñar un ojo o mantenerse en el mismo punto durante un tiempo determinado. La última versión data de 2016 y es compatible únicamente con Windows (XP, Vista, 7 y 8) [6]. En la misma línea, CREA Sistemas Informáticos, ha desarrollado, gracias al apoyo de la Fundación Vodafone España, una aplicación gratuita y de código abierto, llamada EVA FACIAL MOUSE. Esta, permite acceder de forma alternativa (manos libres) a las funciones de un dispositivo Android por medio del seguimiento del rostro del usuario captado a través de la cámara frontal. Los clics se efectúan manteniendo el cursor unos instantes sobre el mismo punto [7]. Estos sistemas gratuitos presentan también algunas limitaciones, sobre todo relacionadas con temas de compatibilidad con ciertas aplicaciones o sistemas operativos, que dificultan una utilización efectiva y práctica.

Por otro lado, se han llevado a cabo iniciativas que emplean la Raspberry Pi y la visión artificial como base para el desarrollo de tecnologías de asistencia. Así, encontramos diferentes aproximaciones para el diseño de una silla de ruedas controlada con movimientos de los ojos [8], [9],

así como sistemas más complejos que permiten el control de la silla de ruedas gracias al desarrollo de interfaces hombre-máquina (HMI, por sus siglas en inglés *Human Machine Interface*) [10], [11].

Por último, han sido publicados proyectos destinados a ofrecer soluciones domóticas que ayuden a mejorar la autonomía y calidad de vida de personas en situación de dependencia, como ocurre en [12], que propone un diseño específicamente para personas con diversidad funcional motora mediante el empleo del Hardware Libre de Arduino, o bien proyectos como [13], que ofrece una alternativa basada en Raspberry Pi y que utiliza la visión artificial, destinada a facilitar el control de las actividades domésticas a personas mayores.

Los proyectos mencionados anteriormente evidencian el interés por el desarrollo de tecnologías de asistencia de calidad y de bajo coste que ayuden a solucionar los problemas cotidianos a los que se enfrentan diariamente las personas con diversidad funcional motora. Además, la Raspberry Pi ha demostrado ser una tecnología adecuada para el diseño de sistemas portables con esta finalidad. No obstante, se encuentran limitaciones que impiden la salida al mercado como son el precio inicial [12] y la correcta captura de imágenes debido a las condiciones ambientales, sobre todo a los errores que el cambio de luz puede introducir en el sistema [9], [13].

1.3. JUSTIFICACIÓN

La utilización de tecnologías de apoyo y medidas de adaptación del entorno han demostrado proporcionar una ayuda sustancial para la movilidad, las destrezas de autocuidado y las habilidades sociales de las personas afectadas, aliviando también el esfuerzo de los cuidadores [14]. Además, como destaca la OMS, les permite llevar una vida más sana, digna, productiva e independiente, haciéndoles partícipes de la vida en sociedad y evitando situaciones graves de exclusión y pobreza [2]. El claro impacto positivo que el manejo de estos dispositivos ha probado tener sobre los diferentes perfiles de las personas que los necesitan [15] incentiva la investigación de propuestas innovadoras que las hagan más accesibles.

Según [16], el uso de instrumentos tecnológicos de apoyo en el hogar es poco frecuente, debido, entre otros factores, a las barreras relacionadas con los dispositivos. Aunque la domótica ha avanzado considerablemente en los últimos años, sigue siendo necesario adaptar los diseños a las necesidades de las personas con discapacidad, cumpliendo con el principio de usabilidad para todos, buscando un desarrollo tecnológico inclusivo y favoreciendo el abaratamiento de costes y la difusión y comercialización de los productos [17].

Para finalizar, por lo que al marco legal respecta, existen leyes que amparan e incentivan el desarrollo de este tipo de dispositivos. El artículo 25 de la Declaración Universal de los Derechos Humanos señala que “toda persona tiene derecho a un nivel de vida adecuado” [18]. Asimismo, la ley General de derechos de las personas con discapacidad y de su inclusión social pone de manifiesto en su artículo 22 el derecho de las personas con discapacidad a “vivir de forma independiente y a participar plenamente en todos los aspectos de la vida”, haciendo hincapié en la necesidad de adoptar las medidas pertinentes, para garantizar la accesibilidad universal y en igualdad de condiciones a los bienes productos y servicios, entre ellos los sistemas y las tecnologías de la información y las comunicaciones, por parte de los poderes públicos [19].

CAPÍTULO 2. MARCO TEÓRICO

2.1. DIVERSIDAD FUNCIONAL MOTORA

Según [20], la discapacidad motriz se define como "una alteración de la capacidad del movimiento que implica, en distinto grado, a las funciones de desplazamiento y/o de manipulación, bucofonatorias o de la respiración que limita a la persona en su desarrollo personal y social. Generalmente son la consecuencia de lesiones medulares, parálisis cerebral, distrofias musculares, esclerosis múltiple, etcétera". Por tanto, se trata de una condición que abarca todas las alteraciones que pueden causar deterioro parcial o total de las habilidades motoras en cualquier parte del cuerpo, lo que ocasiona ciertas limitaciones a la hora de realizar las actividades propias de la edad de la persona [21].

Las personas con diversidad funcional motora presentan, con preocupante frecuencia, una disminución de su autonomía e independencia dentro de sus hogares, puesto que su condición les impide realizar las tareas más sencillas, como encender o apagar los distintos aparatos electrónicos (televisión, aire acondicionado, etc.), controlar la iluminación de la casa, o subir la persiana de la habitación. En un informe titulado "La discapacidad en cifras", elaborado por el Observatorio de la discapacidad del Ministerio de Trabajo y Asuntos Sociales, afirma que 39 de cada 1000 habitantes presentan dificultades para realizar las tareas del hogar [22]. Se trata de la segunda condición más frecuente en personas con diversidad funcional, simplemente superada por las dificultades para desplazarse fuera del hogar.

2.2. SISTEMAS AUMENTATIVOS Y ALTERNATIVOS DE COMUNICACIÓN (SAAC)

Los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) "son formas de expresión diferentes del lenguaje hablado que tienen como objetivo aumentar (aumentativo) y/o compensar (alternativo) las dificultades de comunicación y el lenguaje de muchas personas con dificultades en esta área" [23]. Es frecuente que las personas con una discapacidad física presenten también algún déficit cognitivo que les dificulte emplear un nivel de habla que les permita comunicarse de manera satisfactoria. Esto conlleva que muchas aplicaciones no sean comprensibles para estas personas, limitando su acceso a las mismas. Es el caso, por ejemplo, de gran parte de los pacientes con parálisis cerebral (PC), con alguna enfermedad neurológica como la esclerosis lateral amiotrófica (ELA) o los pacientes afectados gravemente por un ictus, entre otros.

El portal ARASAAC (https://arasaac.org) proporciona un gran número de recursos gráficos (pictogramas) que han sido utilizados en este proyecto para la confección de las pantallas de la aplicación, ampliando así el número de usuarios capaces de manejar el sistema. A continuación, se muestra una serie de ejemplos de pictogramas que describen las actividades domésticas más frecuentes.

Figura 2: Ejemplos de pictogramas descargables. Fuente: ARASAAC.

2.3. RASPBERRY PI

La Raspberry Pi es un ordenador monoplaca de bajo coste desarrollado por la fundación benéfica Raspberry Pi en Inglaterra con la finalidad de promover la enseñanza de la programación en las escuelas. Desde que salió al mercado en 2012, este producto ha ganado mucha popularidad debido a su tamaño reducido, coste, portabilidad y sus altas capacidades de programación y conectividad. Actualmente existen 9 versiones de microordenadores Raspberry Pi con diferentes prestaciones. A continuación, se describirán, de manera comparativa, las características de los tres modelos más adecuados para el presente trabajo.

Todos ellos tienen el mismo tamaño (de una tarjeta de crédito) y requieren una alimentación de 5V y están equipadas con un puerto CSI para la conexión de la Pi cámara, así como de un puerto DSI para la conexión de una pantalla táctil. También disponen de un puerto para tarjetas Micro SD para cargar el sistema operativo y almacenar datos. El sistema operativo recomendado para un uso normal de la Raspberry es Raspberry Pi OS (anteriormente llamado Raspbian) y está basado en Linux. El lenguaje de programación recomendado para trabajar con la Pi es Python.

Además, tienen 40 pines GPIO *(general-purpose input/output)* que permiten conexiones con elementos externos de entrada y de salida para realizar un amplio abanico de funciones diversas.

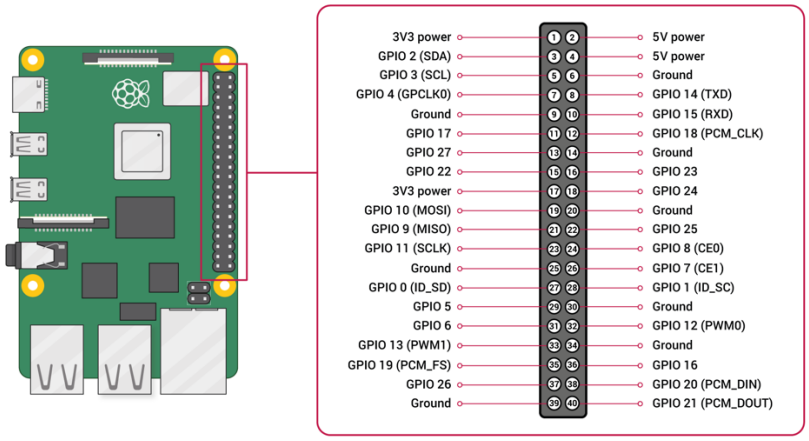
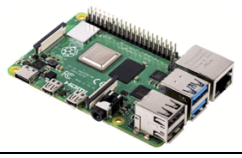
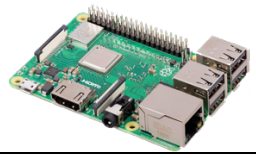


Figura 3: Descripción de los pines GPIO de la Raspberry Pi modelo 2B. Fuente [24].

En la página web de la fundación (https://www.raspberrypi.org) se pueden consultar toda la información relativa a los productos Raspberry Pi. En la siguiente tabla resumen, quedan recogidas las principales diferencias entre los tres modelos de placa principales.

Tabla 1: Tabla comparativa de las características de los tres modelos principales de Raspberry Pi

	Raspberry Pi 4B	**Raspberry Pi 3B+**	**Raspberry Pi 2B**
Precio	Desde 44€	Desde 43€	Desde 43€
HDMI	2 puertos 4K (pantalla dual)	1 puerto HDMI *(full-size)*	1 puerto HDMI *(full-size)*
RAM	2GB, 4GB o 8GB	1GB	1GB
Bluetooth	5.0	4.2 / BLE	NO
Ethernet	Gigabit Ethernet	Gigabit Ethernet	Ethernet RJ45 10/100
Wifi	2 puertos de 2.4/ 5.0 GHz	2 puertos de 2.4/ 5.0 GHz	NO
USB	2 puertos USB2 2 puertos USB3	4 puertos USB 2.0	4 puertos USB 2.0
Unidad central de Procesamiento (CPU)	Broadcom BCM2711, ARM v8, Quad-core Cortex-A72 64bit SoC @ 1,5GHz	Broadcom BCM2837B0, Quad-core Cortex-A53 64bit SoC @ 1,4GHz	900MHz quad-core ARM Cortex-A7
Placa			

Como queda reflejado en la tabla, la última versión de Raspberry ha supuesto un aumento significativo en la memoria, velocidad del procesador, el rendimiento multimedia y la conectividad, manteniendo el mismo precio (para la memoria RAM de 2GB). Destaca también la novedad que la versión 3 incorporó en cuanto a la transmisión inalámbrica (wifi y bluetooth), ausente en la versión 2. A pesar de las ventajas que brinda la versión más actual de Raspberry Pi, en este TFG se ha trabajado con el modelo 2B debido a que era el que se disponía en el laboratorio y cumple con los requisitos mínimos para el desarrollo del proyecto.

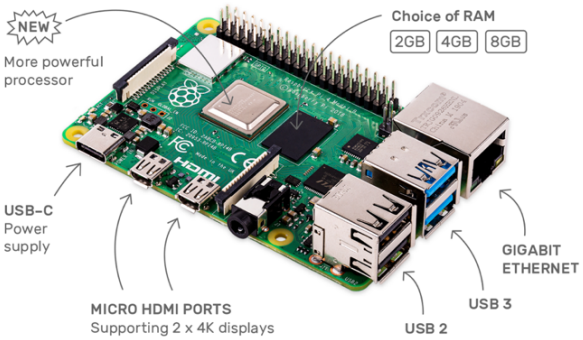


Figura 4: Esquema de las partes de la monoplaca Raspberry Pi 4B. Principales características.

Una de las utilidades más destacables de este producto es la posibilidad de añadir complementos que amplían los usos que se puede hacer del mismo. Mencionaremos dos que han sido esenciales en este trabajo.

- **PiCamera**: Actualmente, la fundación Raspberry Pi comercializa un módulo que dispone de un sensor Sony IMX219 que ofrece una resolución de 8 megapíxeles por un precio de 28€. Existen numerosas librerías que permiten el manejo eficiente de la cámara, entre las que destaca la librería *PiCamera* de Python. También cabe la posibilidad de trabajar con una cámara externa conectada vía USB, o incluso pueden utilizarse ambas.

  En este proyecto se ha trabajado con el modelo PiNoIR, que se diferencia del módulo normal por no disponer de un filtro infrarrojo, de forma que permite la visión nocturna. Se trata de la versión antigua, que presenta un sensor OmniVision OV5647 de 5 megapíxeles.

Figura 5: Raspberry Pi 2B con la cámara PiNoIR conectada. Fuente: página web de la fundación Raspberry Pi.

- **Pantalla:** Es un monitor táctil de 7 pulgadas con una resolución de 800 x 480 píxeles que permite utilizar la Raspberry como una tableta. Se conecta vía DSI y necesita alimentación de 5V.

Figura 6: Pantalla oficial Raspberry Pi. Fuente: página web de la fundación Raspberry Pi.

- **Cargador**: La placa viene con un cargador de micro USB que la alimenta con 2.5A. Puede sustituirse por una batería portátil.

2.4. DIGISPARK

Digispark es una placa de desarrollo basada en el microcontrolador ATTINY85 de ATMEL comercializada por la empresa Digistump LLC [25]. Presenta un USB integrado en la placa y es fácil de programar puesto que es compatible con el IDE de Arduino. Tiene 6 pines de entrada salida, es de pequeño tamaño y de bajo costo (aproximadamente 2€), lo que lo hace una buena opción para realizar funciones sencillas. Se puede comunicar con la Raspberry mediante el protocolo I^2C.

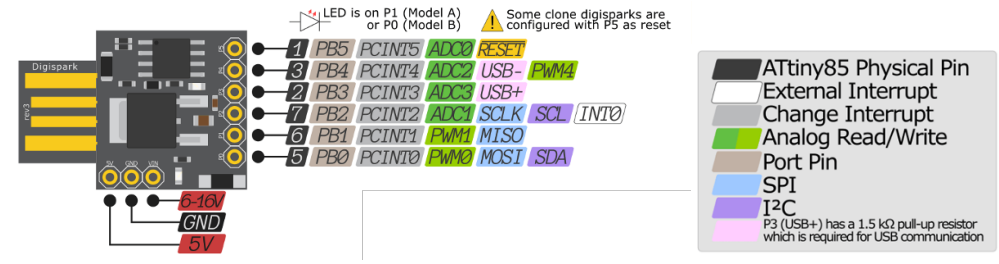


Figura 7: *Digispark Pinout. Fuente: [26]*

2.5. ARDUINO IDE

Para la programación de las placas Digispark se requiere la utilización del entorno de desarrollo de Arduino, el Arduino IDE *(Integrated Development Environment)*. Esta herramienta permite la creación de "*Sketches*", que son programas que se cargarán en las placas utilizadas y que definirán las tareas que han de realizar. La carga del programa en la Digispark se realiza a través la conexión USB con el ordenador.

Trabajar con el entorno de Arduino ofrece la ventaja de poder utilizar gran parte de sus librerías, así como de otras propias disponibles en la wiki de Digistump. El IDE de Arduino es multiplataforma y se puede descargar de forma gratuita desde su página web. Su popularidad hace que existan en Internet numerosos tutoriales y foros de discusión que facilitan el desarrollo de *sketches* y la solución de posibles problemas.

2.6. VISIÓN ARTIFICIAL

Según la Asociación de Imágenes Automatizadas (AIA), la visión artificial "incluye todas las aplicaciones industriales y no industriales en las que una combinación de hardware y software proporciona orientación a diversos dispositivos, por medio de la captura y el procesamiento de imágenes, para la ejecución de sus funciones" [27]. Así pues, esta disciplina permite la extracción de información a partir de imágenes y vídeos mediante la aplicación de técnicas de Inteligencia Artificial como el procesamiento de imágenes, reconocimiento de patrones y diferentes procesos de *machine learning* o *deep learning* [28].

Las tres tareas más frecuentes de la visión artificial son: reconocimiento y clasificación de objetos, detección de movimiento y análisis y reconstrucción de imágenes. En este TFG se explotarán las dos primeras para detectar gestos y traducirlos en acciones. Existen diversas librerías para la implementación de algoritmos de visión artificial. Para este proyecto se han utilizado OpenCV y dlib, dos librerías de código abierto que permiten programación con Python y son compatibles con el sistema operativo de la Raspberry Pi.

2.6.1. OpenCV

Es una librería multiplataforma de visión por computador y *machine learning* sacada al mercado bajo la licencia BSD (Berkeley Software Distribution), por lo que es gratis para uso académico y comercial. Fue creada con la finalidad de proporcionar una infraestructura común para este tipo de aplicaciones que acelerara su uso en productos comerciales. Está desarrollada en lenguaje C/C++ altamente optimizado y orientado a la eficiencia computacional, enfocada para aplicaciones en tiempo real. Incluye más de 2500 algoritmos optimizados de visión artificial, tanto clásicos como del estado del arte. Tiene interfaces de C++, Python, Java y MATLAB y soporta Windows, Linux, Android and Mac OS [29].

2.6.2. Dlib

Es una librería escrita en C++ que contiene algoritmos de *machine learning* y herramientas que permiten desarrollar software para solventar problemas del mundo real. Tiene licencia de código abierto, por lo que se puede utilizar gratuitamente para fines académicos y comerciales (la documentación puede consultarse en http://dlib.net). Se puede utilizar en Python y es compatible con la Raspberry Pi, aunque para su instalación requiere mucha memoria, lo que supone un problema para los modelos de Raspberry con 1GB de RAM, que se soluciona modificando las opciones de memoria por defecto *(swap space y GPU/RAM Split)*.

En este TFG, se va a hacer uso del detector de marcas faciales que ofrece esta librería. Se trata de una implementación del artículo publicado por Kazemi y Sullivan en 2014 llamado "*Millisecond Face Alignment with an Ensemble of Regression Trees"* [30]. En él, se propone una solución óptima para el problema de la alineación facial para una sola imagen, basada en la utilización de un conjunto de árboles de regresión que estiman las posiciones de los puntos de referencia del rostro a partir de un subconjunto de intensidades de píxeles escaso (no se lleva a cabo extracción de características). Para ello, se parte de una colección de imágenes etiquetadas manualmente con las coordenadas de las marcas faciales que se utiliza como datos de entrenamiento para los árboles de regresión. Además, se tienen las probabilidades a priori de las distancias entre pares de píxeles de entrada, cuya inicialización adecuada contribuye positivamente para la selección de características eficiente. El resultado es una cascada de regresiones que son capaces de localizar las marcas faciales en tiempo real con predicciones de alta calidad.

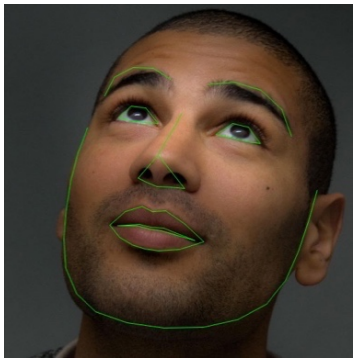


Figura 8: *Resultado de la aplicación del algoritmo de detección de marcas faciales de dlib en una imagen de la base de datos HELEN. Fuente: [31]*

El detector de marcas faciales pre entrenado que está implementado en la librería dlib consigue mapear 68 puntos que definen las estructuras faciales (ojos, cejas, nariz, boca y mandíbula). Estos puntos se obtienen entrenando un predictor con el conjunto de datos etiquetados de iBUG 300-W [32]. Existen otras bases de datos con las que se puede entrenar al predictor como, por ejemplo, la base de datos HELEN [33], que ofrece un modelo con 194 puntos. Además, dlib también permite entrenar detectores con una base de datos propia.

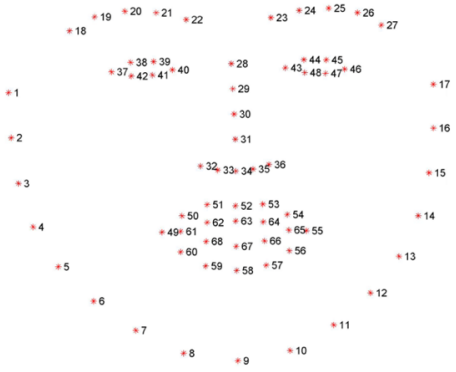


Figura 9: Visualización de las 68 coordenadas de marcas faciales que devuelve la base de datos iBUG 300-W. Fuente: [34].

2.6.3. imutils

El paquete *imutils* (disponible en https://github.com/jrosebr1/imutils), creado por Adrian Rosebrock, ofrece una serie de funciones de visión artificial y procesamiento de imágenes que facilitan el trabajo con OpenCV. Concretamente, el submódulo *face_utils.py* ha sido diseñado específicamente para facilitar el tratamiento de marcas faciales con dlib. Contiene un diccionario que mapea los puntos faciales con las diferentes estructuras del rostro, haciendo corresponder, por ejemplo, los puntos del 48 al 68 con la boca o del 27 al 36 con la nariz. Además, dispone de funciones que permiten trabajar con los puntos de manera eficaz para poder hacer mediciones.

2.7. KIVY

Es una librería de Python de código abierto que sirve para el desarrollo rápido de aplicaciones que utilizan interfaces de usuario innovadoras [35]. Es multiplataforma ya que opera en Linux, Windows, OS X, Android, iOS, y Raspberry Pi. Es de utilización gratuita para uso académico y comercial ya que tiene licencia MIT. Una ventaja que brinda el uso de esta librería es que dispone de una documentación muy completa que facilita su uso, así como guías de programación y ejemplos para principiantes. Además, dispone de un gran número de widgets que hacen de kivy una librería adecuada para aplicaciones muy diversas.

2.8. ENTORNO VIRTUAL

Un entorno virtual puede definirse como un directorio de instalación aislado, lo que permite restringir todas las instalaciones requeridas para un proyecto en un lugar concreto, sin necesidad de instalarlas en todo el sistema. Esto facilita el manejo de diferentes versiones de librerías y herramientas y evitar conflictos de compatibilidad a la hora de desarrollar y lanzar el proyecto [36].

Para el desarrollo de la aplicación que se describe en este TFG, se ha creado un entorno virtual de Python, en su versión 3.7. La creación de entornos virtuales en Python se realiza utilizando el módulo *venv.* Una vez formado el entorno, se utiliza *pip* (el instalador de paquetes de Python) para instalar dentro del entorno OpenCV, kivy, imutils y dlib, y las dependencias que se requerían para su correcto funcionamiento. Los *scripts* de Python que contienen el código de la aplicación y de las funciones implementadas se han guardado en este directorio de manera que es necesario activar el entorno virtual en la línea de comandos para poder ejecutar exitosamente la aplicación.

2.9. TRANSMISIÓN INALÁMBRICA

La comunicación inalámbrica es aquella que no utiliza un medio de propagación físico para la transmisión de los datos. En su lugar, emplea la modulación de ondas electromagnéticas, que se propagan por el espacio sin un medio físico que comunique cada uno de los extremos de la transmisión. Así, los dispositivos físicos únicamente están presentes en los emisores y receptores de la señal: antenas, computadoras portátiles, PDA, teléfonos móviles, etc. [37].

Existen diferentes protocolos que permiten la comunicación inalámbrica entre dispositivos, de los que destacan Bluetooth y wifi. Ambos pueden ser adecuados para la finalidad que persigue el sistema desarrollado en es TFG. La característica distintiva de estos dos protocolos es el propósito de diseño: mientras que el Bluetooth se desarrolló fundamentalmente para conectar dispositivos entre sí a corta distancia y compartir datos, wifi proporciona acceso a Internet [38]. Finalmente, un tercer tipo de comunicación inalámbrica muy conocido es la transmisión por radio frecuencia (RF). Esta permite también la transmisión de datos entre dispositivos que se encuentran a poca distancia, y requiere de módulos emisor-receptor externos para establecer la conexión.

Como queda reflejado en la tabla 1, a partir del modelo 3, la Raspberry Pi dispone de conexión Bluetooth y Wifi incorporada. Para el caso del modelo 2B que se ha utilizado durante la realización de este proyecto, estos dos tipos de conexión inalámbrica podrían implementarse mediante un adaptador USB. Además, sería necesario también que los dispositivos receptores (las lámparas, la calefacción, el motor de la persiana, etc.) tuvieran un módulo de red que les permitiera establecer la conexión.

Sin embargo, por motivos de sencillez y compatibilidad con la placa Digispark, se ha optado por establecer una conexión por RF. Para ello, se han utilizado módulos de RF de 433MHz que se venden por pares emisor-receptor, cuyo coste no supera los 50 cts. y que son compatibles para proyectos con Raspberry Pi y Arduino. Para el control de la transmisión de los datos se ha empleado la librería MANCHESTER [39], desarrollada específicamente para este propósito con la placa Digispark.

Figura 10: Módulos de transmisión inalámbrica por RF de 433MHz, emisor (derecha) y receptor (izquierda). Fuente: naylampmechatronics[2]

2.10. COMUNICACIÓN SERIAL. PROTOCOLO I2C

I^2C *(Inter-Integrated Circuit Bus)* es un protocolo de comunicación serial que define la trama de datos y las conexiones físicas para transferir bits entre dos o más dispositivos digitales [40]. Un circuito I^2C consiste en un bus "maestro" y uno o más buses "esclavos". Es uno de los protocolos más utilizados puesto que admite hasta 127 dispositivos esclavos con los que intercambiar la información a una velocidad aceptable, de unos 100 Kbits/s y, además, su arquitectura permite tener una confirmación de los datos recibidos. El puerto utiliza cuatro conexiones [41]:

[2] Se puede consultar en https://naylampmechatronics.com/inalambrico/13-modulo-rf-433mhz.html

- SDA: línea "*Serial Data*". Es una línea de datos bidireccional que maneja las comunicaciones entre los dispositivos.
- SCL: línea "*Serial Clock*". Es la línea de los pulsos de reloj que sincroniza la transmisión de datos en la línea SDA.
- VCC: referencia de voltaje de nivel lógico, generalmente de 5 o 3,3 voltios.
- GND: Tierra

Para implementar y dirigir la comunicación I^2C se emplea la librería SMBus *(System Management Bus)* [42], que es compatible tanto con la Raspberry (en Python) como con las placas de Arduino.

2.11. DOMÓTICA

La domótica se define como "el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema" [43].

En los últimos años, la demanda de viviendas automatizadas ha aumentado considerablemente, lo que ha producido que las empresas de este sector respondan con una oferta más consolidada, dando respuesta a las nuevas tendencias en la forma de vida de la sociedad. Asimismo, una parte importante de este mercado se ha enfocado en proporcionar soluciones que aumenten la comodidad y la calidad de vida de las personas con movilidad reducida [44].

CAPÍTULO 3. OBJETIVOS

El objetivo principal de este proyecto es desarrollar un prototipo de sistema de ayuda ambiental de bajo coste, enmarcado en el ámbito de la domótica, para personas con diversidad funcional motora, que suponga una mejora de la calidad de vida del usuario y de las personas que están a su cuidado, así como un aumento de su autonomía e independencia.

Para lograr el objetivo principal, se han establecido tres objetivos secundarios. Primeramente, se pretende estudiar el potencial del miniordenador monoplaca Raspberry Pi para el procesamiento de imágenes a tiempo real y el control de dispositivos externos de forma inalámbrica. En segundo lugar, se procurará desarrollar una aplicación que muestre una interfaz simplificada y comprensible, con el objetivo de diseñar una solución útil para el mayor número de personas posible, abarcando distintos grados de afectación. Por último, se busca dotar al sistema de la capacidad de adecuarse a las necesidades específicas de cada usuario, de manera que suponga una solución relevante en su vida cotidiana.

CAPÍTULO 4. DESARROLLO DEL PROTOTIPO

4.1. FUNCIONAMIENTO GENERAL

Para el desarrollo del sistema de ayuda ambiental en el hogar se ha diseñado una aplicación que se ejecuta en la Raspberry Pi. Al abrirla, se inicia un *scroll*, de manera que, en la pantalla, se puede ver cómo se van sucediendo, de una en una, diferentes ventanas que indican las acciones que pueden ser llevadas a cabo por el usuario a través del programa. Así pues, cuando en la pantalla se muestra la actividad que el usuario desea realizar, este indicará, con una acción (elemento de activación), que se ejecute dicha orden, de modo que el programa, tras detectar la acción, enviará la señal correspondiente para ejecutar la operación que el usuario ha seleccionado (y que se muestra en pantalla en ese instante).

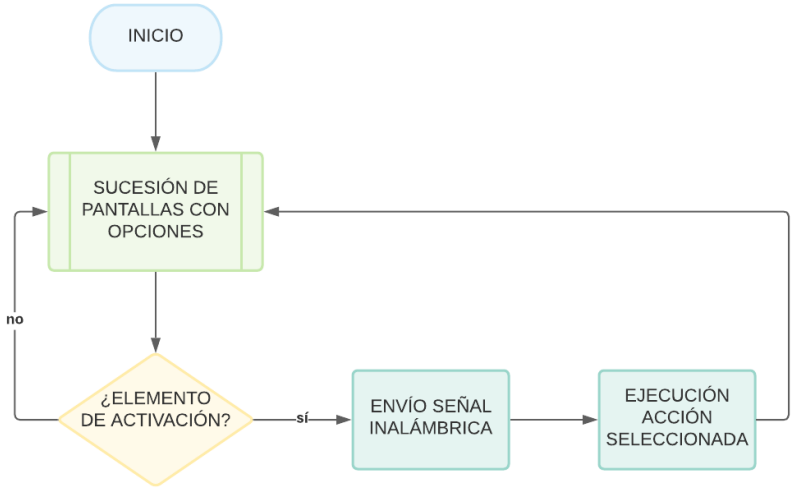


Figura 11: *Diagrama de flujo del funcionamiento general del sistema. Fuente: elaboración propia mediante la aplicación web lucid (https://lucid.co/es).*

Es importante señalar que, para cumplir con el objetivo de hacer llegar la aplicación al mayor número de personas posible, consiguiendo a la vez que la aplicación se ajuste a las necesidades específicas de cada usuario, se plantearán a lo largo del trabajo diferentes opciones de elementos de activación que podrán llevar a cabo una misma orden.

La aplicación principal está desarrollada en el script de Python llamado *main_App.py.* Para controlar el diseño de la app se ha utilizado la librería kivy. El módulo *kivy.uix.screenmanager* permite el diseño y manejo de las diferentes ventanas *(Screens)* de la app, gracias a los métodos que contiene de forma predefinida. Cada ventana es, por lo tanto, un objeto de la clase *Screen*. El aspecto y tamaño de las ventanas es fácilmente personalizable, pueden añadirse imágenes y texto, así como controlar la transición entre las mismas. Por otra parte, para la configuración de la aplicación es necesario crear un objeto de la clase App (llamado *RFDomoticsApp*), donde se añade la función *build()*, que es la que conduce la ejecución del programa.

El paso de una ventana a otra *(scroll)* puede ser o bien de forma automática (barrido continuado), o bien llevado a cabo de manera voluntaria por el usuario, de manera que la ventana quede fija hasta que el usuario decida cambiar a la siguiente. En el primer caso, conviene ajustar el tiempo de espera requerido para pasar a la siguiente ventana, teniendo en cuenta la velocidad de respuesta de cada usuario y las prestaciones del sistema. Para ello, se define la función *cambiar_ventana()*, que lee el nombre de la ventana actual y efectúa el cambio a la siguiente. El módulo *kivy.clock* contiene un método que permite llamar, desde la función principal *build(),* a dicha función repetidamente cada cierto intervalo de tiempo modificable. En el segundo caso, la aplicación esperará hasta detectar el elemento de activación definido para efectuar el cambio de ventana.

A continuación, veremos en detalle el proceso de captura de los diferentes elementos de activación que se han diseñado, así como el mecanismo de envío de señales inalámbricas para la ejecución de la acción escogida.

4.2. ENTRADA. ELEMENTOS DE ACTIVACIÓN

Para el diseño y la implementación de los elementos de activación que el sistema ha de captar para efectuar las órdenes necesarias, se han definido una serie funciones que, en términos generales, operan de la misma manera, la cual consiste en estar leyendo permanentemente una señal de entrada, a la espera de que tome un valor determinado que le indique que debe ejecutar la acción seleccionada. Esta espera supone un problema, puesto que solo se dispone de un procesador que ejecuta una tarea detrás de otra, de modo que, cuando se llame a una de estas funciones, la aplicación quedaría bloqueada aguardando la señal indicada.

Con el fin de hacer compatible la ejecución normal de la aplicación con la implementación de las funciones encargadas de captar los elementos de activación, se ha hecho uso de la programación con hilos *(threading)*, que permite que un programa ejecute múltiples operaciones simultáneamente en el mismo espacio de proceso [45].

Un hilo es un flujo de ejecución diferente. Realmente, los diferentes hilos no funcionan al mismo tiempo, ya que no es posible porque para ello se requeriría más de un procesador, como se ha mencionado con anterioridad. Lo que ocurre es que el programa intercala la ejecución de los diferentes hilos de manera muy rápida de modo que parece que los dos flujos de ejecución estén trabajando al mismo tiempo (figura 12, derecha). Esto permite que, mientras una función espera a recibir una señal de entrada, la aplicación continúe funcionando con normalidad [46].

A la hora de implementar los hilos, se utiliza la librería *Threading* de Python. A cada objeto de clase hilo *(thread)* creado, se le asigna una función diana *(target)*, que se ejecutará cuando se indique el comienzo del hilo mediante el método *start()*, al cual se le llama dentro de la función principal *build()* explicada en el apartado anterior.

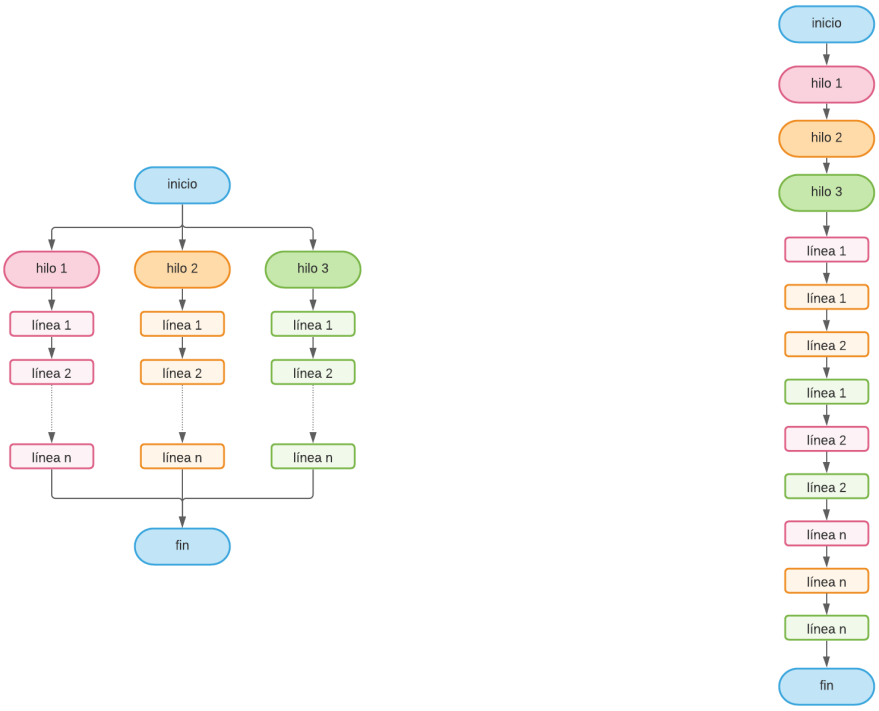


Figura 12: *Flujo de funcionamiento de la programación con hilos teórico (izquierda) y real (derecha).*

Para ganar claridad en el diseño del código, y facilitar la fase de personalizar la aplicación a las necesidades del usuario, se ha creado un hilo por cada elemento de activación que ha de ser detectado seguido siempre la misma estructura:

Por un lado, existen dos acciones que el usuario puede realizar, que son cambiar la ventana y enviar una señal por RF. Estas dos acciones se han descrito en dos funciones sencillas: *cambiar_ventana()* (mencionada previamente) y *rf_envia()* (que se encarga de la transmisión por RF y cuyo funcionamiento se detallará en el siguiente apartado). Por otro lado, se ha creado una función específica para cada elemento de activación diseñado, que devuelve una variable booleana llamada ELEM_ACTIVADO. Esta será positiva *(True)* si el algoritmo detecta el movimiento esperado, es decir, si el elemento de activación se ha activado.

Así pues, se implementan dos hilos, cuya función diana llama, primeramente, a la función que detecta el elemento de activación y, cuando recibe un valor positivo, llama a la función que realiza la acción deseada por el usuario (enviar señal o cambiar de ventana).

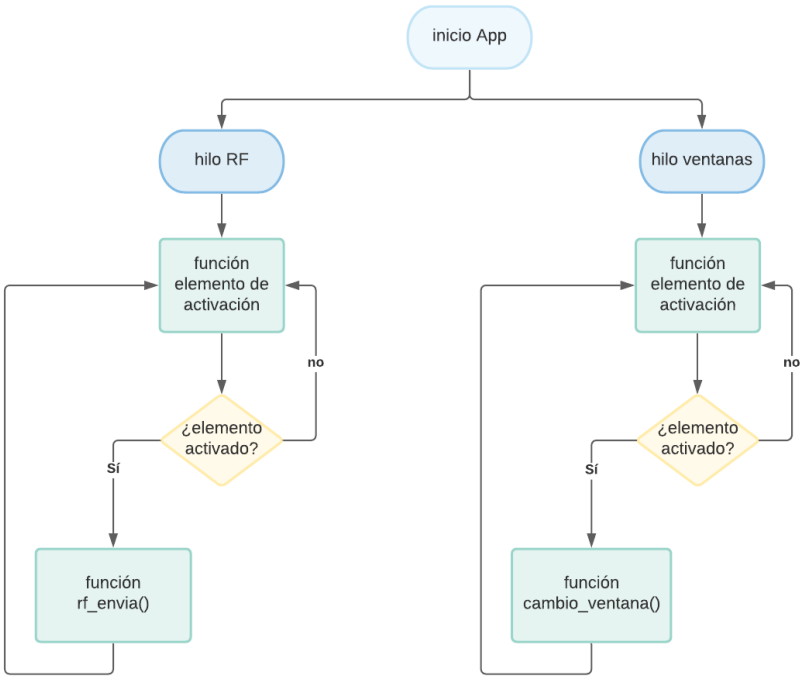


Figura 13: Diagrama de flujo del funcionamiento de la App mediante la programación con hilos.

Debe tenerse en cuenta que, si se decide implementar la opción de cambiar la pantalla automáticamente, el hilo "ventanas" no se ejecutará. En los siguientes apartados, se describirán las diferentes funciones que se han definido para capturar los elementos de activación.

4.2.1. Detección de elementos de activación mediante visión artificial

La librería OpenCV contiene métodos para controlar la toma de imágenes desde la cámara. En este proyecto, se requiere la grabación de un vídeo, a partir del cual se realicen una serie de cálculos que permitan detectar ciertos gestos. Para ello, primeramente, se activa la cámara con el método *VideoStream.start()* y, si es necesario, se ajusta el tamaño de la imagen gracias a la función *resize()* de la librería *imutils*. Para leer los sucesivos fotogramas, es necesario iniciar un bucle infinito *(while True:)*, dentro del cual se llamará al método *cv2.read()*, con lo que, en cada iteración, se realizará la lectura de un fotograma. Antes de proceder al procesamiento de las imágenes para la detección de movimientos, conviene realizar un preprocesado que permita minimizar el ruido y optimizar los resultados.

Los fotogramas captados por la cámara devuelven una imagen en formato RGB, en el que se tienen tres matrices, una por cada canal (*Red-Green-Blue*). La suma aditiva de los tres canales resulta en una imagen a color. No obstante, dado que el color no proporciona ninguna información relevante, inmediatamente después de su lectura, los fotogramas se convierten de RGB a escala de grises, mediante el método *cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY*). Así, se tiene una única matriz de píxeles, con lo que se simplifica considerablemente el número de operaciones a realizar y, por tanto, se mejora el rendimiento del programa.

Seguidamente, se ha aplicado un filtro gaussiano, es decir, un filtro paso bajo que consigue reducir las componentes de alta frecuencia de la imagen utilizando para ello una convolución con una función Gaussiana [47]. Esta técnica de desenfoque da más peso a los píxeles vecinos en el cálculo del valor final de cada píxel. Concretamente, se ha utilizado para la convolución un kernel 9x9.

Finalmente, las líneas de código sucesivas se encargarán de procesar las imágenes y detectar los movimientos específicos definidos como elementos de activación. En este trabajo, se proponen tres elementos de activación diferentes, que son detectados gracias a la implementación de un algoritmo específico para cada uno de ellos. A continuación, se detalla el contenido de cada una de las funciones diseñadas para tal fin.

A. Función de detección de movimiento

Se pretende detectar un movimiento realizado en dirección horizontal con respecto a la cámara. Este movimiento puede ser llevado a cabo con las extremidades superiores del usuario o incluso con la cabeza. Para ello, se ha creado un script, llamado *detecta_movimiento()*, que contiene dos funciones: una que detecta los movimientos hacia la derecha, *dir_derecha()*, y otra, que los detecta hacia la izquierda, *dir_izquierda()*.

El movimiento que se quiere detectar no está bien definido, puesto que cada persona lo realizará de una forma concreta de acuerdo con su condición física. En este contexto, la técnica de sustracción de fondo resulta adecuada y robusta. Esta técnica, ampliamente utilizada en aplicaciones de visión artificial, consiste en el cálculo de la diferencia entre dos imágenes, y la posterior aplicación de un umbral *(threshold)* de forma que se genere como resultado una máscara binaria cuyos píxeles blancos (de intensidad 255) sean aquellos en los que la imagen ha cambiado, es decir, los correspondientes a la zona donde ha habido movimiento (figura 14). Para la determinación del umbral, es importante encontrar el equilibrio entre sensibilidad y especificidad (que sea lo bastante alto como para descartar movimientos no representativos, pero sin llegar a omitir los movimientos que se quieren detectar). Con esta consideración, se ha optado por establecer el valor del umbral en 20.

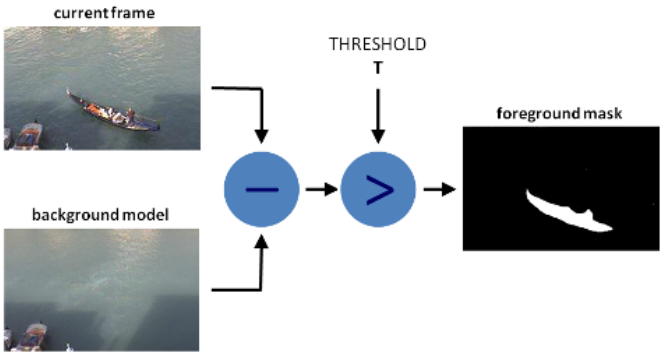


Figura 14: *Esquema del funcionamiento del algoritmo de sustracción de fondo. Fuente: [48].*

La forma básica de aplicación de esta técnica consiste en obtener una imagen de referencia (la imagen de fondo) a la que se le restan los sucesivos fotogramas *(frames)* que la cámara va adquiriendo durante la ejecución del programa. Sin embargo, esta no resulta una metodología práctica, puesto que es muy sensible a cambios de luminosidad e incompatible con posibles recolocaciones de la cámara, que suponen un cambio del fondo. En vistas a solucionar estos problemas, OpenCV dispone de tres algoritmos que consiguen actualizar el fondo de la imagen continuamente, logrando de este modo una implementación más robusta. Estos son:

- *Background sustractor MOG:* Este algoritmo de segmentación, introducido en [49] busca modelar cada píxel del fondo de la imagen mediante una mezcla de K distribuciones gaussianas (K = 3 a 5). Los pesos que se le atribuyen a cada distribución representan las

proporciones de tiempo que cada color permanece en la escena, de modo que los colores del fondo serán, con mayor probabilidad, los que permanezcan más tiempo y más estáticos [50].

- *Background sustractor MOG2:* Se trata de una actualización del anterior algoritmo, desarrollada a partir de dos artículos [51], [52]. También está basado en una mezcla gaussiana, pero consigue proporcionar una mejor adaptabilidad a diferentes escenas debido a cambios de iluminación. Además, es capaz de detectar también las sombras de los objetos en movimiento.

- *Background sustractor GMG:* Está basado en un artículo más reciente [53], que propone combinar la estimación estadística de la imagen de fondo y la segmentación bayesiana por píxel. Para modelar el fondo, utiliza los primeros fotogramas (por defecto 120), por lo que durante los primeros fotogramas se obtiene una imagen negra. Después, emplea un algoritmo probabilístico de segmentación que identifica posibles objetos en primer plano mediante inferencia bayesiana. Para conseguir una adaptación a la iluminación variable se les otorga más peso a las observaciones más recientes. Finalmente, se llevan a cabo operaciones de filtrado morfológico (cierre y apertura) para eliminar ruido [50].

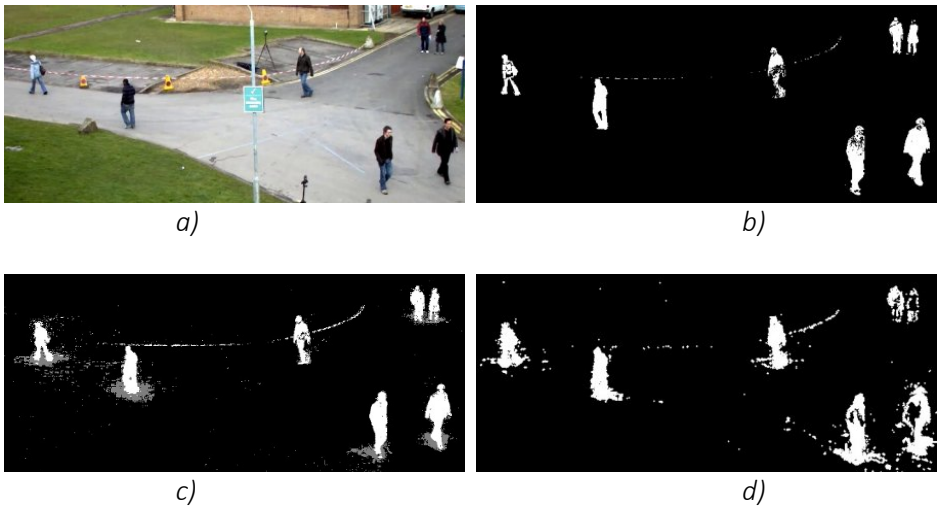


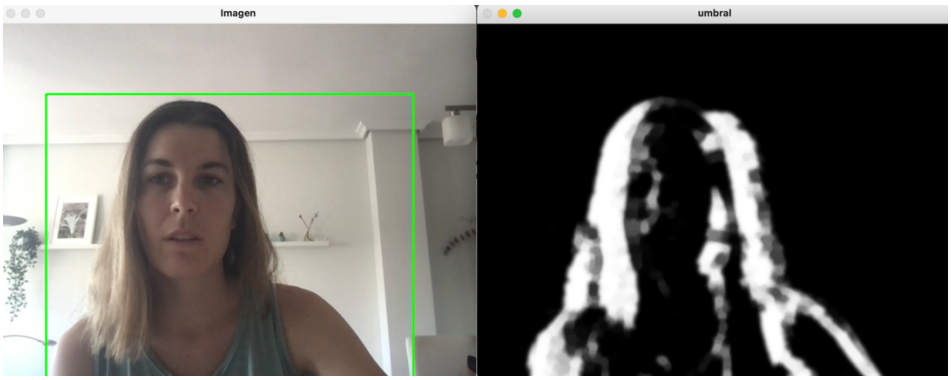
Figura 15: *Resultados de la aplicación de los algoritmos de sustracción de fondo implementados en OpenCV a) imagen original. b) MOG. c) MOG2. d) GMG.*

Se trata de algoritmos con gran potencial, que permiten reajustar el fondo de la imagen a tiempo real de manera automática. Para la aplicación desarrollada en este trabajo, se decide implementar el método MOG2 (sin detección de sombras), puesto que se requiere una detección lo más nítida posible y un reajuste del fondo rápido frente a las posibles variabilidades. Además, es conveniente minimizar los errores debido a los cambios de iluminación. La implementación de estos algoritmos solo requiere de una línea de código:

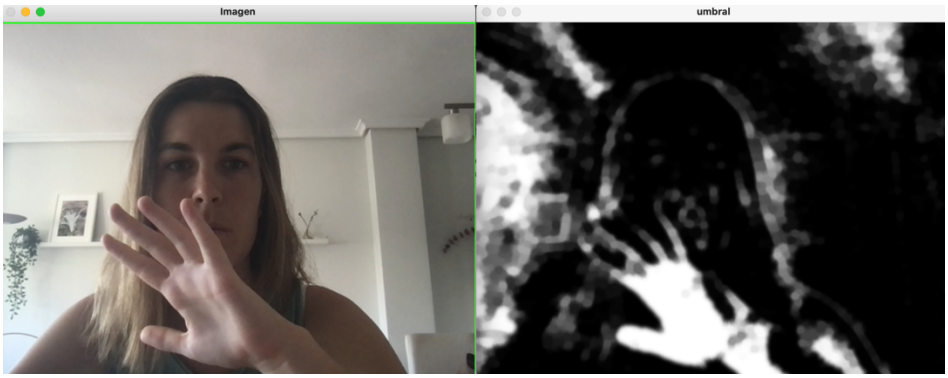
```
fgbg = cv2.createBackgroundSubtractorMOG2(detectShadows=False)
```

Tras la aplicación de este método, se observan tres inconvenientes. En primer lugar, es capaz de detectar movimientos muy sutiles, lo que supone un problema puesto que la cámara va a estar enfocando a una persona que, inevitablemente, estará en movimiento debido a la respiración, el parpadeo o el habla. Esto conlleva que la silueta de la persona siempre aparezca marcada en la

máscara binaria. En segundo lugar, estos algoritmos tardan unos segundos en reajustar el fondo tras un movimiento. De esta manera, si la cabeza se mueve, el centro de la imagen binaria permanecerá blanco durante unos segundos después de que el movimiento haya cesado, detectando movimiento cuando no lo hay (figura 16 a). Por último, se trata de un algoritmo muy sensible a los cambios de iluminación, que se detectan como movimiento en toda la imagen, dando lugar a un número considerable de falsos positivos (figura 16 b). Estas circunstancias desembocan en un número alto de falsos positivos que hacen inviable la opción de utilizar estos algoritmos.



a)



b)

Figura 16: *Resultados de la implementación del algoritmo MOG a) tras mover ligeramente la cabeza hacia la izquierda. b) errores de detección debidos a cambios en la iluminación al pasar la mano por delante de la cabeza.*

Por lo tanto, es necesario hallar una alternativa que actualice la máscara binaria en cada iteración, de manera que los píxeles blancos correspondan a las zonas de la imagen que están en movimiento en ese mismo instante de tiempo. Además, es necesario descartar el ruido generado por movimientos ligeros del usuario o por cambios en la iluminación. La solución propuesta en este TFG consiste en realizar la diferencia entre un fotograma y el inmediatamente anterior, consiguiendo así descartar cualquier posible ruido que haya quedado de movimientos detectados con anterioridad que han quedado dentro del encuadre. La diferencia se calcula mediante la función *cv2.absdiff* y, a la imagen resultante, se le aplica el umbral de 20 con la función *cv2.inRange*, que aplica la siguiente ecuación para obtener así la máscara binaria.

$$s = \begin{cases} 0, & 0 \leq r \leq Th \\ 255, & Th < r \leq 255 \end{cases} \quad (1)$$

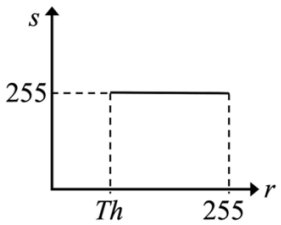


Figura 17: Gráfica del umbral simple aplicado. Fuente: [54].

Seguidamente, se han aplicado técnicas de posprocesado a la imagen para obtener un mejor resultado y reducir el ruido. Conviene resaltar que las técnicas empleadas no buscan obtener una imagen nítida, sino simplemente conseguir detectar un movimiento llevado a cabo de manera evidente y voluntaria por el usuario (movimiento de la mano o de la cabeza). El posprocesado realizado ha consistido en la implementación de operadores morfológicos.

El tratamiento morfológico consiste en el procesado no lineal de imágenes binarias o de grises mediante operaciones de máximos y mínimos. Está basado en la topología de conjuntos y consiste en la aplicación de un elemento estructurante (EE) de forma conocida, que se desplazará a lo largo de la imagen de entrada. El EE es una matriz que identifica el píxel de la imagen que se está procesando y define la vecindad utilizada en el procesamiento de cada píxel [55].

Las operaciones morfológicas más básicas son:

- Dilatación: se desplaza el EE transpuesto y se calcula el supremo de los píxeles de la imagen bajo el mismo. Da lugar a figuras más gruesas y rellena agujeros.

$$Y = \delta_B(X) = \sup\{X_b, b \in B\} = \{B_x, x \in X\} \quad (2)$$

$$\delta_B(X) = X \oplus B \quad (3)$$

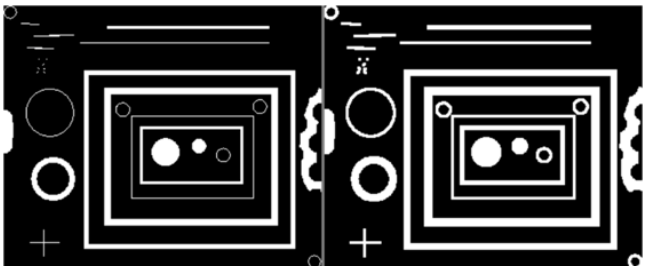


Figura 18: Ejemplo de dilatación de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras la dilatación. Fuente: [56]

- Erosión: se desplaza el EE y se calcula el ínfimo de los píxeles de la imagen bajo el mismo. Da lugar a figuras más finas y elimina objetos pequeños.

$$Y = \varepsilon_B(X) = \inf\{X_b, b \in B^*\} = \{z, B_z \in X\} \quad (4)$$

$$\varepsilon_B(X) = X \ominus B \quad (5)$$

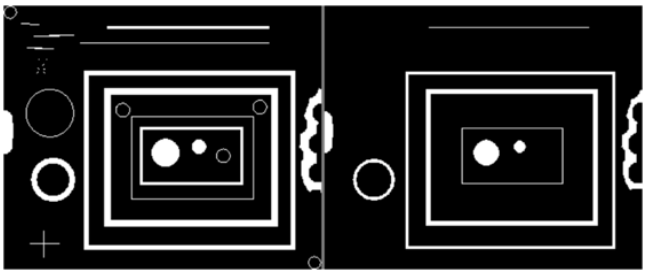


Figura 19: Ejemplo de erosión de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras la erosión. Fuente: [56]

La combinación de la dilatación y la erosión da lugar a operaciones morfológicas más complejas que permiten un mejor procesamiento de la imagen:

- Apertura: erosión seguida de dilatación utilizando el mismo EE. Elimina los objetos pequeños conservando la forma y el tamaño de los objetos más grandes.

$$\gamma_B(X) = \delta_B(\varepsilon_B(X)) \tag{6}$$

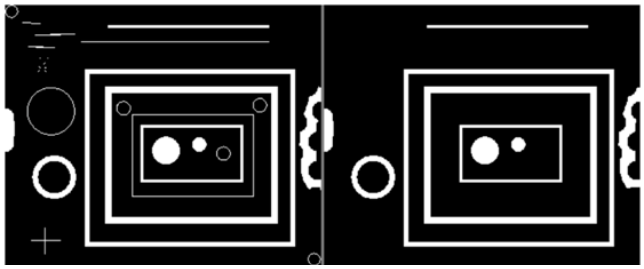


Figura 20: Ejemplo de apertura de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras la apertura. Fuente: [56]

- Cierre: dilatación seguida de erosión con el mismo EE. Rellena los agujeros conservando la forma y el tamaño de los objetos de la imagen.

$$\varphi_B(X) = \varepsilon_B(\delta_B(X)) \tag{7}$$

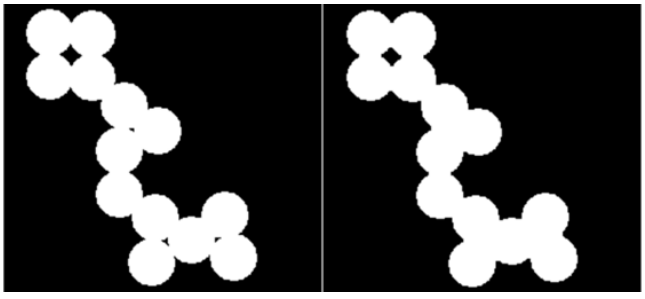
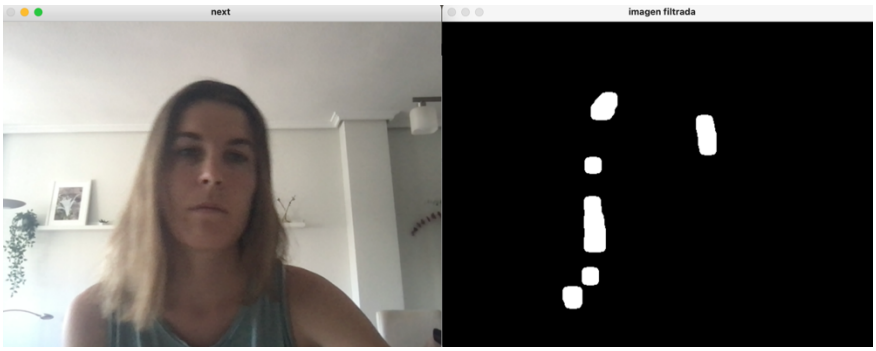


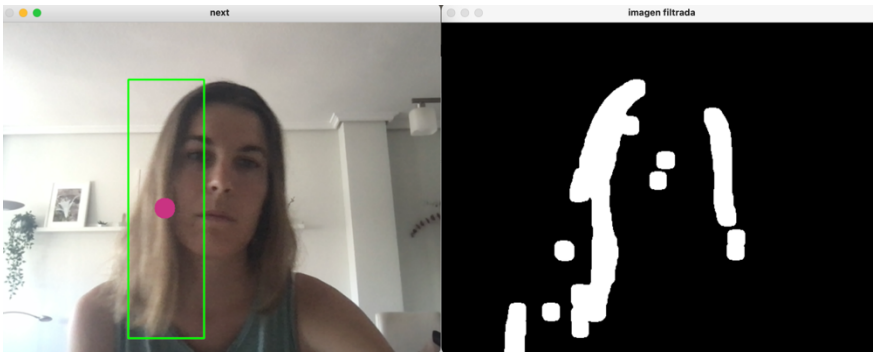
Figura 21: Ejemplo de cierre de una imagen binaria. A la izquierda la imagen original y a la derecha la imagen procesada tras el cierre. Fuente: [56]

Para la detección del movimiento de la aplicación desarrollada en este TFG, se ha realizado una apertura que elimine el ruido (es decir, los movimientos detectados de pequeño tamaño) y un

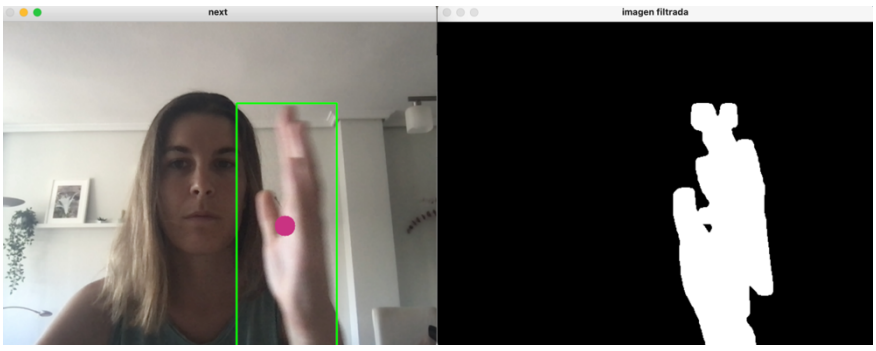
cierre que tape los agujeros. En ambos casos, se ha utilizado un EE con forma de elipse, puesto que no existe ninguna dirección de preferencia. El tamaño del EE utilizado en la apertura es de 15x15, y el del cierre, 30x30. El gran tamaño de este último es debido a que el algoritmo implementado detecta principalmente los contornos de los objetos en movimiento, puesto que su interior es homogéneo y, por tanto, no se aprecia el cambio en esos píxeles al hacer la diferencia entre dos fotogramas consecutivos. Con un *kernel* grande, se consigue rellenar gran parte del área interior, obteniendo un objeto de mayor tamaño que es más fácil de detectar. Finalmente, se dilata la imagen con un EE de tamaño 7x7 para engrosar la figura.



a)



b)



c)

Figura 22: *Resultado de la aplicación del algoritmo de detección de movimiento desarrollado a) tras mover ligeramente la cabeza, aunque detecta cambios, no lo considera como movimiento por no ser significativos. b) tras mover la cabeza intencionadamente, el área de píxeles grandes es lo bastante grande como para considerarse movimiento. c) tras mover la mano por delante de la cabeza, detecta el movimiento correctamente.*

Como se ve en la figura 22, tras aplicar el algoritmo descrito se consigue una imagen binaria en la que los píxeles en blanco corresponden al objeto que está en movimiento en ese instante. Para detectar la dirección en la que el objeto se está moviendo, se requiere obtener sus coordenadas. Para ello, se localizan los contornos del área blanca con *cv2.findContours()* y se calcula su *bounding box* (rectángulo que engloba al objeto, marcado en verde en las imágenes de la izquierda de la figura 22), que nos permite obtener las coordenadas de las cuatro esquinas. A partir de estas, estimamos las coordenadas del centro del objeto en cada fotograma, que se ha marcado con un punto rosa. Finalmente, se define una línea vertical en el centro de la imagen y se guarda el valor del centro en el fotograma actual y en el anterior. Así, comparando la posición de ambos centros con respecto a la línea central de la imagen, se puede detectar cuándo el objeto ha cambiado de un cuadrante a otro y en qué sentido (izquierda o derecha). Las funciones *dir_izquierda()* y *dir_derecha()* son exactamente iguales, salvo que en una de ellas se ha trabajado con el espejo de la imagen capturada, obtenido mediante la función *cv2.flip().*

B. Detección de gestos faciales

En este TFG, se han definido dos gestos como posibles elementos de activación: el pestañeo (con ambos ojos) y la apertura de la boca. Para la detección de cualquier gesto facial, el primer paso es delimitar el área de la imagen correspondiente al rostro de la persona. Una vez detectada, se aplicará sobre esta el predictor de marcas faciales de la librería *dlib* [57], que nos permitirá localizar las estructuras faciales y cuantificar los movimientos que actuarán de elementos de activación en sistema desarrollado en este TFG.

Tanto OpenCV como dlib contienen algoritmos de detección de rostros. Una primera propuesta, consiste en la localización de objetos mediante el uso de la técnica del histograma de gradientes orientados (HOG, por sus siglas en inglés *Histograms of Oriented Gradients*), un descriptor muy utilizado en el procesamiento de imágenes para la detección de objetos, junto con máquinas de vectores de soporte (SVM, de *Support Vector Machine*), un algoritmo de aprendizaje supervisado empleado en problemas de clasificación [58]. Como se demuestra en [59], con esta combinación se consigue entrenar clasificadores altamente precisos. El principal inconveniente es que su coste computacional es elevado, lo que lo hace una mala opción para trabajar en la Raspberry.

Por esta razón, la detección de rostros se ha llevado a cabo mediante la implementación del clasificador *Haar Cascade* de OpenCV preentrenado para esta finalidad. Está basado en el método propuesto por Paul Viola y Michael Jones en [60], que consiste en entrenar una función en cascada mediante aprendizaje automático con un gran número de imágenes positivas (en este caso de rostros) y negativas (que contienen otro tipo de objetos) [61]. A pesar de ser menos preciso, se ha estimado oportuno utilizar este algoritmo por ser más rápido y porque, en el caso de aplicación concreto que se describe en este proyecto, la cámara estará siempre enfocando el rostro del usuario, por lo que disminuye la posibilidad de obtener falsos positivos.

Nada más iniciar el programa, se debe cargar un archivo XML llamado "haarcascade_frontalface_default.xml"[3], que contiene el clasificador *Haar Cascade* preentrenado, mediante el método *cv2.CascadeClassifier('ruta_archivoXML').* Asimismo, el predictor de marcas faciales de dlib también ha de inicializarse haciendo uso de un fichero[4] que se carga mediante el método *dlib.shape_predictor('ruta_fichero.dat')*. Una vez han sido cargados, las funciones podrán hacer uso de los mismos para detectar los gestos faciales definidos.

[3] Disponible en https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

[4] Disponible en https://github.com/davisking/dlib-models#shape_predictor_68_face_landmarksdatbz2

Independientemente del gesto facial que se desea detectar, el primer paso a realizar por las funciones es obtener las coordenadas del *bounding box*, que define la región de interés (ROI, *Region Of Interest*), a continuación, aplicar el detector de marcas faciales de dlib para obtener los 68 puntos que definen las estructuras del rostro. A partir de aquí, con ayuda de la librería *imutils* se puede trabajar con estos puntos para obtener su posición relativa y detectar gestos. Cabe mencionar que el código de las dos funciones que se van a explicar a continuación está basado en los tutoriales del blog de visión artificial "Pyimagesearch" llevado por el doctor Adrian Rosebrock [62].

Función detecta_Pestañeo()

Cada ojo está representado por 6 puntos de coordenadas (del 37 al 42 para el ojo izquierdo, y del 43 al 48 para el derecho) como se muestra en la siguiente imagen.

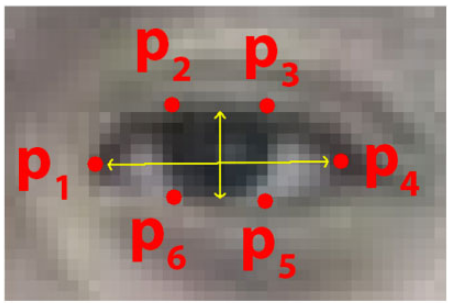


Figura 23: *Las 6 marcas faciales que definen el ojo. Fuente: [63]*

Se considera que la persona ha pestañeado cuando cierra los dos ojos durante un instante breve de tiempo (medido en fotogramas) y los vuelve a abrir. Para poder cuantificar y detectar el pestañeo, se aplica el algoritmo EAR *(Eye Aspect Ratio)*, propuesto en [64]. Este algoritmo propone una ecuación que permite medir si el ojo está abierto o no a partir de la relación entre la distancia horizontal y vertical de los puntos del ojo.

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|} \tag{8}$$

Donde p1, …, p6 son las coordenadas de las marcas faciales que devuelve dlib.

El numerador calcula la distancia en vertical entre las marcas de los ojos (definida por dos pares de puntos), mientras que el denominador calcula la distancia horizontal entre las marcas de los ojos. Como esta última está definida por un único par de puntos, se multiplica el denominador por dos para que sea equiparable. Como resultado de esta ecuación, se obtiene que el EAR es aproximadamente constante cuando el ojo está abierto, y decae rápidamente a cero cuando se cierra, obteniendo así una manera instantánea de determinar si una persona ha pestañeado.

En la siguiente figura, extraída del artículo de Soukupová y Čech, podemos ver una gráfica que describe el EAR durante un vídeo, donde se observa claramente el instante de tiempo donde la persona ha cerrado el ojo.

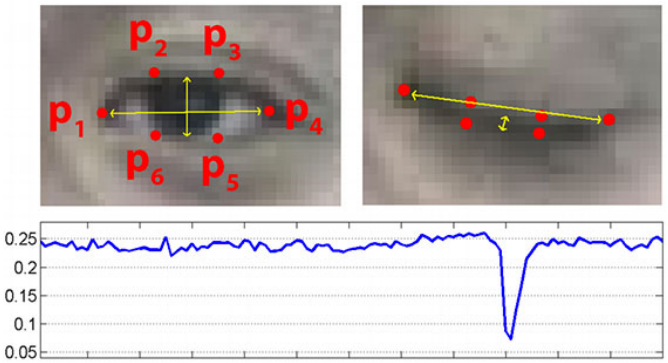


Figura 24: *Marcas faciales en el ojo abierto y cerrado. La gráfica muestra el EAR (ecuación 8) de varios fotogramas de un vídeo donde ha ocurrido un pestañeo. Fuente [64]*

Los autores de este artículo sugieren calcular la media entre el EAR de los dos ojos para obtener una mejor estimación del pestañeo (asumiendo que el usuario cierra los dos ojos a la vez).

La función *detecta_pestañeo.py*, calcula el EAR en cada fotograma y devuelve la variable booleana ELEM_ACTIVADO que será positiva si se detecta que el usuario ha pestañeado. Para decidir el valor *(true* o *false)* de esta variable, se ha establecido un umbral, por debajo del cual se considera que el ojo está cerrado. Así, si durante más un determinado número de fotogramas el EAR es inferior al umbral, cuando el ojo se vuelve a abrir (vuelve a superar el valor umbral), la variable PESTAÑEO toma el valor *true*.

Para definir los umbrales se han realizado diferentes pruebas con la Raspberry Pi para estimar los valores normales del EAR, teniendo en cuenta la distancia aproximada que habría entre la cámara y el usuario. Así, se ha decidido establecer el umbral del EAR en 0,31. En cuanto al número mínimo de fotogramas durante los cuales el EAR ha de estar por debajo del umbral, se han considerado, por una parte, los fotogramas por segundo que toma PiCamara (menos que en el caso de un ordenador normal), así como el enlentecimiento que conlleva la programación con hilos, que implica que no se esté ejecutando la función constantemente. Estos dos factores hacen necesario que este umbral sea bajo, para evitar un número considerable de falsos negativos. Se ha determinado un número mínimo de 3 fotogramas, considerando que el gesto ha de realizarse pausadamente para evitar falsos positivos.

Función detecta_BocaAbierta()

De forma análoga, se propone definir un parámetro (MAR, *Mouth Aspect Ratio*) que mida el ratio de apertura de la boca. Para ello, como se muestra en la figura 25, se seleccionan tres parejas de puntos para calcular la distancia vertical entre los labios, y una pareja para determinar la distancia horizontal. Posteriormente, se calcula el MAR mediante la aplicación de la ecuación 9.

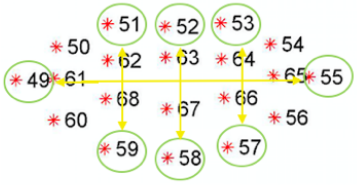


Figura 25: *Selección de los 8 puntos, de entre los 20 que definen el contorno de la boca, utilizados para detectar la apertura de la misma. Fuente: elaboración propia.*

$$MAR = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{3\|p_1 - p_5\|} \quad (9)$$

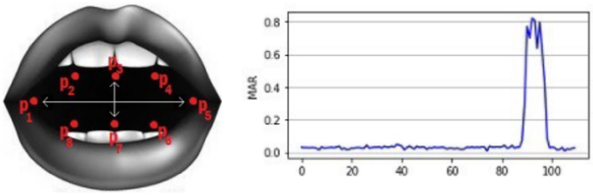


Figura 26: Las 8 marcas faciales que definirán el movimiento de la boca. La gráfica muestra el MAR (ecuación 9) de varios fotogramas de un vídeo donde se ha abierto la boca una vez. Fuente: [65]

La función *detecta_bocaAbierta.py* calcula el MAR y devuelve la variable booleana ELEM_ACTIVADO con valor *true*, cuando el parámetro supera cierto umbral (estimado en 0,8) durante más de una cantidad determinada fotogramas consecutivos. En este caso, se ha determinado un número mínimo de 5 fotogramas, superior al necesario con los ojos, para evitar falsos positivos que podrían darse si el usuario está hablando. También el umbral para el MAR se ha establecido en un valor lo suficientemente alto que define una apertura grande de la boca hecha de manera consciente.

4.2.2. Pulsador como elemento de activación.

Como último elemento de activación se propone un pulsador, que ha de reunir una serie de características que lo condicionen para ser utilizado por personas con diversidad funcional motora. Ha de tratarse, por lo tanto, de un pulsador de tamaño considerable y de un color llamativo, con una sensibilidad notable que facilite su uso a personas con poca fuerza física en los brazos.

La librería gpiozero, desarrollada por Ben Nuttall y Dave Jones, ofrece una interfaz sencilla e intuitiva para trabajar con los dispositivos conectables con la GPIO de la Raspberry Pi. Para trabajar con un botón, simplemente ha de crearse un objeto de la clase *Button* y llamar al método *boton.when_pressed()*, que esperará a que se pulse el botón para llamar a la función a la que se iguale dicho método, que en el caso de esta aplicación pueden ser o bien *rf_envia(),* o bien *cambiar_ventana().*

4.3. SALIDA. PROCESO DE ENVÍO DE LA SEÑAL

El proceso de envío de la señal inalámbrica comienza siempre cuando un elemento de activación es detectado. En ese momento, se ejecuta la función *rf_envía()*, que, en primer lugar, identifica la pantalla que se está mostrando al usuario en ese momento y, posteriormente, envía, vía RF, la señal requerida según la ventana mostrada, que llegará hasta el dispositivo que ha de efectuar la acción deseada por el usuario. A continuación, se detalla el proceso de envío y recepción de la señal.

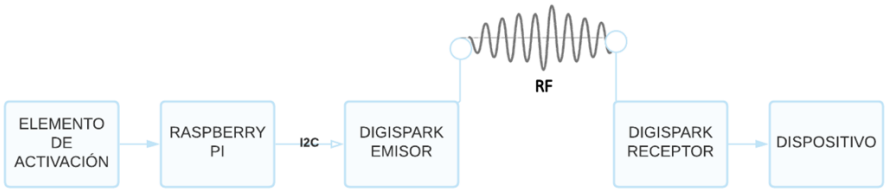


Figura 27: Esquema de las relaciones entre los diferentes elementos del sistema. Fuente: elaboración propia.

A cada pantalla se le ha asociado un índice, de manera que, a la primera pantalla le corresponde el 1, a la segunda, el 2 y así sucesivamente. La función *rf_envia()*, después de leer el nombre de la ventana, transfiere un byte con el índice de la ventana mostrada hasta la placa Digispark (que está conectada al puerto USB de la Raspberry) mediante el protocolo I^2C. Para ello, basta con llamar a la función *write_byte()* de la librería SMBus y pasarle como argumentos la dirección del bus y el byte que se quiere transmitir. En este caso, la Raspberry Pi es el maestro, y la placa Digispark, el esclavo.

Inmediatamente después de recibir un mensaje vía I2C, la placa lee el byte transferido y construye un mensaje que le enviará al módulo emisor de RF utilizando la librería MANCHESTER, que permite el envío de 16 bytes. De este modo, si el byte recibido ha sido un "1", entonces se indicará al módulo emisor de RF que envíe el mensaje "1111" hasta el módulo receptor. Análogamente, si se recibe un "2", se enviará el mensaje "2222".

Cada uno de los módulos receptores de RF, está a su vez conectado a una placa Digispark, encargada de procesar las señales detectadas por el módulo y activar la ejecución que el dispositivo ha de desempeñar. Cuando una señal es emitida por el módulo emisor, todos los receptores que estén al alcance podrán percibirla, pero solamente aquel al que va dirigido el mensaje ejecutará la acción correspondiente. Esto es posible debido a que el código cargado en las placas Digispark lee el mensaje recibido y solamente manda realizar la acción si este mensaje es el adecuado. Por ejemplo, la placa Digispark que esté conectada al dispositivo indicado en la ventana de índice 3 solo responderá ante el mensaje "3333". El funcionamiento descrito queda resumido en el siguiente esquema.

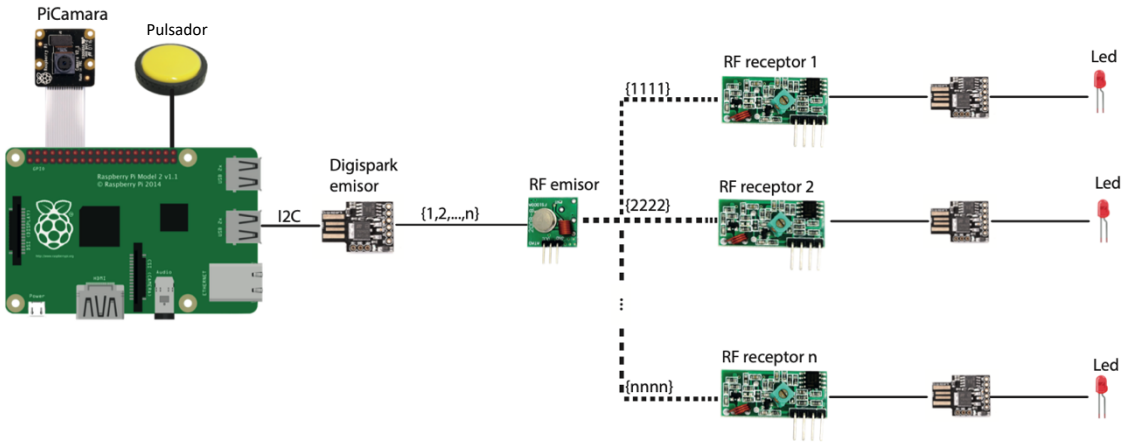


Figura 28: Esquema del funcionamiento del sistema de ayuda. Fuente: elaboración propia.

4. SISTEMA DE TRIAJE, PERSONALIZACIÓN DE LA APP Y CALIBRACIÓN

Se propone el seguimiento de una serie de pautas que contribuyen a conseguir una aplicación personalizada, que cubra las necesidades del usuario, y a lograr un funcionamiento óptimo según las condiciones particulares de cada uno.

En primer lugar, es necesario diseñar un sistema de triaje con el que evaluar las condiciones de cada persona y así decidir si el programa desarrollado en este TFG supone una ayuda adecuada para ella. Con este propósito, el usuario (o, en su lugar, las personas que están a su cuidado) deberá contestar una serie de preguntas que permitan determinar el grado de afectación del paciente y, con ello, el impacto que el sistema de ayuda diseñado tendría en su vida cotidiana. Para llevar a cabo esta evaluación, se requiere la colaboración de un profesional sanitario cualificado, como puede ser un terapeuta ocupacional.

Por otro lado, una vez se ha confirmado que el sistema resulta conveniente y beneficioso para el usuario, se deben consultar sus necesidades específicas para configurar las ventanas que aparecerán en la aplicación, así como los dispositivos a los que habrá que equipar con el hardware adecuado para la compatibilidad con el sistema. Del mismo modo, el usuario deberá elegir los elementos de activación más adecuados para controlar el sistema.

Finalmente, habrá que evaluar individualmente la instalación del dispositivo por lo que respecta al lugar donde se va a ubicar (por ejemplo, en la silla de ruedas), así como la colocación de la cámara y/o el pulsador, buscando en todo momento la comodidad del paciente y, por supuesto, el correcto funcionamiento del sistema.

Una vez el dispositivo está instalado, será conveniente realizar unas pruebas de calibración para comprobar que el sistema responde correctamente ante los gestos del usuario. Es especialmente relevante la calibración cuando los elementos de activación son el pestañeo o la apertura de la boca. En estos casos, la calibración inicial consistiría en medir los valores normales del EAR y del MAR, para poder establecer los umbrales adecuados a los movimientos particulares de cada persona.

CAPÍTULO 5. RESULTADOS Y CONCLUSIÓN

5.1. RESULTADOS

Se ha desarrollado un prototipo del sistema de ayuda ambiental en el que se han diseñado 3 ventanas con 3 acciones diferentes: subir la persiana, encender o apagar la luz y estirar de la cadena de wáter. Para simular el funcionamiento de la aplicación, se han utilizado tres leds que representan el comportamiento de los dispositivos del hogar mostrados en las ventanas, y que se desean controlar con el programa. El circuito se ha implementado en una protoboard, alimentada a partir de los pines de la Raspberry Pi a 5V. En la figura 29 se muestra un esquema de las conexiones del prototipo.

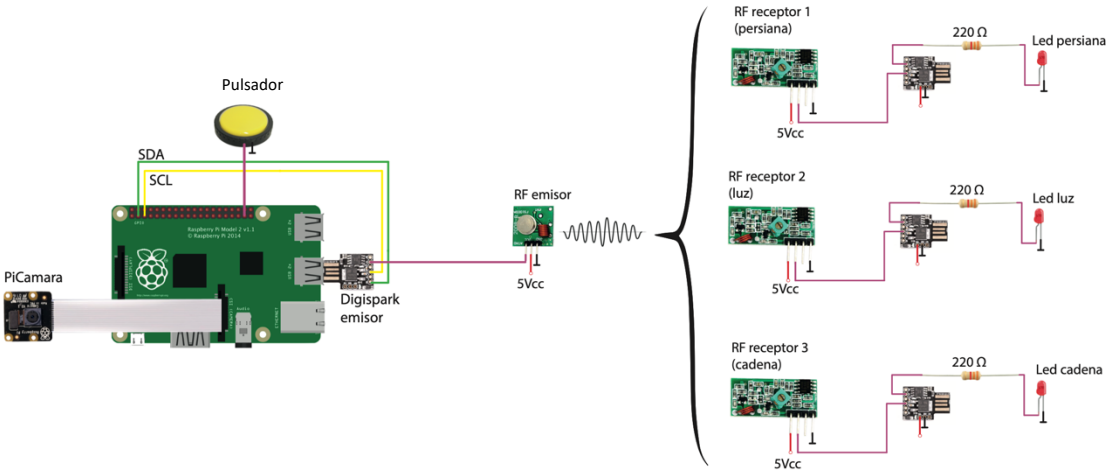


Figura 29: Esquema de conexiones del prototipo desarrollado. Fuente: elaboración propia.

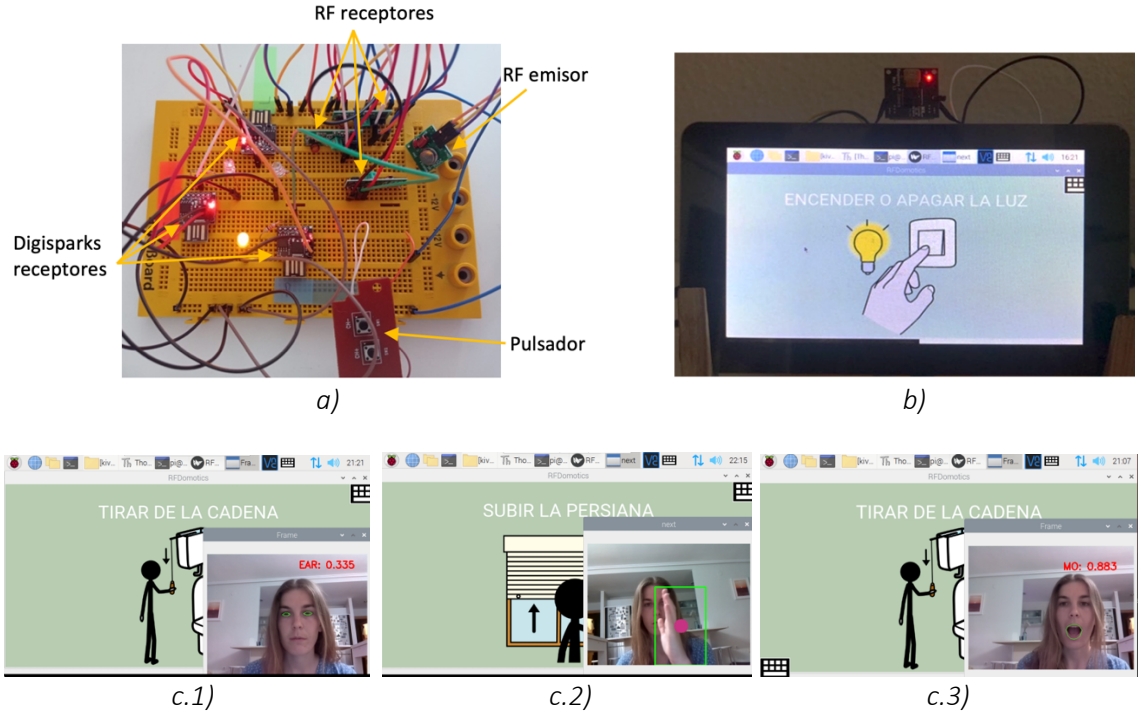
Las pantallas han sido creadas utilizando los pictogramas proporcionados por ARASAAC para garantizar la comprensión por parte de los usuarios que presenten algún déficit cognitivo. En la siguiente figura se muestran las pantallas diseñadas para el prototipo, en orden ascendente según índice asignado a cada ventana.

a) *b)* *c)*

Figura 30: Ventanas de la aplicación diseñadas para el prototipo. a) Ventana 1: persiana. b) Ventana 2: luz c) Ventana 3: cadena. Fuente: elaboración propia.

Antes de ejecutar la aplicación, se tienen que decidir los elementos de activación que controlarán su funcionamiento. Una vez determinados y modificado consecuentemente el código según las preferencias del usuario (véase el punto 4 del anexo), se activa el entorno virtual creado y se ejecuta la aplicación desde la terminal de la Raspberry Pi. Al iniciarse, aparece en primer lugar la ventana "subir la persiana", cuyo índice es 1. El programa entonces ejecutará los dos hilos implementados (ver figura 13), y permanecerá entonces a la espera de una orden, que le indicará que cambie de ventana o que envíe una señal por RF para subir la persiana. En el primer caso, la pantalla mostrará la ventana con índice 2 ("encender o apagar la luz"). Si, por el contrario, el usuario solicita subir la persiana, el programa mandará un byte "1" vía I2C a la placa Digispark emisora, la cual entonces hará mandar, el mensaje "1111" al módulo emisor de RF. Cuando la placa Digispark asociada a la persiana reciba el mensaje, mandará encender el led (si este está apagado), o apagarlo (si está encendido), lo que indicará que la transmisión se ha efectuado adecuadamente. Como se ha explicado a lo largo del trabajo, este mecanismo se repite de forma análoga para la pantalla 2 y para la pantalla 3 ("estirar de la cadena"), encendiéndose o apagándose el led correspondiente.

A continuación, se muestra una imagen del prototipo desarrollado, donde quedan señalados los diferentes componentes que han sido utilizados. Además, se presentan capturas de pantalla de la aplicación en marcha, en las que se puede visualizar la imagen que capta la cámara para comprobar el funcionamiento del algoritmo (en una implementación real no sería necesario mostrar esta imagen).



a) *b)*

c.1) *c.2)* *c.3)*

Figura 31: *Imágenes del prototipo diseñado. a) Protoboard con los diferentes dispositivos externos conectados siguiendo el esquema de la figura 29 b) pantalla de la Raspberry Pi con la aplicación en marcha y la cámara colocada para la captación de las imágenes c) capturas de pantalla de la aplicación en marcha y el output de las funciones de detección de movimientos mediante visión artificial: (1) detecta_pestañeo() (2) dir_derecha() (3) detecta_BocaAbierta(). Fuente: elaboración propia.*

A la hora de comercializar el producto, se debe buscar minimizar el espacio utilizado por el hardware para favorecer la portabilidad del sistema. La pantalla y la cámara, conectadas directamente a la Raspberry Pi, habrían de colocarse en un lugar cómodo para el usuario. Como se ha

comentado anteriormente, los dispositivos que se deseen controlar mediante el programa deberán estar correctamente equipados. Si bien es cierto que en este prototipo se han utilizado los módulos de RF, se debe contemplar la alternativa de utilizar otros mecanismos para la transmisión inalámbrica (como el bluetooth o el wifi), puesto que, actualmente, los dispositivos electrónicos de los hogares están equipados de dicha tecnología, lo que facilitaría la instalación, compatibilidad y versatilidad de sistema de ayuda.

Es importante remarcar que, a pesar de que se han implementado únicamente 3 pantallas, debido a las limitaciones del material disponible y al carácter académico del TFG, el sistema podría ampliarse hasta cubrir todas las necesidades del usuario y personalizarse según sus prioridades. Así pues, el prototipo ha demostrado cumplir con las funciones y los requerimientos básicos planteados. Sin embargo, presenta ciertas limitaciones que se comentan a continuación.

La mayor limitación la encontramos a la hora de captar los elementos de activación mediante la cámara. Existen varios factores que dificultan la correcta interpretación de los movimientos. En primer lugar, el hecho de tener que utilizar el clasificador *Haar Cascade* resulta un inconveniente, puesto que, en ocasiones, detecta rostros donde no los hay, lo cual puede derivarse en una detección errónea de un elemento de activación, haciendo que el programa realice una acción que el usuario no desea.

Por otro lado, en el caso de la detección del movimiento el problema se agrava, ya que cualquier cambio en la imagen con respecto al fotograma anterior que sea lo suficientemente grande, se interpreta como un movimiento. Esto supone que, si cambia la iluminación (por ejemplo, al encender la luz de la habitación), o si aparece en el encuadre un movimiento involuntario, como puede ser que una persona pase por detrás, o que el usuario se mueva al gesticular, puede ocasionar un número significativo de falsos positivos. Además, el sistema está pensado para ser portátil, de forma que la persona que lo vaya a utilizar pueda llevarlo acoplado en la silla de ruedas, lo que dificulta la tarea del procesamiento de la imagen mediante la técnica de sustracción de fondo empleada por el algoritmo, ya que esta requiere una cámara estática para un funcionamiento óptimo.

Una tercera limitación está relacionada con la utilización de los módulos de RF. Alimentados a 5V y con la antena propia del módulo, su alcance es muy corto, lo que hace necesario posicionar los receptores una distancia muy corta del emisor (de unos cuantos centímetros). Esto hace inviable la utilización de dichos módulos a la hora de una implementación real.

Por lo tanto, a la hora de proponer un dispositivo viable, el prototipo debería incluir ciertas mejoras. En primer lugar, es importante optimizar el rendimiento de los algoritmos de detección de gestos faciales. Para ello, además de cerciorarse de que la cámara está colocada de tal forma que la cara del usuario queda perfectamente encuadrada, se recomienda utilizar un modelo de Raspberry Pi más moderno (preferiblemente la Raspberry Pi 4). Esta ofrece un procesador mucho más potente, que ejecutaría los hilos más rápido e incluso permitiría implementar la opción de utilizar el detector implementado en la librería dlib basado en HOG y VSM, que es más preciso. Estas recomendaciones mejorarían también el rendimiento del algoritmo de detección de movimiento, aunque para que este funcionase de manera adecuada, lo más conveniente sería añadir una cámara, que estuviese enfocando a una zona donde la imagen fuera siempre fija, excepto cuando el usuario hiciera pasar un objeto en movimiento por dicha zona. Otra alternativa interesante que se ha contemplado para la detección del movimiento es sustituir la técnica de sustracción de fondo empleada por la detección de objetos según su color. Así, se evitarían todos inconvenientes mencionados en relación con el movimiento de la cámara. La aplicación podría calibrarse para detectar un color determinado, según el tono de piel del usuario o algún elemento distintivo diseñado específicamente para tal fin.

Finalmente, en cuanto al alcance de los módulos de radiofrecuencia, puede mejorarse destacablemente incrementando la alimentación a 12V y añadiendo una antena de cobre lo suficientemente larga (alrededor de 16,5 cm) a cada uno de los módulos (tano el emisor como los receptores). De esta manera, se conseguiría un rango de cobertura de hasta 300 metros en exteriores [66], con lo que las necesidades del sistema quedarían satisfechas.

5.2. CONCLUSIONES

El desarrollo de este trabajo destaca la relevancia de buscar alternativas de bajo coste para la creación de tecnologías de apoyo que supongan una mejora para la calidad de vida de las personas que las necesiten, así como de aquellas que están a su cuidado. Concretamente, en el ámbito de la domótica en el que se enmarca este TFG, es fundamental que estas tecnologías permitan aumentar la autonomía y la independencia de los usuarios.

En el ámbito de la diversidad funcional motora, resulta una tarea compleja encontrar el equilibrio entre adecuar la tecnología a las necesidades específicas de cada usuario y ofrecer una solución para el mayor número de personas posible. Esto es debido a que se trata de un concepto muy amplio, que abarca condiciones muy diversas que afectan en diferentes grados y de distinta manera a cada persona. Por eso, en este TFG se plantea una solución domótica que busca un diseño para todos, ofreciendo alternativas que pueden adecuarse a la situación concreta de cada usuario, dentro de un marco común de condiciones. Así, se ha desarrollado un prototipo que consigue captar ciertos gestos a través de una cámara y transformarlos en órdenes que controlan el funcionamiento de los dispositivos electrónicos del hogar del usuario, en este caso simulados mediante leds.

La placa Raspberry Pi ha demostrado ser una opción viable para el diseño de dispositivos y sistemas de ayuda, gracias a su potencial como miniordenador, su versatilidad y su compatibilidad con un gran número de dispositivos externos, todo ello con un precio económico. Las técnicas de visión artificial para el control de las aplicaciones muestran todavía algunas limitaciones al implementarse en la Raspberry Pi, que suponen desventajas importantes a la hora de implementar una solución óptima. No obstante, los avances tecnológicos en este ámbito son esperanzadores y, por el momento, existen alternativas que hacen posible el desarrollo de tecnologías de apoyo asequibles y de calidad.

Para concluir, cabe mencionar que, al producto que se ha desarrollado, podrían añadírsele ciertas mejoras, como las mencionadas en el capítulo anterior, que permitieran precisar la detección de los elementos de activación para lograr un mejor funcionamiento del sistema. Además, existen diversas alternativas para abordar el problema planteado en un primer momento, que conseguirían alcanzar los objetivos propuestos en este trabajo, como podría ser la interacción con la pantalla táctil o el uso de sensores de movimiento. Se pueden encontrar numerosos módulos muy económicos y compatibles con la Raspberry Pi basados, por ejemplo, en la detección de ultrasonidos, de radiación infrarroja, o de intensidad lumínica. En este TFG, se ha planteado una solución viable que permitía a la alumna profundizar en temas de interés y de relevancia actual, como son la visión artificial y el aprendizaje automático, cumpliendo con los objetivos planteados al inicio de este proyecto.

BIBLIOGRAFÍA

[1] Instituto Nacional de Estadística, «Panorámica de la discapacidad en España. Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia. 2008,» *Boletín Informativo del Instituto Nacional de Estadística,* pp. 1-12, 2009.

[2] Organización Mundial de la Salud, «Tecnologías de asistencia,» 18 Mayo 2018. [En línea]. Available: https://www.who.int/es/news-room/fact-sheets/detail/assistive-technology. [Último acceso: junio 2021].

[3] Mouse4all SL, «Mouse4all box,» [En línea]. Available: https://mouse4all.com/es/comprar/mouse4all-box/. [Último acceso: junio 2021].

[4] Mouse4all, «Domótica Accesible para ser independiente en tu propio hogar,» 21 Mayo 2019. [En línea]. Available: https://mouse4all.com/es/articles/domotica-accesible-para-ser-independiente-en-tu-hogar/. [Último acceso: junio 2021].

[5] BJ Adaptaciones, «TrackerPro 2,» [En línea]. Available: https://bjadaptaciones.com/con-la-cabeza-boca-o-labios/225-tracker-pro-2.html. [Último acceso: Junio 2021].

[6] Cátedra de Accesibilidad a las TIC de la Universidad de Lleida, «Universitat de Lleida: HeadMouse,» marzo 2016. [En línea]. Available: http://robotica.udl.cat/headmouse.htm. [Último acceso: julio 2021].

[7] S. Figueroa, «EVA Facial Mouse, controla tu móvil solo con movimientos de cabeza,» Xataka Android, 31 Marzo 2016. [En línea]. Available: https://www.xatakandroid.com/productividad-herramientas/eva-facial-mouse-controla-tu-movil-con-movimientos-de-cabeza. [Último acceso: Junio 2021].

[8] R. Gupta, R. Kori, S. Hambir, A. Upadhayay y S. Sahu, «Eye Controlled Wheelchair Using Raspberry Pi,» *Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST) 2020,* 2020.

[9] PantechSolutions, «Raspberry Pi based Wheel Chair Control with Eyeball Movement,» 2020. [En línea]. Available: https://www.pantechsolutions.net/raspberry-pi-based-wheel-chair-control-with-eyeball-movement. [Último acceso: junio 2021].

[10] S. Anwer, A. Waris, H. Sultan, S. I. Butt, M. H. Zafar, M. Sarwar, I. K. Niazi, M. Shafique y A. N. Pujari, «Eye and Voice-Controlled Human Machine Interface System for Wheelchairs Using Image Gradient Approach,» 26 Septiembre 2020.

[11] Y. Rabhi, M. Mrabet y F. Fnaiech, «A facial expression controlled wheelchair for people with disabilities,» *Comput Methods Programs Biomed,* Octubre 2018.

[12] L.-M. Yuyuqui y M.-G. Ignacio, «Desarrollo y Modelación de Solución Domótica para la asistencia a Personas con Diversidad Funcional Motora utilizando Hardware Libre,» 30 Septiembre 2015.

[13] Z. Zou, Y. Wang, L. Wang, X. Wu, C. Xu y M. Zhou, «Design of smart home controller based on raspberry PI,» *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC),* pp. 1548-1551, 2020.

[14] S. Ostensjø, E. B. Carlberg y N. K. Vøllestad, «The use and impact of assistive devices and other environmental modifications on everyday activities and care in young children with cerebral palsy,» *Disabil Rehabil,* 22 Julio 2005.

[15] S. H. R. P. Henderson S, «Assistive devices for children with functional impairments: impact on child and caregiver function,» *Dev Med Child Neurol,* Febrero 2008.

[16] S. D. B. S. Huang IC, «Assistive devices and cerebral palsy: factors influencing the use of assistive devices at home by children with cerebral palsy,» *Child Care Health Dev,* Enero 2009.

[17] Servicio de Información sobre Discapacidad, «El diseño para todos implica una tecnología modificada, de bajo coste y con amplia difusión entre los usuarios,» 12 Noviembre 2005. [En línea]. [Último acceso: Junio 2021].

[18] ONU: Asamblea General, «Declaración Universal de Derechos Humanos,» 10 Diciembre 1948. [En línea]. Available: https://www.refworld.org.es/docid/47a080e32.html. [Último acceso: Junio 2021].

[19] España., «Real Decreto Legislativo 1/2013, de 29 de noviembre, por el que se aprueba el Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social,» 4 Diciembre 2013. [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2013-12632&p=20171109&tn=1. [Último acceso: Junio 2021].

[20] J. I. Pérez y M. Garaigordobil, «Discapacidad motriz: autoconcepto, autoestima y síntomas psicopatológicos,» *Studies in Psycology,* vol. 28, pp. 343-357, 2007. Published online: 23 Jan 2014.

[21] Atremo, «¿Qué es la discapacidad motora?,» [En línea]. Available: https://www.atremo.org/que-es-la-discapacidad-motora/. [Último acceso: Julio 2021].

[22] A. Jiménez Lara y A. Huete García, «La Discapacidad en Cifras,» 2002. [En línea]. Available: https://sid.usal.es/idocs/F8/8.1-6367/8.1-6367.pdf. [Último acceso: Junio 2021].

[23] ARASAAC, «¿Qué son los SAAC?,» [En línea]. Available: https://arasaac.org/aac/es. [Último acceso: Julio 2021].

[24] M. G. R. Pi, «Aprendiendo Arduino,» 4 Marzo 2020. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2020/03/04/manejar-gpio-raspberry-pi/. [Último acceso: Junio 2021].

[25] D. LLC, «digistump products,» [En línea]. Available: http://digistump.com/products/1. [Último acceso: junio 2021].

[26] M. Aaserud, Artist, *Digispark Attiny85 Pinout.* [Art]. Deviant Art, 2018.

[27] Cognex Corporation, «Introducción a la visión artificial,» [En línea]. Available: https://bcnvision.es/uploads/videotutoriales/uploads/guias%20por%20sectores/introduccion%20a%20la%20vision%20artificial_compressed.pdf. [Último acceso: Junio 2021].

[28] A. Pajankar, Raspberry Pi Computer Vision Programming, Birmingham - Mumbai: Packt Publishing, 2015.

[29] OpenCV, «About,» [En línea]. Available: https://opencv.org/about/. [Último acceso: Junio 2021].

[30] V. Kazemi y J. Sullivan, «One millisecond face alignment with an ensemble of regression trees,» *2014 IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1867-1874, 2014.

[31] Dlib Library, «Real-Time Face Pose Estimation,» 28 Agosto 2014. [En línea]. Available: http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html. [Último acceso: Junio 2021].

[32] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou y M. Pantic, «A semi-automatic methodology for facial landmark annotation,» *Proceedings of IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR-W), 5th Workshop on Analysis and Modeling of Faces and Gestures (AMFG 2013),* Junio 2013.

[33] V. Le, Z. L. Jonathan Brandt, L. Boudev y T. S. Huang, «Interactive Facial Feature Localization,» *Springer, Berlin, Heidelberg.,* pp. 679-692, October 2012.

[34] A. Rosebrock, «Facial landmarks with dlib, OpenCV, and Python,» 3 Abril 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/. [Último acceso: Junio 2021].

[35] Kivy, [En línea]. Available: https://kivy.org/#home. [Último acceso: Junio 2021].

[36] Python Documentation, «Entornos Virtuales y Paquetes,» [En línea]. Available: https://docs.python.org/es/3/tutorial/venv.html. [Último acceso: Junio 2021].

[37] Federación de Enseñanza de CCOO de Andalucía, «La conectividad Inalámbrica: un enfoque para el alumno,» *Revista digital para profesionales de la enseñanza,* nº 6, Enero 2010.

[38] R. Olivarex, «WiFi vs Bluetooth: diferencias, ventajas e inconvenientes,» 3 Abril 2019. [En línea]. Available: https://www.actualizatucasa.com/wifi-vs-bluetooth/. [Último acceso: Junio 2021].

[39] Digistump Forums, «Use cheap little 433mhz OOK transmitters and receivers to link your sparks up,» 23 Marzo 2013. [En línea]. Available: https://digistump.com/board/index.php?topic=879.0. [Último acceso: Junio 2021].

[40] HETPRO Tutoriales, «I2C – Puerto, Introducción, trama y protocolo,» 2018. [En línea]. Available: https://hetpro-store.com/TUTORIALES/i2c/. [Último acceso: Junio 2021].

[41] DroneBot Workshop, «I2C Between Arduino & Raspberry Pi,» 2020. [En línea]. Available: https://dronebotworkshop.com/i2c-arduino-raspberry-pi/. [Último acceso: Junio 2021].

[42] mikroElektronika, «SMBus Library,» [En línea]. Available: http://download.mikroe.com/documents/compilers/mikroc/8051/help/smbus_library.htm. [Último acceso: Junio 2021].

[43] CEDOM - Asocioación Española de Domótica e Inmótica, «Qué es Domótica,» [En línea]. Available: http://www.cedom.es/sobre-domotica/que-es-domotica. [Último acceso: Julio 2021].

[44] discapnet, «Domótica y discapacidad. Guía del usuario,» 2007. [En línea]. Available: https://www.discapnet.es/sites/default/files/minisites/guia_domotica/domdisc.html. [Último acceso: Junio 2021].

[45] E. Rico Schmidt, «threading — Gestionar operaciones concurrentes dentro de un proceso,» [En línea]. Available: https://rico-schmidt.name/pymotw-3/threading/. [Último acceso: Junio 2021].

[46] J. Anderson, «Real Python: An Intro to Threading in Python,» 2019. [En línea]. Available: https://realpython.com/intro-to-python-threading/. [Último acceso: Junio 2021].

[47] Wikipedia, «Gaussian blur,» 9 Junio 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Gaussian_blur. [Último acceso: Junio 2021].

[48] OpenCV, «How to Use Background Subtraction Methods,» 2021. [En línea]. Available: https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html. [Último acceso: Junio 2021].

[49] K. P. y B. R., «An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection,» *Springer, Boston, MA,* 2002.

[50] OpenCV, «Background Subtraction,» 2021. [En línea]. Available: https://docs.opencv.org/master/d8/d38/tutorial_bgsegm_bg_subtraction.html. [Último acceso: Julio 2021].

[51] Z. Zivkovic, «Improved adaptive Gaussian mixture model for background subtraction,» de *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, Cambridge, UK, 2004.

[52] Z. Zivkovica y F. d. Heijdenb, «Efficient adaptive density estimation per image pixel for the task of background subtraction,» *Elsevier B.V.,* vol. 27, nº 7, pp. 773-780 , 2006.

[53] A. B. Godbehere, A. Matsukawa y K. Goldberg, «Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation,» de *2012 American Control Conference (ACC)*, Montreal, QC, Canada, 2012.

[54] «Imágenes biomédicas (13062),» Dpto. de Física Aplicada Dpto. de Comunicaciones, UPV, 2021.

[55] The MathWorks, Inc., «Elementos estructurante,» [En línea]. Available: https://es.mathworks.com/help/images/structuring-elements.html. [Último acceso: Junio 2021].

[56] MathWorks Inc, «Tipos de Operaciones Morfológicas,» [En línea]. Available: https://es.mathworks.com/help/images/morphological-dilation-and-erosion.html. [Último acceso: Junio 2021].

[57] A. Rosebrock, «Raspberry Pi: Facial landmarks + drowsiness detection with OpenCV and dlib,» 23 Octubre 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/10/23/raspberry-pi-facial-landmarks-drowsiness-detection-with-opencv-and-dlib/. [Último acceso: Junio 2021].

[58] The MathWorks, Inc., «Hiperplanos óptimos como límites de decisión,» [En línea]. Available: https://es.mathworks.com/discovery/support-vector-machine.html. [Último acceso: Junio 2021].

[59] N. Dalal y B. Triggs, «Histograms of Oriented Gradients for Human Detection,» de *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). Ieee.*, 2005.

[60] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features,» de *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001 (Vol. 1, pp. I-I). Ieee.*, 2001.

[61] OpenCV, «Cascade Classifier,» 2021. [En línea]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. [Último acceso: Junio 2021].

[62] A. Rosebrock, «Pyimagesearch,» [En línea]. Available: https://www.pyimagesearch.com. [Último acceso: Junio 2021].

[63] A. Rosebrock, «Eye blink detection with OpenCV, Python, and dlib,» 24 Abril 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/. [Último acceso: Junio 2021].

[64] T. Soukupová y Jan Cech, «Real-Time Eye Blink Detection using Facial Landmarks,» de *21st computer vision winter workshop*, Rimske Toplice, Slovenia, 2016.

[65] A. L. Chandra, «Mouse Cursor Control Using Facial Movements — An HCI Application,» 7 Octubre 2018. [En línea]. Available: https://towardsdatascience.com/mouse-control-facial-movements-hci-app-c16b0494a971. [Último acceso: Junio 2021].

[66] L. Llamas, «Comunicación inalámbrica en Arduino con módulos RF 433MHz,» 6 Diciembre 2016. [En línea]. Available: https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/. [Último acceso: Junio 2021].

PRESUPUESTO

PRESUPUESTO

1. INTRODUCCIÓN

A continuación, se procede a exponer la valoración económica del proyecto desarrollado en el presente Trabajo de Fin de Grado, con el fin de estimar el coste final producto. En esta introducción expondremos algunas consideraciones generales que se han tenido en cuenta para la realización del presupuesto.

Se ha dividido este documento en dos apartados. Primeramente, se presentará el precio de la elaboración del prototipo, donde se mostrará el coste de la mano de obra involucrada en su fabricación, detallando las tareas realizadas en un cuadro de mediciones, así como los gastos del material utilizado. En segundo lugar, se realizará una valoración del presupuesto de la comercialización del producto, teniendo en cuenta todos los gastos involucrados en la salida al mercado del dispositivo. Para ello, se ha supuesto que se fabrica una serie de 100 dispositivos, teniendo en cuenta que cada uno controlará una media de 5 aparatos en el hogar del usuario.

Por otro lado, el coste de los materiales que aparecen en las tablas siguientes ha sido extraído de las páginas web de sus fabricantes sin el IVA añadido. Además, para el cálculo de la amortización de los ordenadores utilizados para el desarrollo del software, se ha considerado tienen un tiempo de vida útil de 8 años. A continuación, se detallarán, en cada uno de los apartados siguientes, las consideraciones específicas oportunas.

2. PRESUPUESTO DEL DESARROLLO DEL PROTOTIPO

En la primera tabla se describen las tareas realizadas durante el desarrollo del prototipo del sistema de ayuda ambiental para personas con diversidad funcional motora. Las tareas se han dividido por capítulos en orden cronológico de elaboración.

En la segunda tabla, se muestra la cantidad remunerada a cada trabajador que ha intervenido en la fabricación del prototipo. Se ha considerado a la alumna como una ingeniera biomédica junior, con un sueldo de 13€ la hora. Los profesores que han contribuido en la realización del proyecto se consideran ingenieros expertos, cuyo sueldo asciende a 31€ la hora, y el del terapeuta ocupacional que ha colaborado como asesor tendría un sueldo de 25 euros por hora.

Finalmente, en la última tabla de este apartado se desglosan los materiales utilizados durante el desarrollo del prototipo. Los gastos de las dos últimas tablas se prorratearán sobre la serie de 100 unidades que se quiere comercializar.

Tabla 2: Cuadro de mediciones de las actividades llevadas a cabo para el desarrollo del prototipo.

CUADRO DE MEDICIONES PROTOTIPO

01 DEFINICIÓN DEL PROYECTO

Código	Ud.	Descripción	Cantidad
01.01	h	Reunión inicial con el tutor del TFG	2
01.02	h	Reunión con terapeuta ocupacional para la consulta de las necesidades de las personas con diversidad funcional motora	2
01.03	h	Estudio de los antecedentes	5
01.04	h	Reuniones para planificación y control de las actividades a realizar	12
01.04	h	Selección y compra del material	1
		TOTAL:	22

02 INVESTIGACIÓN DEL ESTADO DEL ARTE

Código	Ud.	Descripción	Cantidad
02.01	h	Investigación del estado del arte	25
02.02	h	Proceso de selección de los materiales y programas que se van a emplear	20
02.03	h	Aprendizaje del manejo de materiales y programas	30
		TOTAL:	75

03 DESARROLLO DEL SISTEMA DE AYUDA

Código	Ud.	Descripción	Cantidad
03.01	h	Puesta en marcha del SO de la Raspberry	8
03.02	h	Instalación de librerías en la Raspberry	30
03.03	h	Desarrollo del código	40
03.04	h	Desarrollo de los algoritmos	60
03.05	h	Diseño de la aplicación	35
03.06	h	Montaje del circuito	4
03.07	h	Realización de pruebas y corrección de errores	80
		TOTAL:	257

04 REDACCIÓN Y DEFENSA DEL TFG

Código	Ud.	Descripción	Cantidad
04.01	h	Redacción de los documentos	60
04.02	h	Preparación de la exposición	2
		TOTAL:	62

TOTAL HORAS:	416

Tabla 3: Cuadro de precios de la mano de obra que interviene durante el desarrollo del prototipo.

Cuadro de precios MANO DE OBRA

MANO DE OBRA PARA EL DISEÑO DEL PROTOTIPO					
Código	**Descripción del recurso**	**Precio (€/h)**	**Cantidad**	**Amortización**	**Total**
MO.IBJ	Ingeniero Biomédico Junior	13	416	-	5408
MO.IT	Ingeniero tutor	31	30	-	620
MO.IC	Ingeniero cotutor	31	5	-	155
MO.TO	Terapeuta Ocupacional	25	2	-	50
				TOTAL:	6233

Tabla 4: Cuadro de precios de los materiales que se han utilizado durante el desarrollo del prototipo.

Cuadro de precios MATERIALES

MATERIALES PARA EL DISEÑO DEL PROTOTIPO					
Código	**Descripción del recurso**	**Precio (€)**	**Cantidad**	**Amortización**	**Total**
HAR.01	Placa Raspberry PI 2B	24,31	1	-	24,31
HAR.02	Raspberry Pi 7" Touch Screen LCD	41,68	1	-	41,68
HAR.03	Raspberry Pi PiNoir camera modul	18,75	1	-	18,75
HAR.04	Placa Digispark Attiny 85 USB	1,68	1	-	1,68
HAR.05	Emisor RF	0,32	1	-	0,32
HAR.06	Receptor RF	0,32	3	-	0,96
HAR.07	Tarjeta microSD 16 GB	2,51	1	-	2,51
HAR.08	Portátil MacBook Pro 2018	1100	1	0,07	80,21
HAR.09	Microsoft Office 365	69	1	0,58	40,25
HAR.10	Leds	0,13	3	-	0,40
HAR.11	Protoboard	5	1	-	5,00
HAR.12	Pulsador prueba	0,192	1	-	0,19
HAR.13	Cables	2,50	1	-	2,50
				TOTAL:	218,76

Con todo ello, podemos calcular el precio total de la elaboración del prototipo, que se contabilizará posteriormente como gasto para la comercialización del producto que se repartirá entre los 100 dispositivos que se van a fabricar.

Tabla 5: resumen del coste total de la elaboración del prototipo.

COSTE TOTAL DE LA ELABORACIÓN DEL PROTOTIPO	
Mano de obra	6233
Materiales	218,76
Total	6451,76

3. COMERCIALIZACIÓN DEL PRODUCTO

Para la comercialización del producto se requiere, en primer lugar, mano de obra para poner a punto el sistema. Para ello, se contará con un informático que optimice los algoritmos, así como un ingeniero y un terapeuta ocupacional que trabajen como asesores para la validación. Además, se requiere un técnico superior electrónico para el prototipado y la comprobación de las placas de circuito impreso. Por otra parte, serán necesarios un par operarios para proceso de fabricación que se encarguen del ensamblaje, soldaduras de algunos elementos, realizar un control de calidad, embalar y empaquetar los dispositivos. Se estima que cada operario tarda alrededor de 3 horas en poner a punto cada producto, por lo que las horas trabajadas son 300h.

Por otra parte, es necesario añadir ciertos elementos de hardware que no se requieren durante el desarrollo del prototipo. Así, se han incorporado las fuentes de alimentación de los elementos receptores (que se pueden alimentar por pilas), así como una batería portátil y una carcasa de protección para la Raspberry Pi. También los elementos de soporte, que ayudarán a la colocación del producto de forma adecuada para el uso particular de cada usuario, cuyo precio se aproximan a 20 € de media, suponiendo que se pueden fabricar con una impresora 3D. Igualmente, se han tenido en cuenta los complementos para el embalaje (cajas, material protector, etc.).

Cabe destacar que los accionamientos de potencia que se encontrarán en los dispositivos receptores (el motor para subir la persiana o para encender la luz, por ejemplo) no están incluidos en este presupuesto, aunque cada dispositivo que se desee controlar mediante el sistema habrá de adaptarse de manera adecuada. Por otra parte, es importante mencionar, como se observa en la tabla 7, que todo el software utilizado tiene una licencia que permite su uso comercial gratuito, por lo que no supone ningún coste.

Además, se han incluido los gastos en publicidad. Debido a lo específico de la patente, se ha escogido una estrategia de presentación al mercado en puntos clave, con públicos afines (asociaciones, residencias y hospitales) y muy bajo coste, que solo incluiría el desplazamiento al lugar y el tiempo que invierte el ponente en la preparación y la exposición de la charla, por lo que un presupuesto total de 400 euros sería suficiente.

Para concluir, se han añadido también los gastos de propiedad industrial, que incluyen la creación de la marca y el registro de la patente. Este gasto, al igual que el del desarrollo del prototipo, se ha repartido entre cada uno de los 100 dispositivos fabricados añadiéndoles un factor de amortización de 1/100.

Tabla 6: Cuadro de precios de la mano de obra que interviene durante diseño del producto comercial.

Cuadro de precios MANO DE OBRA

MANO DE OBRA PARA LA MEJORA DEL PRODUCTO

Código	Descripción del recurso	Precio	Cantidad	Amortización	Total
MO.01	Informático especialista	20	240	-	4800
MO.02	Terapeuta Ocupacional asesor	25	15	-	375
MO.03	Ingeniero Senior asesor	31	15	-	465
MO.04	Técnico superior electrónico	15	132	-	1980
MO.05	Operarios para el proceso de fabricación	13	300	-	3900
				TOTAL:	11520

Tabla 7: Cuadro de precios del material necesario para la fabricación de un producto comercializable.

Cuadro de precios MATERIALES

HARDWARE

Código	Descripción del recurso	Precio	Cantidad	Amortización	Total
HAR.01	Placa Raspberry PI 2B	24,31	100	-	2431
HAR.02	Raspberry Pi 7" Touch Screen LCD	41,68	100	-	4168
HAR.03	Raspberry Pi PiNoir camera modul	18,75	100	-	1875
HAR.04	Placa Digispark Attiny 85 USB	1,68	500	-	840
HAR.05	Emisor RF	0,32	100	-	32
HAR.06	Receptor RF	0,32	500	-	160
HAR.07	Tarjeta microSD 16 GB	2,51	100	-	251
HAR.08	Caja Protectora de la Raspberry	7,85	100	-	785
HAR.09	Batería portátil para Raspberry	15	100		1500
HAR.10	Portapilas	3	500		1500
HAR.11	Cables y conectores	0,5	500	-	250
HAR.12	Ordenador	800	1	0,021	16,67
				TOTAL:	13808,7

SOFTWARE

Código	Descripción del recurso	Precio	Cantidad	Amortización	Total
SOFT.01	Arduino IDE	0	1	-	0
SOFT.02	Python 3.7	0	1	-	0
SOFT.03	Dlib Library	0	1	-	0
SOFT.04	OpenCV	0	1	-	0
SOFT.05	Licencia Kivy	0	1	-	0
SOFT.06	Raspberry Pi OS	0	1	-	0
				TOTAL:	0,00

EXTRAS DE COMERCIALIZACIÓN

Código	Descripción del recurso	Precio	Cantidad	Amortización	Total
EX.01	Cajas embalaje	0,61	100	-	61
EX.02	Envolventes, material protector	65	1	-	65
EX.03	Elementos de soporte y posicionamiento	20	100	-	2000
EX.04	Publicidad	400	1	-	400
				TOTAL:	2526,00

PROTOTIPO Y PATENTE

Código	Descripción del recurso	Precio (€)	Amortización	Coste por dispositivo
EX.PROT	Gastos producción prototipo	6451,76	0,01	64,52
EX.PAT	Derechos de propiedad industrial	2000	0,01	20
	TOTAL:	8451,76		

A partir de los gastos detallados en las anteriores tablas, podemos estimar el precio de fabricación del producto. Para obtener el precio de venta, se ha añadido un 13% al presupuesto de ejecución material como gastos generales de la empresa, así como un 10% más de beneficio industrial.

Tabla 8: : resumen del coste total del producto comercializable.

COSTE TOTAL DE LA COMERCIALIZACIÓN DEL PRODUCTO	
Mano de obra	11520,00
Materiales	13808,7
Extras de comercialización	2526
Prototipo y patente	8451,76
Coste total	36306,43
Coste por dispositivo	363,06
Gastos generales empresa (13%)	47,2
Beneficio industrial (10%)	36,31
PRECIO DE VENTA DEL DISPOSITIVO sin IVA	446,57

Por lo tanto, podemos concluir que el precio de venta de uno de los sistemas de ayuda diseñados en este TFG sería de 446,57 € (sin IVA), y requeriría una inversión inicial de 36306,46€.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

ANEXOS

ANEXO 1. CÓDIGO SCRIPTS DE LA RASPBERRY

1. INTRODUCCIÓN

En este anexo, se adjunta el código desarrollado en el lenguaje de programación Python para la elaboración del prototipo sistema de ayuda que ha sido diseñado en este TFG. En total se han creado tres *scripts*, que se ejecutan en la Raspberry, con la finalidad de presentar el código de la manera más clara y organizada posible. Los dos primeros contienen las funciones necesarias para la detección de los elementos de activación mediante visión artificial comentados a lo largo del trabajo. El último, contiene el código de la aplicación principal, que define la interfaz de usuario y llama a las funciones de detección necesarias para interactuar con la persona que utiliza la aplicación y realizar las acciones necesarias.

2. SCRIPT DETECTA_GESTOFACIAL.PY

```
##

# detecta_gestofacial.py
# Este script contiene las dos funciones principales para detectar gestos faciales utilizando
# las 68 marcas faciales obtenidas mediante la librería dlib. Estas funciones son:
# det_bocaAbierta(): calcula el MAR en cada frame y si es superior a un umbral durante un
# número determinado de fotogramas, la variable booleana ELEM_ACTIVADO toma el valor True.
# det_pestañeo(): análoga a la anterior, pero calculando el EAR.

# También contiene tres funciones auxiliares a las que se llama dentro de cada una de las
# funciones principales para hacer operaciones básicas:
# dist_euclidea: calcula la distancia euclídea entre dos coordenadas de marcas faciales
# mouth_aspect_ratio: calcula el MAR
# eye_aspcet_ratio: calcula el EAR

##

# IMPORTAMOS LIBRERÍAS ##

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

import imutils
import time
import dlib
import cv2

def dist_euclidea(ptA, ptB):
    return np.linalg.norm(ptA - ptB)


def mouth_aspect_ratio(mouth):
    v1 = dist_euclidea(mouth[2], mouth[10])
    v2 = dist_euclidea(mouth[3], mouth[9])
    v3 = dist_euclidea(mouth[4], mouth[8])
    h = dist_euclidea(mouth[0], mouth[6])

    mar = (v1 + v2 + v3) / (3.0 * h)
    return mar

def eye_aspect_ratio(eye):
```

```
    v1 = dist_euclidea(eye[1], eye[5])
    v2 = dist_euclidea(eye[2], eye[4])
    h = dist_euclidea(eye[0], eye[3])

    ear = (v1 + v2) / (2.0 * h)
    return ear

def det_bocaAbierta(cap, detector, predictor, mStart, mEnd, CONT_FRAMESb, MAR_UMBRAL,
CONSEC_FRAMESb):

    while True:
        ELEM_ACTIVADO = False

        ## CAPTURA Y PREPROCESADO ##

        frame = cap.read()
        frame = imutils.resize(frame, width=350)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.medianBlur(gray, 9)  # filtro gaussiano

        ## POSPROCESADO ##

        # detección caras bounding box
        rects = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
                                          minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

        for (x, y, w, h) in rects:

            # detección 68 marcas faciales
            rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)

            #extraer puntos boca y calcular MAR
            boca = shape[mStart:mEnd]
            MAR = mouth_aspect_ratio(boca)


            # visualizar contorno boca
            mouthHull = cv2.convexHull(boca)
            cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)
            cv2.putText(frame, "MO: {:.3F}".format(MAR), (200, 50),
                                        cv2.FONT_HERSHEY_SIMPLEX,0.7, (0,0,255),2)

            if MAR >= MAR_UMBRAL:
                CONT_FRAMESb += 1
            else:
                if CONT_FRAMESb >= CONSEC_FRAMESb:
                    cv2.putText(frame, "BOCA ABIERTA :O", (200, 70),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                    ELEM_ACTIVADO = True
                    return ELEM_ACTIVADO

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

        if key == ord("q"):
            break


def det_pestañeo(cap, detector, predictor, iStart, iEnd, dStart, dEnd, CONT_FRAMESo,
EAR_UMBRAL, CONSEC_FRAMESo):

    while True:
        ELEM_ACTIVADO = False

        ## CAPTURA Y PREPROCESADO ##

        frame = cap.read()
        frame = imutils.resize(frame, width=350)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.medianBlur(gray, 9)  # filtro gaussiano

        ## POSPROCESADO ##

        # detección caras bounding box
        rects = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
```

```
                                    minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

        for (x, y, w, h) in rects:

            # detección 68 marcas faciales
            rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)

            #extraer puntos ojos y calcular EAR
            ojoIzq = shape[iStart:iEnd]
            ojoDer = shape[dStart:dEnd]
            EARizq = eye_aspect_ratio(ojoIzq)
            EARder = eye_aspect_ratio(ojoDer)

            EARprom = (EARizq + EARder) / 2.0 # EAR promedio

            # visualizar contornos ojos
            iOjoHull = cv2.convexHull(ojoIzq)
            dOjoHull = cv2.convexHull(ojoDer)
            cv2.drawContours(frame, [iOjoHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [dOjoHull], -1, (0, 255, 0), 1)
            cv2.putText(frame, "EAR: {:.3F}".format(EARprom), (200,30),
                            cv2.FONT_HERSHEY_SIMPLEX,0.7, (0,0,255),2)


            if EARprom >= EAR_UMBRAL:
                CONT_FRAMESo += 1
            else:
                if CONT_FRAMESo >= CONSEC_FRAMESo:
                    cv2.putText(frame, "PESTAÑEO ;)", (200, 70), cv2.FONT_HERSHEY_SIMPLEX,
                                                        0.7,(0, 0, 255), 2)
                    ELEM_ACTIVADO = True
                    return ELEM_ACTIVADO

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

        if key == ord("q"):
            break
```

3. SCRIPT DETECTA_MOVIMIENTO.PY

```
##

# detecta_movimiento.py
# Este script contiene las funciones que detectan el movimiento de un objeto (en este TFG
# será una parte del cuerpo del usuario, normalmente las extremidades superiores o la
# cabeza)si este se desplaza en la dirección horizontal con respecto a la cámara. Se definen
# dos funciones, una # para detectar los movimientos que se ejecutan hacia la izquierda
# (dir_izquierda ()) y otra, para los movimientos que se ejecutan hacia la derecha
# (dir_derecha).

# Ambas comparten el mismo código,basado en la técnica de sustracción de fondo. La única
# diferencia es que, en la primera, se trabaja con el espejo de la imagen captada por la
# PiCamera.

##

## IMPORTAMOS LIBRERÍAS ##

import cv2
import numpy as np
import time


def dir_izquierda(cap, kernel, anchoMitad, ant, gprev, centro_prev, dir):

    while True:
        ELEM_ACTIVADO = False

        next = cap.read()
        next = imutils.resize(next, width=350)

        # Hacemos el espejo de la imagen
```

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

        ## PREPROCESADO ##

        gnext = cv2.cvtColor(next, cv2.COLOR_RGB2GRAY)  # imagen de grises
        gnext = cv2.medianBlur(gnext, 9)  # filtro gaussiano

        ## POSPROCESADO ##

        dif = cv2.absdiff(gprev, gnext)  # resta
        # cv2.imshow("resta", dif)

        dif = cv2.inRange(dif, 15, 255)  # umbral
        dif = cv2.morphologyEx(dif, cv2.MORPH_OPEN, kernel)  # apertura
        dif = cv2.morphologyEx(dif, cv2.MORPH_CLOSE, kernel)  # cierre
        dif = cv2.dilate(dif, None, iterations=5)  # dilatación
        # cv2.imshow('imagen filtrada', dif)

        contornos, _ = cv2.findContours(dif, 1, 2)  #contornos de los píxeles en movimiento

        centro = [0, 0]
        n = 0

        for c in contornos:
            area = cv2.contourArea(c)
            if area > 11000:  # para no detectar objetos peque√±os en movimiento
                x, y, w, h = cv2.boundingRect(c)
                cv2.rectangle(next, (x, y), (x + w, y + h), (0, 255, 0), 2)# muestra
                                                                           # bounding box
                centro[0] += x + float(w) / 2
                centro[1] += y + float(h) / 2
                n += 1

                if centro[0] != 0 and centro[1] != 0:
                    centro = np.array(centro)
                    centro_new = centro_prev + 0.9 * (centro - centro_prev)

                    cntX = int(centro_new[0] / n)
                    cntY = int(centro_new[1] / n)
                    cv2.circle(next, (cntX, cntY), 15, (130, 50, 200), -1)

                    if centro_prev[0] > anchoMitad > centro_new[0]:
                        dir = "izquierda"
                        ELEM_ACTIVADO = True
                    else:
                        pass

                    centro_prev = centro_new
                    if dir != "":
                        return ELEM_ACTIVADO

        gprev = gnext
        cv2.imshow('next', next)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break


def dir_derecha(cap, kernel, anchoMitad, ant, gprev, centro_prev, dir):
    ELEM_ACTIVADO = False
    while True:

        next = cap.read()
        next = imutis.resize(next, width=350)

        ## PREPROCESADO ##

        gnext = cv2.cvtColor(next, cv2.COLOR_RGB2GRAY)  # imagen de grises
        gnext = cv2.medianBlur(gnext, 7)  # filtro gaussiano

        ## POSPROCESADO ##

        dif = cv2.absdiff(gprev, gnext)  # resta
        # cv2.imshow("resta", dif)

        dif = cv2.inRange(dif, 15, 255)  # umbral
        dif = cv2.morphologyEx(dif, cv2.MORPH_OPEN, kernel)  # apertura
```

```
        dif = cv2.morphologyEx(dif, cv2.MORPH_CLOSE, kernel)  # cierre
        dif = cv2.dilate(dif, None, iterations=5)  # dilatación
        # cv2.imshow('imagen filtrada', dif)

        contornos, _ = cv2.findContours(dif, 1, 2)  #contornos de los píxeles en movimiento

        centro = [0, 0]
        n = 0

        for c in contornos:
            area = cv2.contourArea(c)
            if area > 11000:  # para no detectar objetos pequeños en movimiento
                x, y, w, h = cv2.boundingRect(c)
                cv2.rectangle(next, (x, y), (x + w, y + h), (0, 255, 0), 2)
                centro[0] += x + float(w) / 2
                centro[1] += y + float(h) / 2
                n += 1

                if centro[0] != 0 and centro[1] != 0:
                    centro = np.array(centro)
                    centro_new = centro_prev + 0.9 * (centro - centro_prev)

                    cntX = int(centro_new[0] / n)
                    cntY = int(centro_new[1] / n)
                    cv2.circle(next, (cntX, cntY), 15, (130, 50, 200), -1)

                    if centro_prev[0] > anchoMitad > centro_new[0]:
                        dir = "derecha"
                        ELEM_ACTIVADO = True

                    else:
                        pass

                    centro_prev = centro_new
                    if dir != "":
                        return ELEM_ACTIVADO

        gprev = gnext
        cv2.imshow('next', next)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

3. SCRIPT MAINAPP.PY

```
##

# main_App.py
# versión FINAL para el prototipo. Este script define la interfaz diseñada para la aplicación
# que controla el sistema de ayuda. Se implementan dos hilos: hilo ventanas e hilo RF. Dentro
# de cada hilo, se llamará a la función que detecta el movimiento de activación necesario
# para que se ejecute la acción (pasar de ventana o enviar una señal por RF, respectivamente).
# En este script para el prototipo, dentro de cada hilo se llama a todas las posibles
# funciones, y se comentan todas las líneas de código menos la que se quiere probar (que
# sería la que el usuario elegiría para cada realizar cada acción).

##

## IMPORTAMOS LIBRERÍAS ##

from kivy.app import App
from kivy.lang.builder import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.core.window import Window
from kivy.clock import Clock
from gpiozero import LED, Button
from time import sleep
from smbus import SMBus
import threading
import detecta_gestofacial
import detecta_movimiento
import cv2

from imutils.video import VideoStream
from imutils import face_utils
```

```
import imutils
import time
import dlib
import numpy as np


## puesta en marcha de la cámara

cap = VideoStream(usePiCamera=True).start()
time.sleep(1.0)

# inicialización variables para detecta_movimiento()

prev = cap.read()
prev = imutils.resize(prev, width=350) # tamaño ventana
gprev = cv2.cvtColor(prev, cv2.COLOR_RGB2GRAY)  # escala de grises
gprev = cv2.medianBlur(gprev, 9)  # filtro gaussiano
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (15, 15)) # operaciones morfológicas
anchoMitad = prev.shape[1] // 2
ant = 0
centro_prev = np.array([0, 0])
dir = ""

# inicialización variables para detecta_gestofacial()
# boca
CONT_FRAMESb = 0
CONSEC_FRAMESb = 5
MAR_UMBRAL = 0.8

# ojos
CONT_FRAMESo = 0
CONSEC_FRAMESo = 3
EAR_UMBRAL = 0.31

## Cargamos el clasificador Haar Cascade de OpenCV y el predictor de marcas faciales de
dlib
print("[INFO] loading facial landmark predictor...")
detector = cv2.CascadeClassifier("/home/pi/kivy_venv/haarcascade_frontalface_default.xml")
predictor = dlib.shape_predictor("/home/pi/kivy_venv/shape_predictor_68_face_land-
marks.dat")
(iStart, iEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(dStart, dEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]


## diseño de las ventanas
screen_helper = """

ScreenManager:
    PersianaScreen:
    LuzScreen:
    CadenaScreen:

<PersianaScreen>:
    name: 'persiana'
    id: 1
    Label:
        text: 'SUBIR LA PERSIANA'
        font_size: 40
        halign: 'center'
        pos_hint: {'center_y':0.85}

    Image:
        id: imv
        source: "iconos/pers.png"
        size_hint_y: 0.8
        allow_stretch: True

<LuzScreen>:
    name: 'luz'
    id: 2
    Label:
        text: 'ENCENDER O APAGAR LA LUZ'
        halign: 'center'
        pos_hint: {'center_y':0.85}
        font_size: 40
    Image:
```

```
        id: imr
        source: "iconos/luz.png"
        size_hint_y: 0.8
        allow_stretch: True

<CadenaScreen>:
    name: 'cadena'
    id: 3
    Label:
        text: 'TIRAR DE LA CADENA'
        halign: 'center'
        pos_hint: {'center_y':0.85}
        font_size: 40
    Image:
        id: imr
        source: "iconos/cade.png"
        size_hint_y: 0.8
        allow_stretch: True

"""

Window.size = (800, 400) #800x480
Window.top = 35

#definir clases de ventanas
class PersianaScreen(Screen):
    pass
class LuzScreen(Screen):
    pass
class CadenaScreen(Screen):
    pass

# definir variables
boton = Button(26)
addr = 0x26 # dirección del bus
bus = SMBus(1) # indica /dev/ic2-1


# funciones
def rf_envia():
    print("INICIO rf_envia")
    ventana_actual = sm.current

    if ventana_actual == 'persiana':
        bus.write_byte(addr, 0x1)
        print('señal RF 1111 enviada - persiana')

    elif ventana_actual == 'luz':
        bus.write_byte(addr, 0x2)
        print('señal RF 2222 enviada - luz')

    elif ventana_actual == 'cadena':
        bus.write_byte(addr, 0x3)
        print('señal RF 3333 enviada - cadena')


def cambiar_ventana():

        print("Función SWICHT")

        if sm.current == 'persiana':
            sm.current = 'luz'
            print("cambio a pantalla LUZ")


        elif sm.current == 'luz':
            sm.current = 'cadena'
            print("cambio pantalla a CADENA")

        else:
            sm.current = 'persiana'
            print("cambio pantalla a PERSIANA")

def hilo_ventana():
    print("HILO VENTANA")
    while True:
        ## elegimos elemento de activación
```

```
        #botón
        boton.when_pressed = cambiar_ventana

        #boca
        #ELEM_ACTIVADO = detecta_gestofacial.det_bocaAbierta(cap, detector, predictor,
               mStart, mEnd, CONT_FRAMESb, MAR_UMBRAL, CONSEC_FRAMESb)

        #ojos
        #ELEM_ACTIVADO = detecta_gestofacial.det_pestañeo(cap, detector, predictor, iStart,
                       iEnd, dStart, dEnd, CONT_FRAMESo, EAR_UMBRAL, CONSEC_FRAMESo)

        #movimiento mano (izq o drch)
        #ELEM_ACTIVADO = detecta_movimiento.dir_izquierda(cap, kernel, anchoMitad, ant,
               gprev, centro_prev, dir)
        #ELEM_ACTIVADO = detecta_movimiento.dir_derecha(cap, kernel, anchoMitad, ant,
               gprev, centro_prev, dir)

        ## ejecutamos acción
        #if ELEM_ACTIVADO:
            #print("¡¡ ELEMENTO DE ACTIVACIÓN DETECTADO !!")
            #rf_envia()

def hilo_rf():
    print("HILO RF")
    while True:
        ## elegimos elemento de activación

        #botón
        #boton.when_pressed = rf_envia

        #boca
        ELEM_ACTIVADO = detecta_gestofacial.det_bocaAbierta(cap, detector, predictor,
               mStart, mEnd, CONT_FRAMESb, MAR_UMBRAL, CONSEC_FRAMESb)

        #ojos
        #ELEM_ACTIVADO = detecta_gestofacial.det_pestañeo(cap, detector, predictor, iStart,
                       iEnd, dStart, dEnd, CONT_FRAMESo, EAR_UMBRAL, CONSEC_FRAMESo)

        #movimiento mano (izq o drch)
        #ELEM_ACTIVADO = detecta_movimiento.dir_izquierda(cap, kernel, anchoMitad, ant,
               gprev, centro_prev, dir)
        #ELEM_ACTIVADO = detecta_movimiento.dir_derecha(cap, kernel, anchoMitad, ant,
               gprev, centro_prev, dir)

        ## ejecutamos acción
        if ELEM_ACTIVADO:
            print("¡¡ ELEMENTO DE ACTIVACIÓN DETECTADO !!")
            rf_envia()

## creamos los hilos
h_ventana = threading.Thread(target = hilo_ventana)
h_RF = threading.Thread(target = hilo_rf)

## clase App principal
sm = ScreenManager()

class RFDomoticsApp(App):
    def build(self):

        Window.clearcolor = (183/255, 204/255, 177/255, 0.7)

        screen = Builder.load_string(screen_helper)
        scr_p = PersianaScreen(name = 'persiana')
        scr_l = LuzScreen(name = 'luz')
        scr_c = CadenaScreen(name = 'cadena')

        sm.add_widget(scr_p)
        sm.add_widget(scr_l)
        sm.add_widget(scr_c)

        # pasar ventanas automáticamente cada X segundos
        #print("llamamos a clock")
        #Clock.schedule_interval(self.cambiar_ventana, 5)

        ## iniciamos hilos
        h_RF.deamon = True
```

```
        h_RF.start()

        h_ventana.deamon = True
        h_ventana.start()

        return sm

    #para pasar pantallas automáticamente
    def cambiar_ventana(self, *args):

        print("Función cambiar_ventana AUTOMÁTICA")

        if sm.current == 'persiana':
            sm.current = 'luz'
            print("cambio a pantalla LUZ")


        elif sm.current == 'luz':
            sm.current = 'cadena'
            print("cambio pantalla a CADENA")

        else:
            sm.current = 'persiana'
            print("cambio pantalla a PERSIANA")

if __name__ == '__main__':
    RFDomoticsApp().run()
```

ANEXO 2. CÓDIGO SKETCHS DE LAS PLACAS DIGISPARK

1. INTRODUCCIÓN

Cada una de las placas Digispark ha sido programada desde el IDE de Arduino para dotarla de las funcionalidades requeridas. A continuación, se adjunta el código implementado en la placa emisora y las tres placas receptoras utilizadas para la elaboración del prototipo descrito a lo largo del trabajo. Dicho código, está basado en los ejemplos proporcionados en [39].

2. SKETCH DE LA PLACA DIGISPARK EMISORA

```
#include <MANCHESTER.h>
#include "TinyWireS.h"
#define I2C_SLAVE_ADDR  0x26

void setup() {
  TinyWireS.begin(I2C_SLAVE_ADDR);
  MANCHESTER.SetTxPin(4);

  //Set the pin's modes
  pinMode(1, OUTPUT);
}

void loop() {
  byte byteRcvd = 0;
  if (TinyWireS.available()){           // got I2C input
    byteRcvd = TinyWireS.receive();     // get the byte from master

    if(byteRcvd == 1 ) { //persiana

      unsigned int data = 1111;
      MANCHESTER.Transmit(data);
      pinMode(1, OUTPUT);
      digitalWrite(1, HIGH);
      delay(250);
      digitalWrite(1,LOW);
      delay(250);
    }
    else if (byteRcvd == 2) { //luz
      unsigned int data = 2222;
      MANCHESTER.Transmit(data);
      pinMode(1, OUTPUT);
      digitalWrite(1, HIGH);
      delay(250);
      digitalWrite(1,LOW);
      delay(250);
    }
    else if (byteRcvd == 3) { //cadena
      unsigned int data = 3333;
      MANCHESTER.Transmit(data);
      pinMode(1, OUTPUT);
      digitalWrite(1, HIGH);
      delay(250);
      digitalWrite(1,LOW);
      delay(250);
    }
  }
}
```

3. SKETCH PLACA RECEPTORA PERSIANA

```
#include <MANCHESTER.h>
#define RxPin 2

void setup() {
  // Set digital TX pin
  MANRX_SetRxPin(2);
  // Prepare interrupts
  MANRX_SetupReceive();
  // Begin receiving data
  MANRX_BeginReceive();

  pinMode(4, OUTPUT);
  pinMode(1, OUTPUT);
}

void loop() {
  if (MANRX_ReceiveComplete()) {
    unsigned int data = MANRX_GetMessage();
    MANRX_BeginReceive();

    if (data == 1111 || data == 111 || data == 11) { //comprobamos que le habla a persiana
      digitalWrite(4, !digitalRead(4)); //si está subida, baja (viceversa)
    }

    digitalWrite(1, HIGH);
    delay(250);
    digitalWrite(1,LOW);
    delay(250);
  }
}
```

4. SKETCH PLACA RECEPTORA LUZ

```
#include <MANCHESTER.h>
#define RxPin 2

void setup() {
  // Set digital TX pin
  MANRX_SetRxPin(2);
  // Prepare interrupts
  MANRX_SetupReceive();
  // Begin receiving data
  MANRX_BeginReceive();

  pinMode(4, OUTPUT);
  pinMode(1, OUTPUT);
}

void loop() {
  if (MANRX_ReceiveComplete()) {
    unsigned int data = MANRX_GetMessage();
    MANRX_BeginReceive();

    if (data == 2222 || data == 222 || data == 22) { //comprobamos que le habla a luz
      digitalWrite(4, !digitalRead(4)); //si está encendida, apaga (viceversa)
    }

    digitalWrite(1, HIGH);
    delay(250);
    digitalWrite(1,LOW);
    delay(250);
  }
}
```

5. SKETCH PLACA RECEPTORA CADENA

```
#include <MANCHESTER.h>
#define RxPin 2

void setup() {
  // Set digital TX pin
  MANRX_SetRxPin(2);
  // Prepare interrupts
  MANRX_SetupReceive();
  // Begin receiving data
  MANRX_BeginReceive();

  pinMode(4, OUTPUT);
  pinMode(1, OUTPUT);
}

void loop() {
  if (MANRX_ReceiveComplete()) {
    unsigned int data = MANRX_GetMessage();
    MANRX_BeginReceive();

    if (data == 3333 || data == 333 || data == 33) { //comprobamos que le habla a cadena
      digitalWrite(4, HIGH); //estirar
      delay(700);
      digitalWrite(4, LOW);
    }

    digitalWrite(1, HIGH);
    delay(250);
    digitalWrite(1,LOW);
    delay(250);
  }
}
```