GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

NIRA: Diseño de un sistema de seguridad

Autor: Nicolás Santiago Heinrich

Director: Álvaro Pérez Bello

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título NIRA: Diseño de un sistema de seguridad en la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/2021 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Nicolás Santiago Heinrich Fecha: ..06../ ..07../ ..2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Pérez Bello Fecha: ..06../ ..07../ ..2021

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

NIRA: Diseño de un sistema de seguridad

Autor: Nicolás Santiago Heinrich

Director: Álvaro Pérez Bello

Madrid

Agradecimientos

Este trabajo ha sido posible gracias al apoyo recibido a lo largo de estos duros años de carrera. No han sido fáciles, pero gracias a aquellos que me han ayudado, he podido seguir hacia adelante y tener una motivación mayor para alcanzar mis objetivos.

En primer lugar, quiero agradecer a mi familia por el apoyo y esfuerzo realizado para poder estudiar en esta universidad. Este trabajo ha sido posible gracias a ellos y estoy eternamente agradecido.

En segundo lugar, quiero agradecer a todos mis amigos que me han apoyado en estos duros años. Algunos años más duros que otros, pero siempre han estado ahí para ayudarme a seguir hacia delante. Gracias a ellos he podido mantener una mentalidad más positiva y llevar a cabo todo lo que me he propuesto.

También quiero a agradecer a mi director de proyecto Álvaro Pérez por haberme dado la oportunidad de realizar este proyecto. Siempre quise aplicar todos los conocimientos adquiridos estos últimos años en un proyecto práctico. También quiero agradecerle por haberme guiado desde el principio de este curso y por ofrecerme todas las herramientas necesarias para llevarlo a cabo.

Por último, muchas gracias a Paloma por haberme ayudado con los problemas relacionados con SmartWorks y siempre haber estado dispuesta ayudarme rápida y eficazmente.

NIRA: DISEÑO DE UN SISTEMA DE SEGURIDAD

Autor: Santiago Heinrich, Nicolás.
Director: Pérez Bello, Álvaro.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El proyecto consiste en el diseño de un sistema de seguridad que cuente con una cámara para el reconocimiento de objetos y para la detección de caras y con un lector de huellas dactilares. Además, debe contar con un altavoz con la finalidad de guiar al usuario y con una plataforma donde monitorizar el estado del sistema. Este sistema tendrá como objetivo dotar a cualquier tipo de residencia un mecanismo de seguridad eficiente para poder acceder a ella a un coste muy asequible.

Palabras clave: Yolo, OpenCV, Python

1. Introducción

En los últimos años, gracias al avance de la tecnología se ponen a la venta un mayor número de productos conectados entre sí. Por un lado, se debe a la mejora de la conectividad con tecnologías como el 5G y, por otro lado, gracias al avance del hardware con procesadores cada vez más pequeños y eficientes.

En este mundo de productos conectados también se encuentran las cámaras de seguridad con conectividad Wifi, las cuales están sufriendo un gran auge debido a los motivos explicados anteriormente y al creciente interés por monitorizar el hogar. La finalidad es proteger a las personas que viven en la vivienda, junto con sus objetos de valor como pueden ser joyas, documentos con información muy relevante, por lo que se crea la necesidad de saber el estado actual del domicilio y detectar posibles incidentes como destrozos provocados por animales o inclemencias climáticas. Además de querer saber cuál es el estado, se quiere prevenir el acceso de personas, que no habitan en la propiedad, a través de lectores de huellas, detectando a las personas que acceden y recibir un aviso a través del correo electrónico.

Uno de los problemas de los productos de seguridad que se encuentran a la venta hoy en día, es su elevado precio. A pesar del gran avance de la tecnología y la reducción de los costes no se aprovecha para diseñar un producto completamente centralizado e integrado. Se tienen que comprar dispositivos distintos para obtener un conjunto de utilidades. Además, no se aprovecha esta integración para ofrecer aplicaciones adicionales.

2. Definición del proyecto

En este proyecto se desarrolla un sistema completo de seguridad con el objetivo de implementarlo en cualquier residencia. Para ello, se ha utilizado principalmente una cámara con capacidad de visión nocturna, que obtiene la imagen necesaria para su posterior análisis en una Raspberry Pi. Dicho análisis consiste en el uso del algoritmo YOLO [1] para la detección de objetos y, al detectar una persona, se emplea una serie de algoritmos [2] para conseguir reconocerla.

Por otro lado, el sistema cuenta con un lector de huellas dactilares para confirmar la identidad de la persona detectada por cámara, además de disponer de un altavoz que tiene la finalidad de guiar al usuario.
Asimismo, la Raspberry se encarga de enviar una imagen de las personas detectadas a través de correo electrónico además de facilitar vídeo en tiempo real a través de la web desde cualquier punto.
Por otro lado, el sistema está conectado a la plataforma de Altair llamada SmartWorks [3], en la cual se puede monitorizar el estado del sistema de seguridad.
Por último, se tiene como objetivo diseñar una carcasa que integre todos estos componentes.

3. Descripción del modelo/sistema/herramienta

El sistema está compuesto por una parte de hardware y otra de software. En la parte de hardware se encuentra una Raspberry Pi 4B, una cámara con visión nocturna, un lector de huellas y un altavoz. En el software se ha usado Python como lenguaje de programación y cuenta con el algoritmo YOLO para el reconocimiento de objetos, algoritmos para la detección facial, la plataforma de Altair, y todas las librerías de Python necesarias para ejecutar todo el sistema.

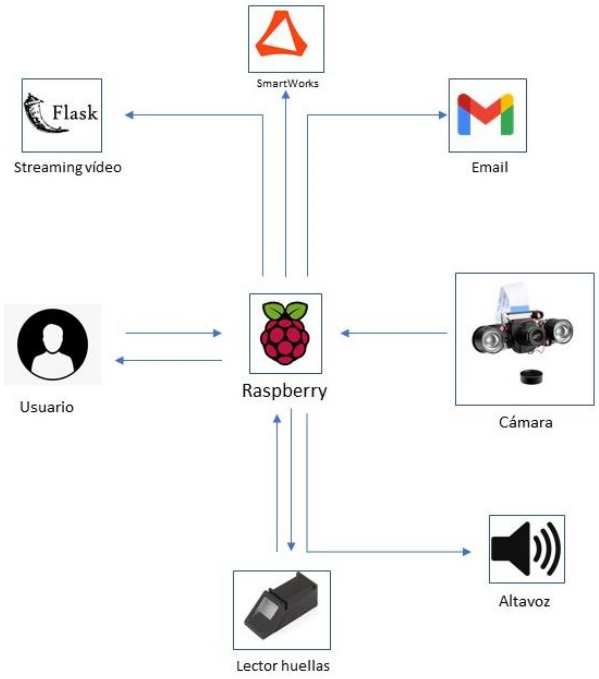


Ilustración 1 - Esquema general del sistema de seguridad

El funcionamiento consiste en el primer arranque de la Raspberry en el cual se tendrá que ejecutar el programa principal y en el que aparece un menú donde elegir configurar persona a detectar, correo electrónico, huella dactilar y ejecutar el sistema de seguridad. Una vez en funcionamiento, el programa está continuamente analizando la imagen para detectar objetos y, en el caso de detectar una persona un número suficiente de veces se pasa al algoritmo de detección facial que la reconocería y pondría en marcha el lector de huellas dactilares. Mientras se está ejecutando el programa, se están

enviando continuamente datos de la Raspberry a la plataforma SmartWorks y se puede ver la imagen por streaming gracias al empleo del framework Flask. Adicionalmente, una vez que se detecta una persona, se envía un email al correo electrónico registrado y se avisa en los casos de acceso al domicilio. Finalmente, para facilitar tanto la configuración del sistema como la detección del usuario, el sistema cuenta de un altavoz con el que se va guiando al usuario.

4. Resultados

En este proyecto se han conseguido alcanzar todos los objetivos planteados. Además, se han podido implementar ciertas funcionalidades que no estaban previstas. Los resultados se pueden resumir en los siguientes puntos:

- Reconocer objetos con un dispositivo con capacidad de procesamiento limitado.
- Reconocer facialmente a una velocidad y fiabilidad razonables.
- Implementar un lector de huellas altamente fiable como segunda capa de seguridad.
- Implementar un altavoz para guiar al usuario tanto en la configuración como en el control de acceso a la vivienda.
- Implementar un sistema de refrigeración eficiente para reducir la temperatura del sistema.
- Integrar todos los componentes mencionados anteriormente en una carcasa diseñada específicamente para este sistema de seguridad.
- Lograr diseñar un sistema que consuma la mitad de energía que sistemas similares en el mercado.
- Facilitar al usuario final la configuración del sistema e implementar una plataforma que ofrece vídeo en tiempo real de la cámara del sistema.
- Ofrecer al usuario una plataforma donde monitorizar el estado del sistema de seguridad.
- Ofrecer un sistema de seguridad completo con una gran cantidad de funcionalidades a un precio muy competitivo.

5. Conclusiones

En este proyecto se ha conseguido diseñar un sistema de seguridad completo desde el software hasta el hardware y para ello se han tenido que aplicar conocimientos de diversas áreas. Para diseñar este sistema, se ha empleado una Raspberry Pi 4B como componente central debido a que ofrece el mejor rendimiento en relación calidad precio y ofrece una gran cantidad de puertos vitales para conectar todos los componentes que conforman el sistema. Por otro lado, se ha logrado diseñar un sistema completo a un precio muy competitivo y completamente funcional. Para alcanzar estos objetivos se ha tenido que aprender un nuevo lenguaje de programación, entender el funcionamiento de algoritmos de detección de objetos y reconocimiento de caras y aplicar muchos conocimientos adquiridos a lo largo de la carrera.

A pesar de haber conseguido diseñar con éxito este sistema de seguridad, tiene mucho margen de mejora en varios aspectos. Por otro lado, se le pueden añadir una gran cantidad de funcionalidades nuevas que pueden resultar muy interesantes para el usuario final.

6. Referencias

[1] YOLO Algorithm. Último acceso: 14/04/2021. https://pjreddie.com/darknet/yolo/

[2] Geitgey, A. Machine learning is fun. Part 4. Último acceso: 02/02/2021. Mayo, 2014. https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471

[3] SmartWorks. Último acceso: 02/07/2021. https://www.altairsmartworks.com/

NIRA: DESIGN OF A SECURITY SYSTEM

Author: Santiago Heinrich, Nicolás.
Supervisor: Pérez Bello, Álvaro.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

The project consists of the design of a security system that counts with a camera for the object detection and for the facial recognition and with a fingerprint sensor. Furthermore, it must have a speaker with the objective to guide the user and a platform where to monitor with ease the state of the system. This system will have to provide to any residence an efficient security system to access at a reduced cost.

Keywords: Yolo, OpenCV, Python

1. Introduction

In the last years, thanks to the advance of the technology a larger number of connected products are sold. On the one hand, this is because of the improvement of the connectivity with technologies such as 5G, and on the other hand, thanks to the progress of the hardware with smaller and more efficient processors.
In this world of connected products, security cameras with Wi-Fi connectivity can be found and they are suffering a big peak because of the reasons mentioned before and because of the increasing interest to monitor the house. The purpose is to protect the people that live in the house and the objects with value such as jewels and documents with important information. This creates the necessity to know the state of the residence and to detect incidents made by animals or climate inclemency. In addition to this, the access of people is also wanted to prevent using fingerprint sensors and by also receiving notifications by email.
One of the problems of the existent security systems is their price. Despite the improvement of technology and the reduction of costs, these are not used to design a completely centralized and integrated product. It is needed to buy several different devices to obtain a group of utilities. Furthermore, this integration is not used to offer additional functionalities.

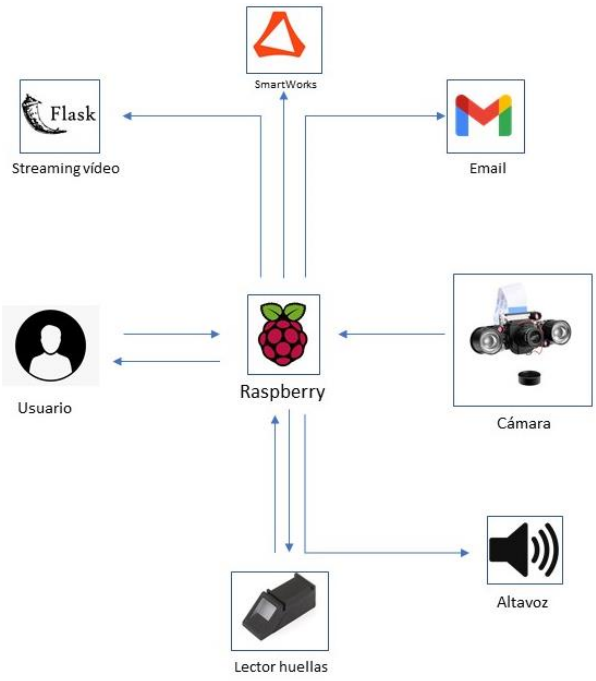
2. Definition of the project

In this project a complete security system is developed with the objective of implementing it in any residence. For that, a night vision camera has mainly been used, which obtains the necessary image for after analyzing it in a Raspberry Pi. This analysis consists of the use of the YOLO algorithm [1] for the object detection and, when a person is detected, a series of algorithms are used for recognizing the person.
On the other side, the system counts with a fingerprint sensor to confirm the identity of the identified person by the camera. In addition to this, the system has a speaker that guides the user.
Additionally, the Raspberry Pi sends an image of the detected persons by email and provides streaming video at any point.
On the other hand, the system is connected to an Altair platform called SmartWorks [3], where the state of the security system can be monitored.

Lastly, the design of the framework that will contain all the security system components will be designed.

3. Description of the system

The security system is composed of a hardware and a software part. In the hardware part is the Raspberry Pi 4B, a night vision camera, a fingerprint sensor, and a speaker. In the software part Python has been used as the programming language and it counts with the YOLO algorithm for the object recognition, facial detection algorithms, the Altair platform, and all the needed python libraries for executing the system.
The functioning consists of the first execution of the Raspberry and the execution of the main program. When the program is executed, a menu will firstly appear where to choose between configuring the person to detect, the email, the fingerprint and to carry out the security system. When the system is working, the program will continuously analyze the image for detecting objects and in the case a person has been detected several times, the facial detection algorithm starts to work for recognizing this person and it would activate the fingerprint sensor. While the program is working, data is continuously sent to the SmartWorks platform, and the image can be seen thank to use of the Flask framework. Furthermore, when a person is detected, an email is sent to the registered email account, and it also notify when a person accesses the place of residence. Finally, to facilitate the system configuration and the user detection, the system has a speaker to guide the user.



Picture 2 – General scheme of the security system

4. Results

In this project it has been possible to achieve all the suggested goals. Furthermore, it has been possible to implement some functionalities that were not planned. The results can be summarized in the following points:

- Recognize objects with a limited processing device.

- Detect facially at a reasonable speed and reliability.
- Set up a highly reliable fingerprint sensor as a second layer of security.
- Implement a speaker to guide the user in the configuration process and in the access of the user to the residence.
- Set up an efficient cooling system to reduce the system temperature.
- Integrate all the mentioned components in a shell specifically designed for this security system.
- Achieve to design a system that consumes half of the energy consumed by similar security systems.
- Ease the configuration process of the system to the final user and implement a platform to view in real time the image of the camera.
- Provide to the user a platform where to monitor the state of the security system.
- Provide a complete security system with a variety of functionalities at a very competitive price.

5. Conclusions

In this project a complete security system has been designed that goes from software to hardware and for achieving this goal it has been needed to apply knowledge from many areas. To design this system, a Raspberry Pi 4B has been used as the main component because it offers the best performance at a price cost ratio, and it offers many ports to connect all the integrated components in this security system. On the other hand, a complete security system with a very competitive price and completely functional one has been designed. To achieve these goals a new programming language has been needed to learn, the functioning of the object detection and facial detection algorithms have been needed to learn how they work and to apply all the learned knowledge of the degree. Although it has been achieved to design with success this security system, it can be improved in many aspects. On the other hand, many additional functionalities can be added, and they can result very useful for the final user.

6. References

[1] YOLO Algorithm. Last accessed: 14/04/2021. https://pjreddie.com/darknet/yolo/

[2] Geitgey, A. Machine learning is fun. Part 4. Last accessed: 02/02/2021. May 2014. https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471

[3] SmartWorks. Last accessed: 02/07/2021. https://www.altairsmartworks.com/

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

En los últimos años, gracias al avance de la tecnología se ponen a la venta un mayor número de productos conectados entre sí. Por un lado, se debe a la mejora de la conectividad con tecnologías como el 5G y, por otro lado, gracias al avance del hardware con procesadores cada vez más pequeños y eficientes. Esto último ha permitido a los fabricantes producir todo tipo de productos conectados desde bombillas con conectividad Wifi a dispositivos como Alexa con capacidad de reconocimiento de voz. No solo conectarlos, también son capaces de ejecutar ciertas acciones como realizar búsquedas en la web, encender y apagar una bombilla o la calefacción. Además, debido al creciente interés por los productos conectados, los fabricantes de procesadores se están centrando cada vez más en la fabricación de chips que ya incluyen módulos 5G y así satisfacer toda la demanda generada por todos estos productos.

En este mundo de productos conectados también se encuentran las cámaras de seguridad con conectividad Wifi, las cuales están sufriendo un gran auge debido a los motivos explicados anteriormente y al creciente interés por monitorizar el hogar. La finalidad es proteger a las personas que viven en la vivienda, junto con sus objetos de valor como pueden ser joyas, documentos con información muy relevante, por lo que se crea la necesidad de saber el estado actual del domicilio y detectar posibles incidentes como destrozos provocados por animales o inclemencias climáticas. Además de querer saber cuál es el estado, se quiere prevenir el acceso de personas, que no habitan en la propiedad, a través de lectores de huellas, detectando a las personas que acceden y recibir un aviso a través del correo electrónico.

Uno de los problemas de los productos de seguridad que se encuentran a la venta hoy en día, es su elevado precio. A pesar del gran avance de la tecnología y la reducción de los costes no se aprovecha para diseñar un producto completamente centralizado e integrado. Se tienen que comprar aparatos distintos para obtener un conjunto de utilidades. Además, no se aprovecha esta integración para ofrecer aplicaciones adicionales.

1.1 MOTIVACIÓN DEL PROYECTO

Tras observar las alternativas que existen hoy en día en el mercado para los problemas planteados anteriormente, el objetivo de este proyecto se ha enfocado en encontrar una alternativa segura y barata que sea capaz de proporcionar las soluciones que ofrecen las cámaras de seguridad con reconocimiento facial, vídeo en tiempo real y sistema de alertas junto con reconocimiento de huellas dactilares o de tarjetas integrado en un mismo producto y a un precio más asequible.

Para conseguir estos objetivos, se quiere buscar un dispositivo que funcione como cerebro del sistema de seguridad. Además, debe ser barato y disponer de suficiente capacidad de cálculo para realizar tareas tan complejas como detectar objetos y personas. También debe permitir programar con el lenguaje de programación Python para poder usar librerías que permiten ofrecer funciones adicionales al sistema de seguridad. Por otro lado, este dispositivo debe contar con numerosos puertos a los que conectar los diferentes periféricos necesarios para poder implementar un sistema de este tipo.

Adicionalmente, se quiere buscar los periféricos más idóneos para las tareas requeridas: una cámara con capacidad de visión nocturna para poder usarlo tanto de día como de noche, un lector de huellas con el que ofrecer una capa de seguridad adicional, un altavoz con el que indicar al usuario lo que debe realizar y un sistema de refrigeración.

Por otro lado, para poder integrar todos estos componentes, se desea diseñar una carcasa que puede ser fabricada mediante impresión 3D y de esta forma poder personalizar el máximo posible el sistema y ofrecer un producto completo. El diseño debe tener en cuenta factores como ventilación, espacio entre componentes y materiales.

1.2 ESTRUCTURA DEL DOCUMENTO

Este documento se organiza en ocho capítulos. El primero es este y en él se realiza una introducción del proyecto y posteriormente se explica la motivación para realizarlo.

En el segundo capítulo se realiza una descripción de las tecnologías empleadas para la realización de este proyecto. Para ello, se describe tanto la parte de hardware como de software. Cabe destacar que, para un mejor entendimiento del desarrollo del proyecto, se ha explicado con cierto detalle el funcionamiento de los algoritmos de detección empleados.

En el tercer capítulo se pone en contexto este proyecto dentro de la situación actual del mercado de los sistemas de seguridad. Se analizan desde los sistemas más básicos a los más avanzados para observar qué productos se encuentran disponibles hoy en día y qué se pretende mejorar. Por otro lado, también se analizan los distintos algoritmos de detección existentes actualmente.

Una vez puesto en contexto al lector y explicado las tecnologías empleadas para este proyecto, se explica en el capítulo cuatro, los objetivos que se quieren alcanzar y qué metodología se va a emplear para lograr desarrollar este proyecto. En el siguiente capítulo se explica el desarrollo del proyecto mostrando la funcionalidad de cada uno de los componentes del sistema en el apartado de hardware y las funciones empleadas en el apartado de software. En el sexto capítulo se describen los resultados obtenidos tanto a nivel de seguridad del sistema como de consumo energético. Finalmente, se aclaran los objetivos alcanzados y las posibles mejoras que se pueden realizar en futuros proyectos.

Por último, se encuentra la bibliografía empleada en este documento y dos anexos. En el primer anexo se explica brevemente el proceso de instalación de los diversos programas empleados en este proyecto y, en los casos de configuración de mayor dificultad, se entra en más detalle. Por otro lado, en el anexo II se encuentran los códigos empleados en el sistema de seguridad y, por otro, los códigos usados en la plataforma SmartWorks.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Para entender correctamente todo el proyecto, en este apartado se encuentran las descripciones de las tecnologías usadas tanto a nivel de software como de hardware.

2.1 SOFTWARE

2.1.1 PYTHON 3.7

Python es uno de los ejes principales del proyecto, puesto que va a ser el lenguaje de programación que se ha empleado para ejecutar todo el programa. Destaca por ser uno de los lenguajes más usados en la actualidad gracias a la facilidad que ofrece para acceder a otros códigos desarrollados previamente (librerías), poder implementarlos en este proyecto y que cumplan con todas las funciones necesarias para poner en marcha todo el sistema de seguridad.

La versión de Python que se ha utilizado es la 3.7 puesto que es la que menos conflictos genera a la hora de ejecutar OpenCV, la cual va a ser una de las librerías centrales para el reconocimiento de objetos y la detección de caras.

Ilustración 3: Python

2.1.2 PYCHARM COMMUNITY EDITION

Para la programación de todo el código se ha empleado el entorno de desarrollo (IDE) PyCharm, desarrollado por la empresa JetBrains. En concreto, se ha utilizado la versión Community Edition, la cual es gratuita y ofrece todo lo necesario para desarrollar correctamente todo el programa. Tiene versiones para Windows, macOS y GNU/Linux.

Se podría haber optado por diversos entornos de desarrollo. En este caso, se ha escogido PyCharm por su comodidad y sencillez, además de que ofrece todo lo necesario para instalar de forma sencilla todas las librerías requeridas.

Como se comentará más adelante, el sistema operativo de Raspberry (Raspberry Pi OS), incluye por defecto entornos de desarrollo para programar directamente sobre ella, pero resulta menos cómodo por la menor velocidad de ejecución del programa. Esto se ha dado especialmente cuando el proyecto era más complejo.

Ilustración 4: PyCharm

2.1.3 RASPBERRY PI IMAGER

Se trata de un programa desarrollado por la propia fundación Raspberry. Se emplea para crear la partición en una microSD donde se va a alojar el sistema operativo Raspberry Pi OS. Éste último se ha usado para ejecutar todo el sistema.

2.1.4 RASPBERRY PI OS

Como se ha comentado anteriormente, se trata del sistema operativo oficial de Raspberry, y es el que se emplea para ejecutar todo el sistema de seguridad. Existen tres versiones de Raspberry Pi OS:

- Raspberry Pi OS Lite: Se trata de la versión que menos espacio ocupa. Además, no tiene interfaz gráfica y se interactúa a través de una terminal.
- Raspberry Pi OS with Desktop: Dispone de interfaz gráfica y trae instalado el software básico.
- Raspberry Pi OS Full: Dispone de interfaz gráfica y lleva instalados programas adicionales.

Ilustración 5: Raspberry Pi OS

Debido a las limitaciones de procesamiento y de memoria RAM de la Raspberry se pensó inicialmente en usar la versión más ligera. Sin embargo, más adelante es necesario usar una interfaz de usuario para configurar la cara de la persona inicialmente y para comprobar que se realiza la foto correctamente. Por esto último se ha escogido la versión con interfaz gráfica y con el software básico instalado.

Es cierto que existen otro tipo de sistemas operativos que se enfocan únicamente en la detección de objetos a través de vídeo. En este sentido, están más optimizados, pero particularmente para este proyecto tenían ciertas limitaciones para implementar otro tipo de módulos y dificultaban su uso.

2.1.5 PUTTY

Es un programa de código abierto que permite conectar al usuario a través de SSH (Secure Shell Protocol) a la Raspberry de forma remota. Este programa ofrece un canal seguro entre el ordenador y la Raspberry donde toda la información intercambiada entre ambos equipos está cifrada y servirá para realizar las primeras configuraciones de la Raspberry.

Ilustración 6: PuTTY

2.1.6 VNC VIEWER

VNC Viewer es un programa de la empresa RealVNC, la cual usa la tecnología VNC (Virtual Network Computing) y que se ha empleado para poder acceder de forma remota a la Raspberry. La gran ventaja de este sistema respecto a PuTTY es que se puede visualizar toda la interfaz gráfica de Raspberry Pi OS y resulta más práctico para trabajar con la Raspberry que con una terminal.

Ilustración 7: VNC Viewer

2.1.7 WINSCP

Es un programa de código abierto, el cual, solo se puede ejecutar en Windows. Este software también emplea la conexión SSH y se usa para transferir archivos de forma sencilla, rápida y en remoto. De esta forma se elimina la necesidad de sacar la microSD de la Raspberry y conectarla al ordenador para realizar el intercambio de archivos desde el ordenador a la Raspberry. Este programa se ha usado para cuando se quieran probar nuevos códigos en la Raspberry Pi.

Ilustración 8: WinSCP

2.1.8 YOLO V3 ALGORITHM

Se trata de uno de los ejes principales del sistema de seguridad. Se encarga de realizar en tiempo real, la búsqueda de personas y da paso a otros algoritmos en cuanto detecta una. YOLO (You Only Look Once) es un algoritmo que revolucionó la detección de objetos por su gran velocidad a la hora de realizar el procesamiento. Anteriormente existían algoritmos que funcionaban muy bien, pero a la hora de realizar detección de objetos en tiempo real presentaban grandes dificultades debido a su bajo rendimiento. Este tipo de algoritmos se

comentarán en la sección Estado de la Cuestión. En este proyecto, se ha empleado para detectar personas rápidamente y con una precisión aceptable teniendo en cuenta el hardware disponible. Más adelante, se cuenta con otros algoritmos especializados en detección de caras, los cuales son más eficaces para este propósito y son los que van a permitir el acceso de una persona en un domicilio.

Según [4], los algoritmos que se empleaban antes de la aparición de YOLO estaban basados en la clasificación de partes de la imagen con una red neuronal convolucional (CNN, Convolutional Neural Network) y eligen aquellas regiones que resultan más interesantes para analizar. En cambio, YOLO está basado en regresión, es decir, en lugar de elegir partes interesantes de la imagen, realiza predicciones de clases para toda la imagen entera en una sola ejecución.

Para entender el código que se ha empleado y que se va a mostrar más adelante, resulta necesario explicar una serie de conceptos, la arquitectura y la estructura interna del algoritmo YOLO. Cabe destacar que hay distintas versiones de YOLO [5], y, en este caso, se ha usado YOLO v3, cuya arquitectura es más compleja que las versiones anteriores para satisfacer una serie de requisitos que se comentarán más adelante. Además de emplear la tercera versión, también se ha usado una versión con una estructura más sencilla y que requiere de menor capacidad de cálculo. A pesar de que su precisión no es tan elevada, en este proyecto, no se necesita una gran precisión para esta parte del sistema de seguridad.

En el Estado de la Cuestión, además de comentar algoritmos basados en clasificación, se comentarán las otras dos versiones más potentes que hay de YOLO v3 y de los cuales se han tenido que prescindir debido a la limitada capacidad de procesamiento del sistema.

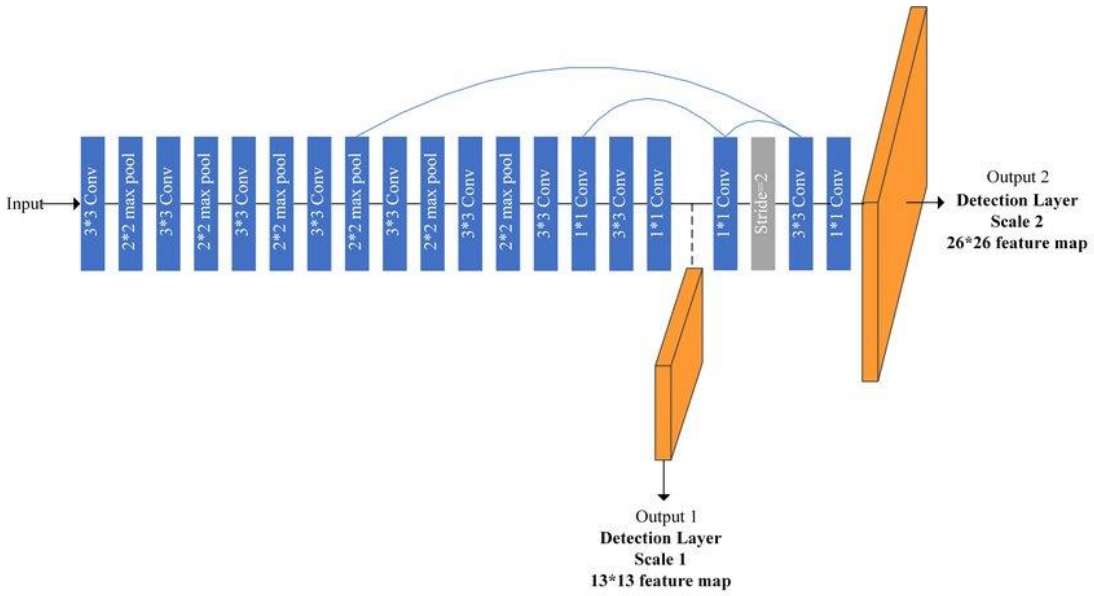


Ilustración 9: Arquitectura de Tiny YOLO V3 [28]

Tal como se puede observar en la figura anterior, el algoritmo recibe una imagen (input) y lo hace pasar por una serie de capas. La imagen está formada por miles de píxeles y, antes de introducirla en la red neuronal, la propia red se encarga de reescalar la imagen, para obtener una matriz inicial 416x416. Las principales capas son filtros CNN (Convolutional Neural Networks). Estos filtros son matrices 3x3 que se van multiplicando por la matriz de imagen de entrada. Puesto que, a medida que se va avanzando en la red, se va multiplicando por otras capas con sus respectivas matrices, la resolución de la imagen se va haciendo más pequeña hasta llegar a las dos salidas (Output1 y Output2 observadas en la imagen) de esta arquitectura. El tamaño de la salida 1 será 13x13 y la de la salida 2 26x26. Cabe destacar que en la versión de YOLO v3 más completa se obtienen tres salidas.

Tal como se observa en la imagen comentada anteriormente, la red neuronal está también formada por capas de max pooling [9]. Estas capas se encargan de eliminar los valores menores resultantes de la multiplicación de matrices. Almacenan aquellos valores más importantes consiguiendo de esta forma reducir aún más el tamaño de las matrices. Tal como se puede observar en la Ilustración 9: Arquitectura de Tiny YOLO V3, esto se realiza varias veces a lo largo de la red completa.

Hay que añadir que el algoritmo proporciona dos salidas con tamaños distintos para mejorar la precisión del sistema. La salida con mayor número de celdas ofrece mejores resultados para la detección de pequeños objetos debido a que se analizan un mayor número de celdas y la salida 13x13 da mejores resultados para objetos de tamaños mayores.

Cabe destacar que toda esta información se puede observar en el archivo de configuración de la red neuronal de YOLO (tyolov3_tiny.cfg). En la ilustración 11 se puede observar el archivo abierto con PyCharm con parte de los parámetros de configuración. De la imagen, cabe destacar que se establece el tamaño de la imagen recibida con ancho y alto 416x416, para las capas CNN se definen los tamaños 3x3 y que las capas de max pooling son de tamaños 2x2.

Una vez vista de forma simplificada la arquitectura de Tiny YOLO v3, se va a comentar qué valores se obtienen en las salidas puesto que son los que se emplean en la programación del código.

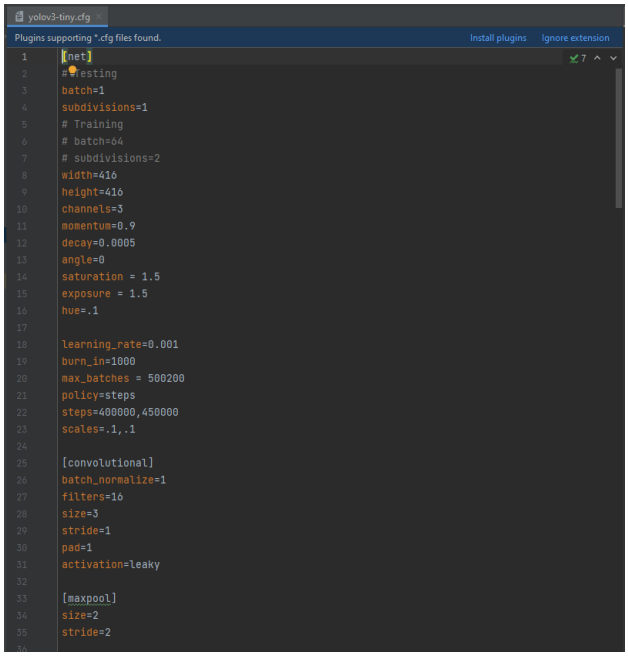


Ilustración 10: Código de configuración de la red neuronal YOLO

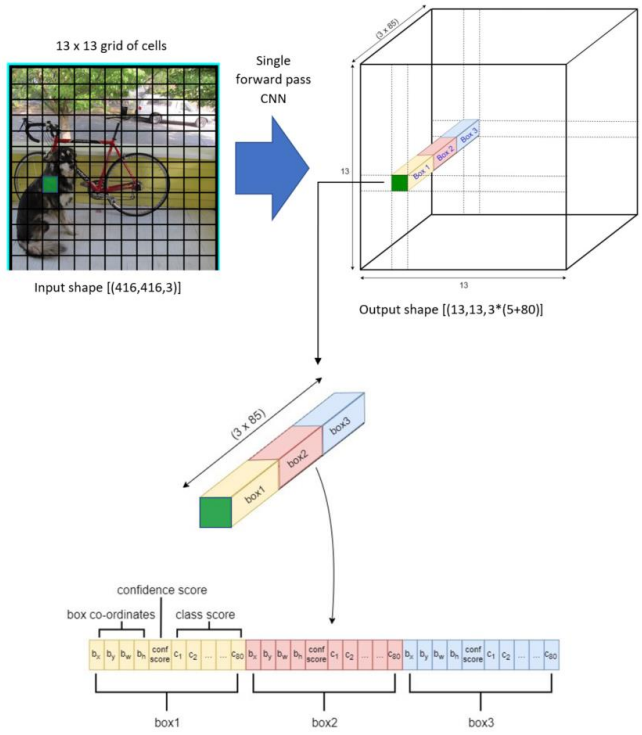


Ilustración 11: Representación de lo que se obtiene a la salida [7]

Como se ha comentado anteriormente, YOLO se encarga de dividir la imagen en celdas, de realizar una predicción de la clase (tipo de objeto) para cada celda e indicar las posiciones de la caja que va a rodear ese objeto. Cabe destacar que las cajas son rectángulos que rodean el objeto detectado por el algoritmo. Para cada celda, el algoritmo proporciona tres bounding boxes, predicciones con la información que se observa en la ilustración 12:

-**Coordenadas de la caja que rodea el objeto (box coordinates)**. Proporciona la posición en "x" e "y" del centro de masas del objeto detectado, además del ancho y altura de la caja que va a rodearlo. Cabe destacar que cada celda tiene unos ejes de coordenadas como los mostrados en la ilustración siguiente.

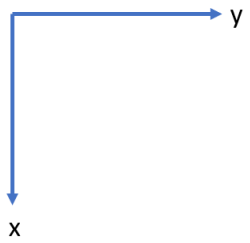


Ilustración 12: Eje de coordenadas de YOLO

-**Confianza en que haya un objeto (confidence score)**. Da un 1 en caso de haber un objeto de cualquier clase y un 0 en caso de que no.

-**Confianza en que sea el objeto de una clase (class score)**. YOLO ofrece una base de datos entrenada de 80 clases y en la ilustración 14 se puede observar parte de esa base. En la columna correspondiente se muestra la confianza que tiene el algoritmo en que sea de un determinado objeto u otro. Los valores oscilan entre 0 y 1 siendo 1 para la máxima confianza.

En total, existen 4 valores de posición, uno de confianza junto con otros 80, por lo que se tienen 85 valores por cada Bounding Box. Al tener 3 bounding boxes por celda, se obtienen un total de 255 valores por cada celda de salida.

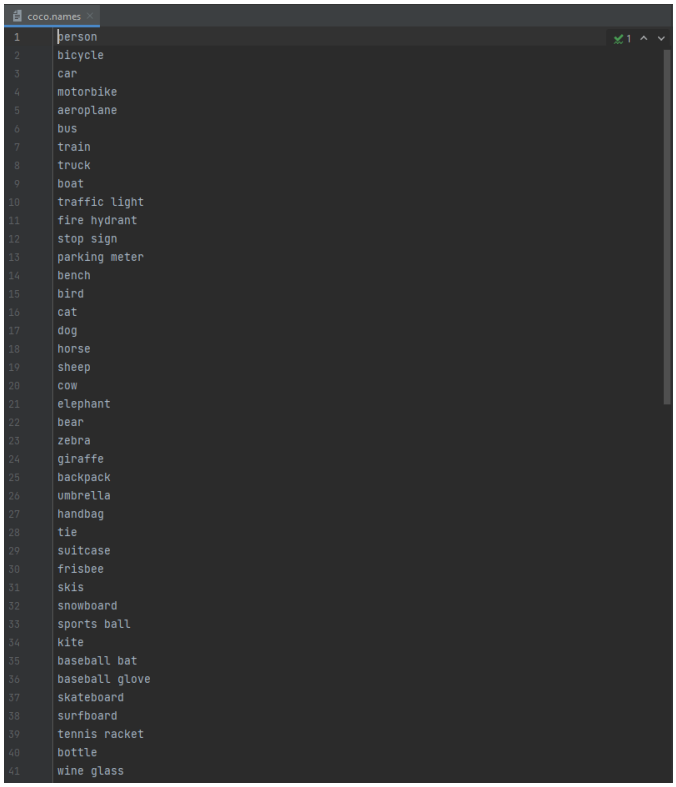


Ilustración 13: Clases de la base de datos de YOLO

Para tener una mejor idea de los valores que se obtienen, en la siguiente imagen se muestra el resultado obtenido por el algoritmo YOLO ejecutando el código. Se obtienen dos matrices correspondientes a las dos salidas de la arquitectura empleada en este caso. En cada fila se encuentran los tres elementos comentados anteriormente.

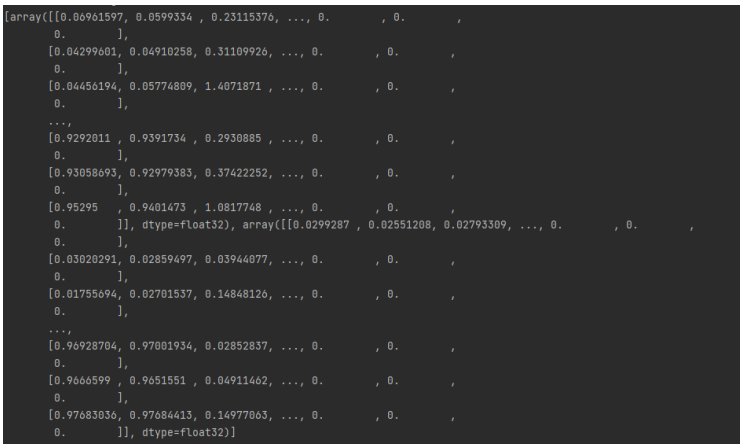


Ilustración 14: Matrices de salida de Outpu1 y Output2

Puesto que cuesta distinguir las filas y columnas de cada una de las matrices se han resumido los resultados obtenidos en la Tabla 1.

	0	1	2	3	4	5	6	7	…	85
	Posición				Confianza	Confianza para cada clase				
Número caja	X	Y	W	H	Confianza	Person	Bicycle	Car	…	toothbrush
0	0.1	0.9	0.37	0.58	0.93	0	0	0.93	0	0
…										
299										

Tabla 1: Ejemplo de una matriz de salida

Para cada fila se obtiene en las 4 primeras columnas la posición del centro de masas en "x" e "y", además del ancho y del alto de la caja que va a rodear el objeto. En la quinta columna, se obtiene 0.93, que es la confianza de que hay un objeto de cualquier clase. El resto de las columnas corresponden a la confianza de que el objeto detectado sea de una clase u otra. En este caso, todas las clases tienen confianza 0 de que sea de ese tipo, pero la clase car tiene una de 0.93.

En el código se tiene como objetivo coger aquellos valores de mayor confianza, ver de qué clase son y dibujar un recuadro alrededor de él conociendo la posición del objeto de interés. Esto se explicará de forma más detallada en la Definición del Trabajo.

2.1.9 FACE RECOGNITION

Es una librería creada por Adam Geitgey [11], y se trata de una de las más sencillas de implementar para la detección facial. Además de presentar una gran precisión, tiene un buen rendimiento y se puede ejecutar en la Raspberry Pi a unos fotogramas por segundo bastante razonables. Para conseguir esto último, se han realizado una serie de cambios en el código que se comentarán en la sección de Desarrollo.

Para su funcionamiento hace uso de Dlib [12], librería que se comentará en la sección de Librerías de Python. Contiene una gran base datos de imágenes que se han empleado para el reconocimiento facial. Además, cuenta con los algoritmos necesarios para realizar todo el procesamiento de imagen. En esta librería, se van a comentar los distintos algoritmos necesarios para conseguir reconocer la cara de una persona puesto que no es tarea fácil reconocer una cara por la cantidad de retos que supone.

En primer lugar, se emplea el algoritmo HOG, el cual se explica en el siguiente apartado, para encontrar las caras de la imagen. Una vez detectada la cara, es necesario obtener una imagen de la cara centrada y mirando en perpendicular a la imagen. Para conseguir corregir la posición de la cara, se usa el algoritmo Face Landmark Estimation. Una vez que se reconocen los puntos más importantes de la cara, se hacen mediciones de esta para poder distinguirlas de otras. Para ello, se emplea una red neuronal que toma las medidas que más interesan y las convierte en vectores. Por último, se compara con las caras recogidas en la base de datos y se usa el algoritmo VSM Classifier.

2.1.9.1 Histogram of Oriented Gradients algorithm

Tal como se explica en 0, el funcionamiento de este algoritmo es relativamente sencillo y se encarga de realizar las siguientes tareas:

- Preprocesar la imagen haciéndola más pequeña y convierte la imagen en blanco y negro.
- Calcular el gradiente de luminosidad para cada píxel.
- Dividir la imagen en zonas de varios píxeles para obtener gradientes más interesantes.

Tal como se comentó anteriormente, se procesa la imagen para reducir su tamaño y de esta forma obtener un mejor rendimiento al realizar los cálculos.

Una vez que se tiene una imagen más comprimida se pasa al cálculo de los gradientes [2] donde se calculan los gradientes horizontales y verticales mediante el uso de filtros. En la ilustración 17 se observan en las imágenes de la izquierda y del centro los valores

absolutos del gradiente en x y del gradiente en y respectivamente, consiguiendo en la imagen de la derecha la magnitud del gradiente.

Ilustración 15: Gradientes calculados [2]

Una vez obtenidos las magnitudes de los gradientes en cada píxel, se divide la imagen original en celdas y se calcula el histograma de gradientes.

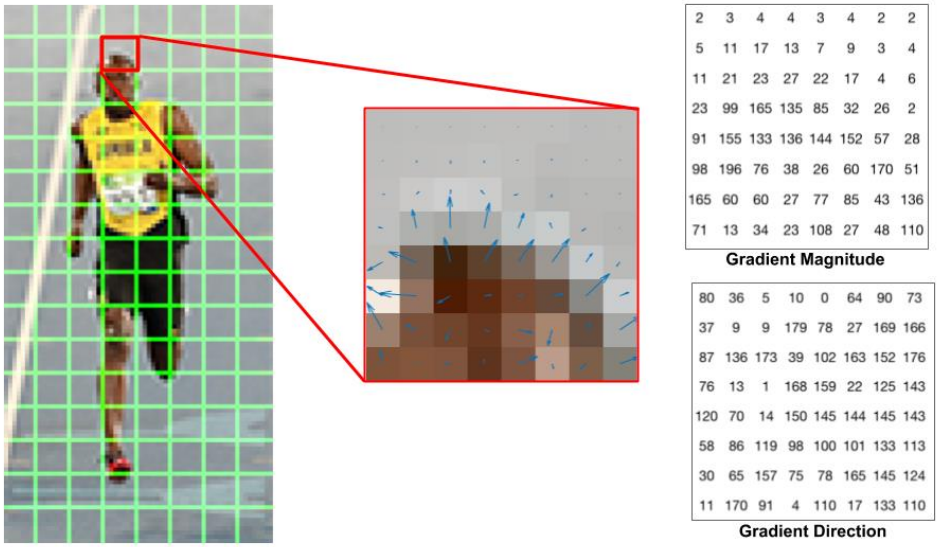


Ilustración 16: División por celdas y cálculo del histograma de gradientes [2]

Ilustración 17: Resultado de la aplicación del algoritmo HOG [25]

2.1.9.2 Face Landmark Estimation algorithm

Según [3], los puntos de referencia faciales se usan para identificar y categorizar partes importantes de la cara. El objetivo de este algoritmo es obtener puntos de referencia en la cara para posteriormente realizar medidas entre puntos y así categorizar la cara. Además, en [3], se menciona el uso del detector de puntos de referencia faciales de Dlib, que es el que se usa también en la librería de face recognition comentada anteriormente.

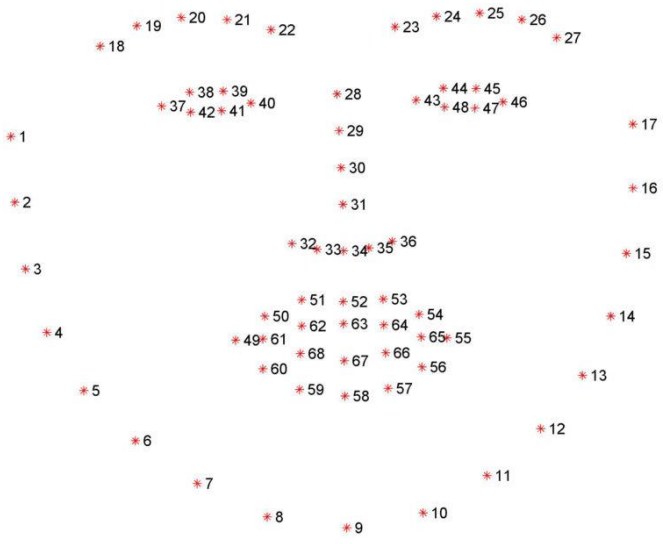


Ilustración 18: Facial landmarks [3]

Este algoritmo se encarga de obtener 68 puntos (landmarks) de las caras que se han detectado con los algoritmos anteriores. Luego se introducen estos puntos en una red neuronal y se obtienen los puntos más interesantes de la cara.

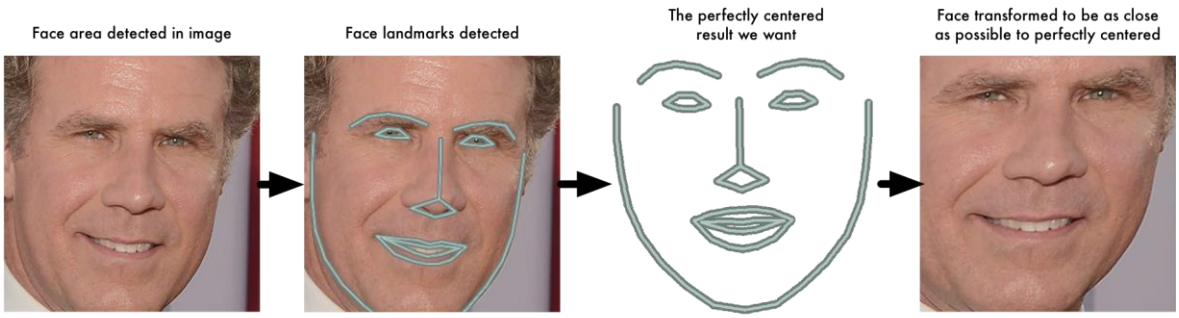


Ilustración 19: Resumen de lo realizado por el algoritmo [1]

2.1.10 ALTAIR SMARTWORKS

Se trata de una plataforma creada por la empresa Altair que tiene como objetivo mejorar la eficiencia operacional de los productos conectados, además de ofrecer una serie de utilidades que permiten al usuario final monitorizar y analizar el producto conectado.

En este proyecto se emplea para reducir la carga de trabajo del sistema de seguridad. Para ello, la plataforma recibe datos del sistema y mediante funciones que se ejecutan en los servidores de Altair, se consigue reducir la carga computacional. Además, se utiliza para monitorizar el estado del sistema mediante la recepción de datos enviados a través del protocolo MQTT. Estos datos son utilizados para generar gráficos que facilitan la monitorización del sistema de seguridad.

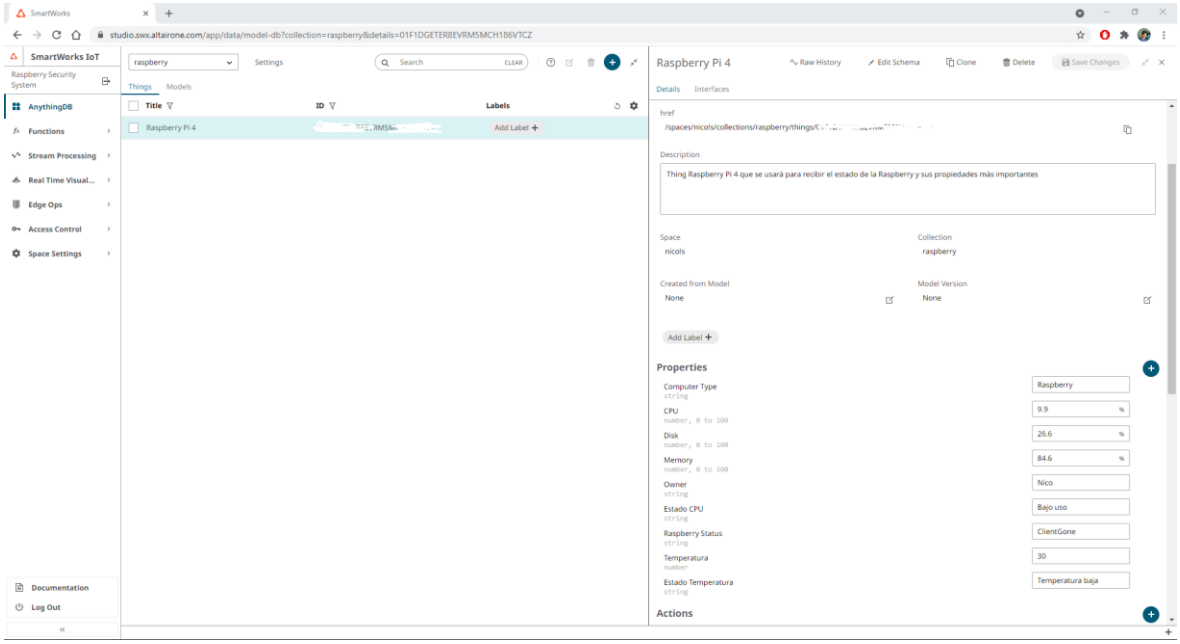


Ilustración 20: Plataforma SmartWorks

2.1.11 SOLIDWORKS

Es un programa CAD que permite diseñar piezas en tres dimensiones y se emplea para el diseño de la carcasa que va a integrar todos los componentes del sistema de seguridad. Además de permitir diseñar piezas, se pueden realizar simulaciones de todo tipo. Por otro lado, presenta la ventaja añadida de permitir diseñar piezas por separado para posteriormente ensamblarlas en una única pieza. Esto resulta especialmente útil para, en primer lugar, diseñar las piezas más pequeñas como la que va a sujetar la cámara del sistema e ir ajustándola hasta conseguir que se adapte perfectamente. Posteriormente se integran todos estos pequeños componentes en el diseño y se imprimen con una impresora 3D.

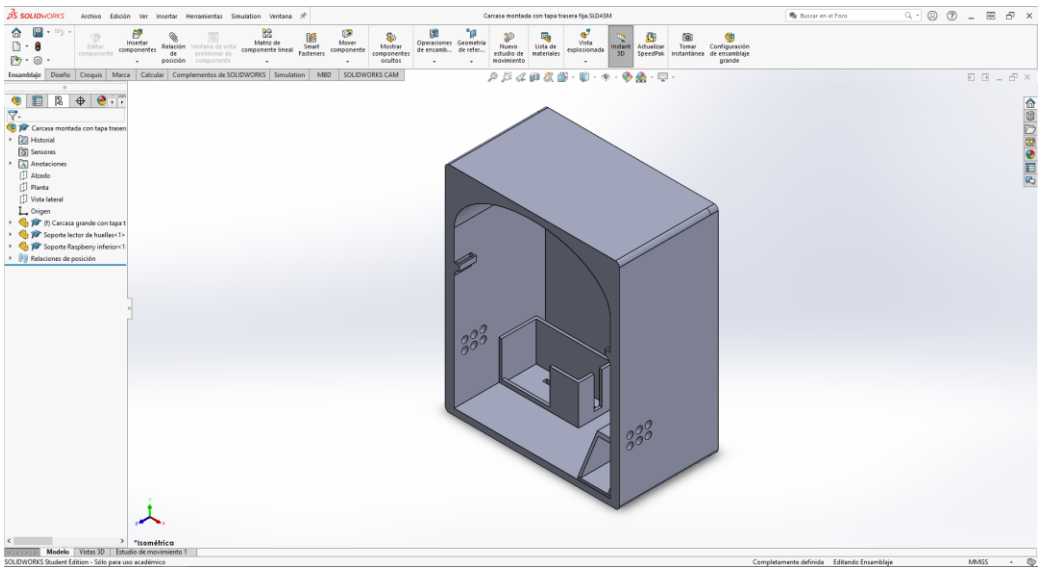


Ilustración 21: Interfaz de SolidWorks

2.1.12 ULTIMAKER CURA

Es un programa informático que permite ajustar los parámetros necesarios para la impresión de la carcasa en 3D. Es de código libre y actualmente solo está disponible en Windows. En él se configuran parámetros como la altura de cada capa impresa en 3D, la velocidad de impresión, la forma del interior de las paredes de la pieza, etc.

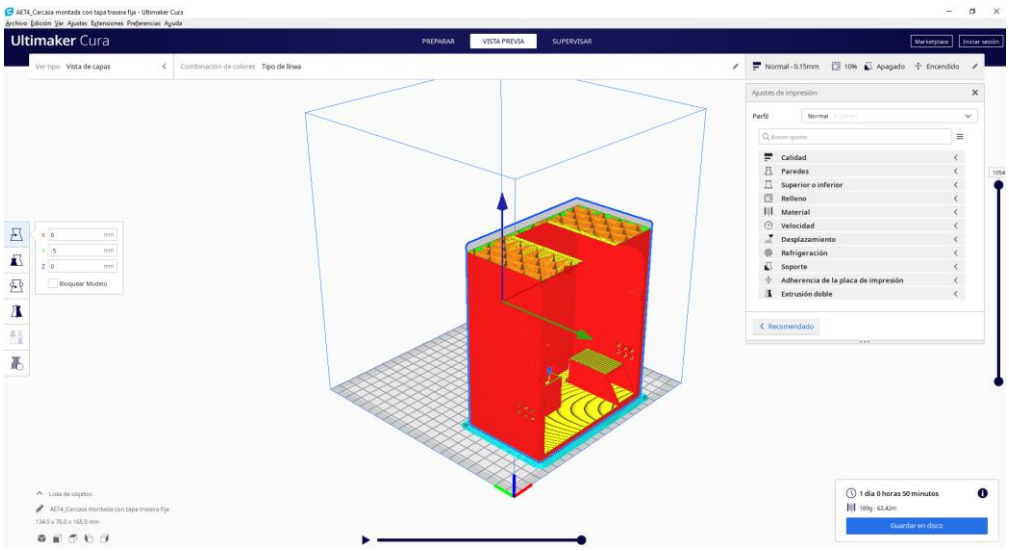


Ilustración 22: Interfaz de Ultimaker Cura

2.1.13 LIBRERÍAS DE PYTHON

Para la implementación de todos los componentes necesarios de este proyecto se han utilizado las siguientes librerías:

2.1.13.1 OpenCV

Open Computer Vision, es una librería de visión artificial de código abierto que cuenta con una gran cantidad de funciones relacionadas con el reconocimiento de vídeo y fotografía. Es la librería más popular en este ámbito ya que ofrece múltiples funciones. Está documentada y explicada, lo cual facilita la implementación de manera relativamente sencilla, es multiplataforma y es muy eficiente. En la sección de Definición del Trabajo, se comentará cómo se ha empleado en este proyecto, ya que supone uno de los ejes principales para el uso de los algoritmos comentados.

Ilustración 23 OpenCV

2.1.13.2 Flask

Es una librería que permite crear un framework de Flask, es decir, permite crear una aplicación web a la cual se puede acceder de forma sencilla desde cualquier ordenador conectado a Internet. Con él se puede ver en tiempo real la imagen de la cámara de la Raspberry. Se podrían haber usado otras alternativas, pero suponían una curva de aprendizaje mayor y el proyecto no tenía como objetivo el diseño web.

Ilustración 24 Flask

2.1.13.3 Numpy

Se trata de una librería que permite al usuario crear vectores y matrices multidimensionales de gran tamaño. Estas matrices son utilizadas para la ejecución de los algoritmos ya que es donde se almacenan cada uno de los datos leídos en la imagen analizada.

2.1.13.4 DLIB

Es una librería de código abierto programada en C++. Cuenta con todo tipo de algoritmos (Machine Learning y algoritmos numéricos), además de múltiples funciones. Para este proyecto se van a usar los algoritmos de Deep Learning que permiten ejecutar el algoritmo YOLO y, por otro lado, el algoritmo de detección facial.

2.1.13.5 Pyfingerprint

Esta librería permite implementar el sensor de huellas dactilares. Está basada en otra librería de una empresa de hardware de código abierto llamada Adafruit Industries. Esta no es compatible con el modelo del sensor de huellas empleado para el proyecto y, es por ello, que se ha recurrido a la de Pyfingerprint para poder usar el lector de huellas de la marca ZFM.

2.1.13.6 Threading

Permite generar distintos hilos en el programa con los cuales se pueden realizar diversas tareas de forma simultánea. De esta forma se evita paralizar el programa cuando está ejecutando otra tarea. Esta librería es especialmente importante porque permite ejecutar el servicio Flask en segundo plano sin paralizar el funcionamiento de los algoritmos. Por otro lado, también se emplea para los distintos temporizadores utilizados en el programa.

2.1.13.7 Smtplib

SMTPLIB permite el envío de correos electrónicos a cualquier tipo de cliente con compatibilidad SMTP o ESMPTP tal como es el caso de Gmail. Esta librería se emplea para los avisos del acceso de una persona al domicilio y para el envío de las imágenes adjuntas.

2.1.13.8 Paho-mqtt

Permite la conexión con un bróker MQTT para enviar y recibir mensajes. Esta librería se usa para conectarse con la plataforma SmartWorks a la cual se envían diferentes datos del estado de la Raspberry.

MQTT es un protocolo de red que actúa como intermediario entre el sistema de seguridad y SmartWorks. El sistema de seguridad "publica" la información, el bróker MQTT filtra la información y la envía a la plataforma.

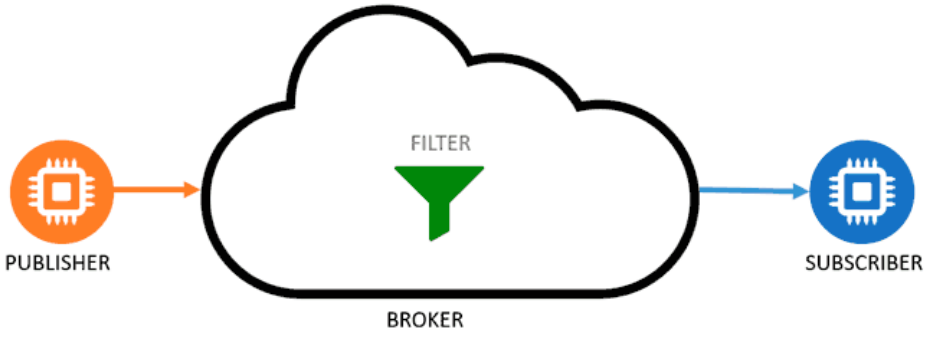


Ilustración 25: Función del bróker MQTT [26]

2.1.13.9 Gtts

Es una librería de Google y significa Google Text to Speech. Permite convertir texto de las líneas de código en un archivo de audio para posteriormente leerlo y reproducirlo con otras librerías.

2.1.13.10 Pygame

Se trata de una librería de código abierto que permite el desarrollo de juegos en Python. Sin embargo, para este proyecto únicamente se usa la función que permite reproducir archivos

de audio. Estos archivos se escuchan a través del altavoz que lleva incorporado la Raspberry Pi.

2.1.13.11 Time

Permite acceder a la fecha en la que se encuentra el sistema de seguridad.

2.1.13.12 OS

Permite interactuar con el sistema operativo en el cual se ejecuta el código. Permite acceder a archivos almacenados en el sistema operativo.

2.1.13.13 Json

Permite convertir datos a formato json (JavaScript Object Notation) para posteriormente poder enviar datos a la plataforma SmartWorks.

2.1.13.14 Gpiozero

Es una librería que se emplea para obtener los datos de temperatura del procesador de la Raspberry Pi.

2.2 HARDWARE

2.2.1 RASPBERRY PI 4B

La Raspberry Pi 4 es una placa de tamaño reducido fabricada por la fundación Raspberry Pi. Se podría considerar como un pequeño ordenador por presentar todos los componentes característicos de uno: procesador, memoria RAM, almacenamiento por microSD, conexión Wifi y Bluetooth, puerto ETHERNET, salidas microHDMI para poder conectar pantallas además de contar con puertos USB y un puerto Jack.

Se ha optado por este modelo por tratarse de la última versión disponible que cuenta con un procesador suficientemente potente para ejecutar todo el sistema de manera óptima.

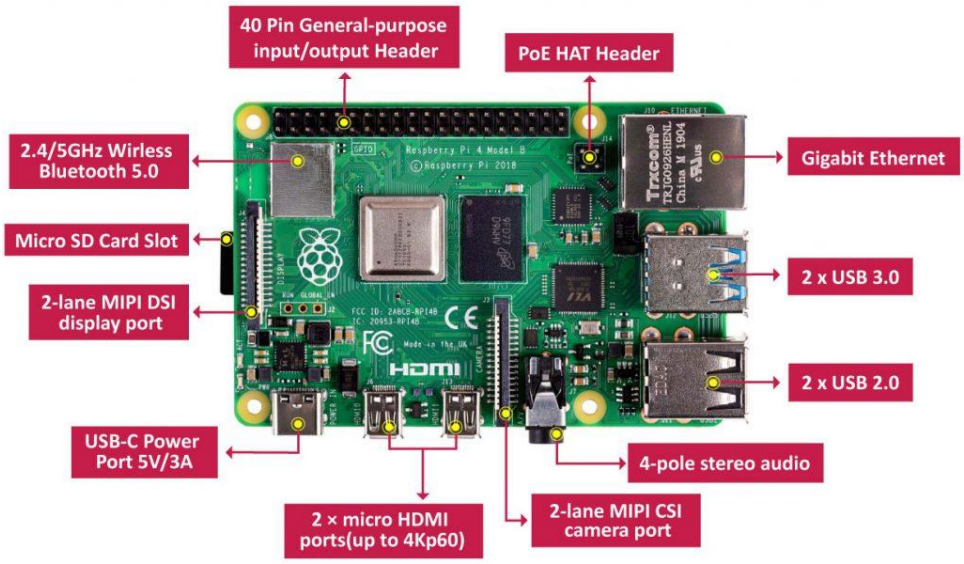


Ilustración 26: Raspberry Pi 4B [29]

La Raspberry Pi es el cerebro de todo el sistema de seguridad y se encarga de ejecutar el programa principal, enviar todos los datos necesarios a través de la conexión Wifi y gestiona todos los periféricos que componen el sistema.

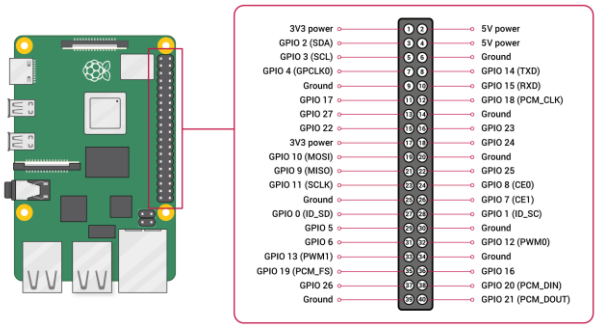


Ilustración 27: Pines Raspberry Pi 4B [30]

2.2.2 CÁMARA CON VISIÓN NOCTURNA

Se trata de uno de los componentes principales del sistema, ya que es el empleado para realizar la identificación de personas y objetos. Puesto que el sistema está en funcionamiento tanto de día como de noche cuenta con visión nocturna.

La cámara se conecta al puerto de cámaras que integra la Raspberry Pi y no necesita alimentación externa.

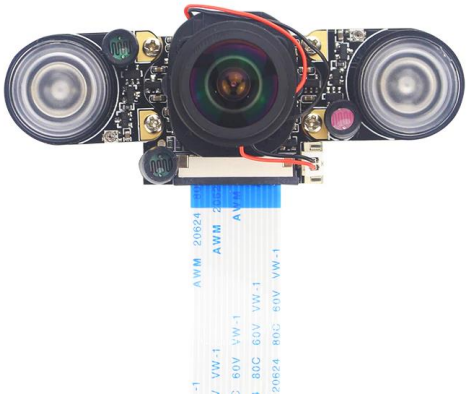


Ilustración 28: Cámara con visión nocturna [31]

2.2.3 MÓDULO TTL

Se trata de una pequeña placa que se conecta en uno de los puertos USB de la Raspberry y ofrece los pines necesarios para conectar el lector de huellas.

Ilustración 29: Módulo TTL CP2102 [32]

Para el conexionado del lector de huellas al módulo se ha empleado el siguiente esquema:

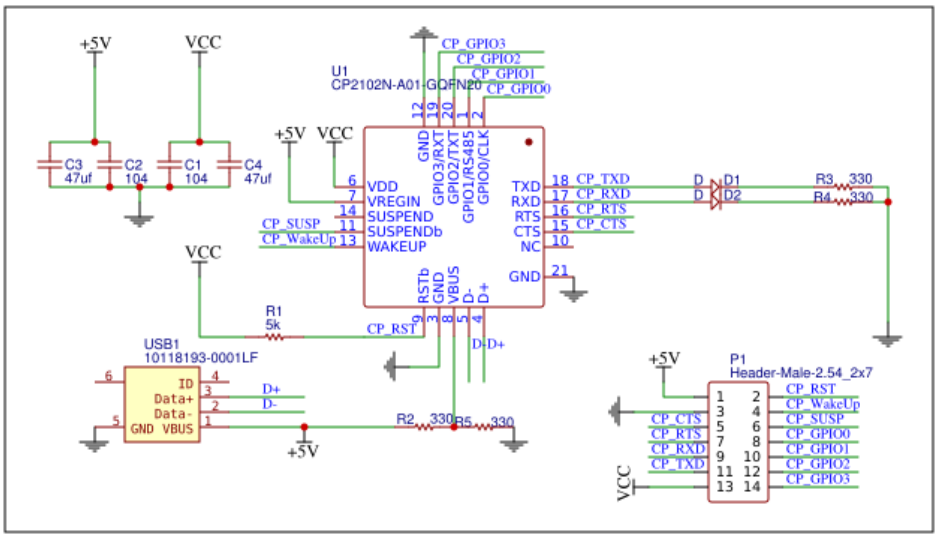


Ilustración 30: Esquema conexionado módulo TTL [33]

2.2.4 LECTOR DE HUELLAS

Como se indicó en el apartado de software se trata de un sensor de huellas óptico de la marca ZFM. Se emplea para el reconocimiento de huellas dactilares y va conectado al módulo TTL sin necesidad de alimentación externa.

Ilustración 31: Lector de huellas ZFM-70 [35]

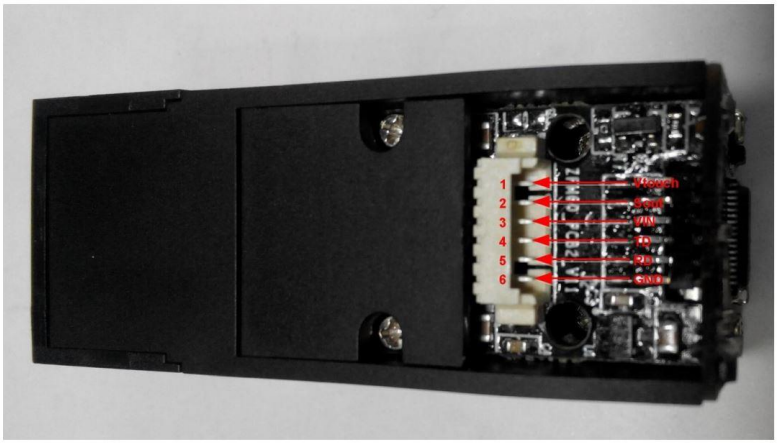


Ilustración 32: Puertos del sensor de huellas [34]

2.2.5 SISTEMA DE REFRIGERACIÓN

Uno de los inconvenientes de la Raspberry es que sufre sobrecalentamientos debido al cálculo requerido para ejecutar todos los algoritmos del sistema. Otro factor a tener en cuenta es el tiempo que se está ejecutando y puesto que va a estar funcionado las 24 horas del día va a resultar necesario dotarlo de un buen sistema de refrigeración. Inicialmente se probaron con unos simples disipadores como los de la figura que se muestra a continuación, pero sufría sobrecalentamientos.

Ilustración 33: Disipadores empleados al comienzo del proyecto [38]

Otro aspecto que había que tener en cuenta es que la Raspberry va a estar encerrada en una caja y no habría mucho flujo de aire y las pruebas se realizaron sin estar metido en una caja. Viendo que era necesario desalojar el calor de la zona, al final se ha optado por un sistema de disipación con dos ventiladores tal como se muestra en la ilustración 32. El sistema contaba con dos pines: uno de tierra y otro de tensión que se ha conectado en el pin de 5V de la Raspberry para que funcionaran a una mayor velocidad

Ilustración 34: Sistema de refrigeración final [36]

2.2.6 ALTAVOZ

Conectado al puerto Jack de la Raspberry, se ha conectado un altavoz barato de Amazon. Puesto que el altavoz necesita alimentación externa se ha aprovechado uno de los puertos USB para alimentarlo ofreciendo la potencia suficiente para su correcto funcionamiento.

Cabe destacar que se podría emplear cualquier otro tipo de altavoz que cuente con puerto Jack y que sea alimentado por USB.

Ilustración 35: Altavoz empleado [37]

2.2.7 CARCASA

Se trata de una carcasa impresa en 3D que integra a todos los componentes que conforman el sistema de seguridad. Está diseñada con el programa CAD SolidWorks y tiene como objetivo ocupar el menor espacio posible permitiendo una correcta ventilación. Además, está diseñada de tal forma que permite extraer cualquier componente de forma sencilla en caso de necesitar reemplazar alguno. Cabe destacar que la imagen que se muestra a continuación tiene la parte delante abierta para poder visualizar el interior, pero en la realidad estaría completamente cerrado.

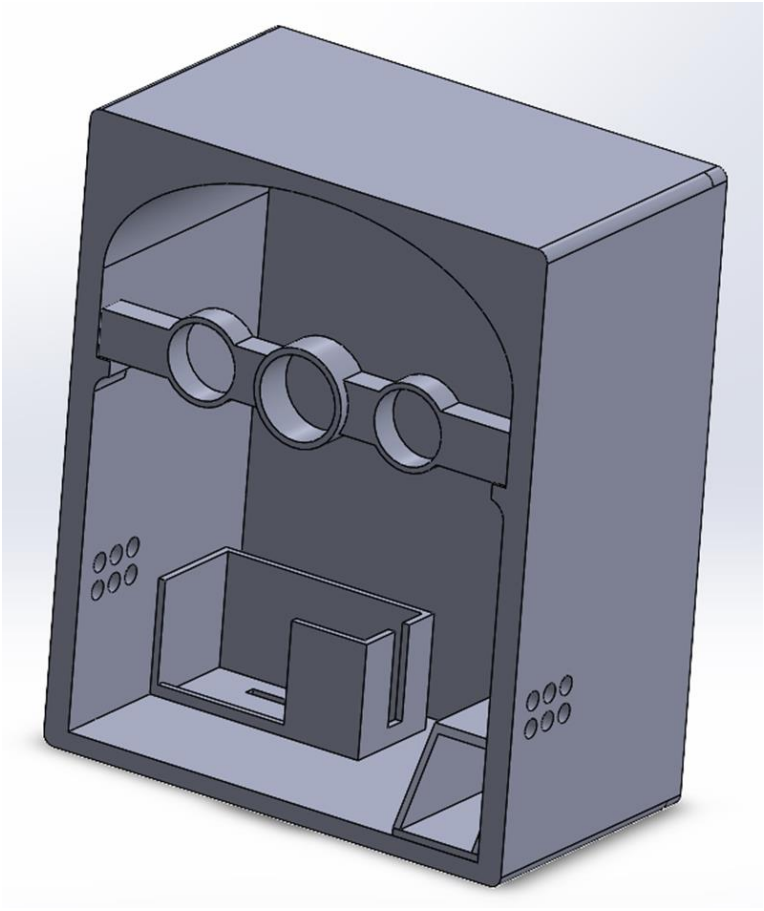


Ilustración 36: Carcasa diseñada

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 SISTEMAS DE SEGURIDAD EN EL MERCADO

En el mercado existen diversas alternativas a todos los problemas planteados en este proyecto. Esto se debe al gran desarrollo de la tecnología necesaria para este tipo de soluciones lleva años lo suficientemente desarrollada para ofrecer alternativas viables.

3.1.1 CÁMARAS DE VIGILANCIA BÁSICAS

Uno de los productos que se ha puesto muy de moda por su reducido coste y fácil instalación son las cámaras IP. Éstas se suelen colocar en interiores tal como los que se pueden encontrar en [14]. Permiten el acceso de forma remota, fuera de la vivienda y sin necesidad de estar conectado a la red local. En las versiones más caras llegan a dar avisos de movimiento a través de aplicaciones móviles creadas por el propio fabricante. Se trata de productos que oscilan entre los 30€ y los 150€ siendo los más caros los que cuentan con visión nocturna y capacidad de rotación y cambio de posición de la cámara.

Ilustración 37: Cámara IP básica [14]

3.1.2 CÁMARAS DE VIGILANCIA MÁS AVANZADAS

Un peldaño más arriba se pueden encontrar cámaras que realizan las mismas funciones, pero que también usan Deep Learning para el reconocimiento de rostros. Adicionalmente, cuentan con visión nocturna para poder usarla en cualquier situación de luz. Se pueden encontrar cámaras como las de la empresa Hikvision [15] con precios que rondan los 400€ pero que no cuentan con retransmisión en tiempo real.

Ilustración 38: Cámara con detección facial [15]

Los modelos superiores de la misma marca o de la competencia, ofrecen también retransmisión en tiempo real de la imagen de las cámaras, pero el precio es más elevado. Axis [18] y Marshall Electronics [19] son líderes en este segmento y ofrecen cámaras como las comentadas anteriormente hasta cámaras protegidas contra explosiones como las que se pueden encontrar en [18].

En este segmento también se pueden incluir las cámaras externas con capacidad de detección de movimiento y de visión nocturna. Se colocan alrededor del jardín de una casa y disponen de conexión remota. Todas estas cámaras se conectan a una central situada en la propia casa y es la encargada de ofrecer al usuario un sistema de avisos y de retransmisión en tiempo real. Un ejemplo sería el de la marca Reolink [16] y que se puede encontrar en tiendas por 400€.

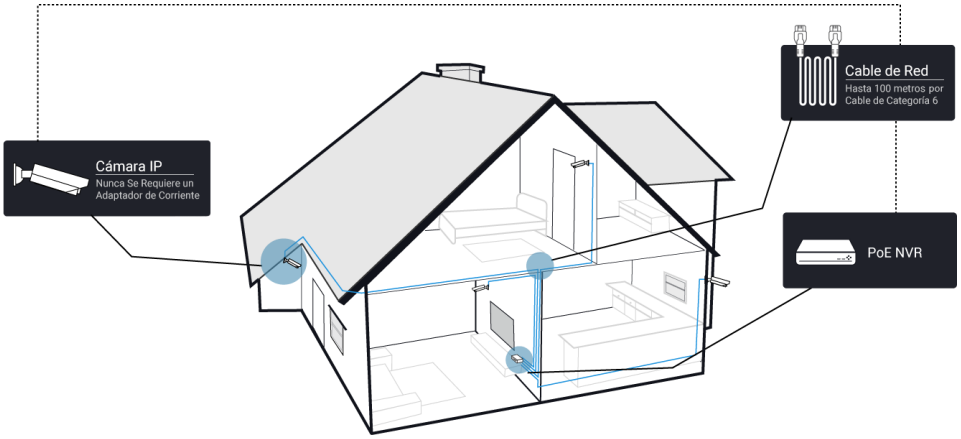


Ilustración 39: Cámaras de vigilancia centralizadas [16]

3.1.3 SISTEMAS DE SEGURIDAD COMPLETOS

Uno de los inconvenientes que presentan las cámaras de reconocimiento facial es que se pueden manipular usando imágenes de personas que viven en la casa para mostrarla a la cámara entre otras acciones para engañar al sistema. Debido a esto, surge la necesidad de añadir una capa adicional de seguridad para permitir el acceso a un domicilio. Una de las soluciones más eficaces es usar un lector de huellas dactilares. Otra de las soluciones que se emplean es el reconocimiento por voz o el uso de una tarjeta.

Hikvision ofrece este tipo de productos [17], pero con un coste que supera los 1000€, por lo que se limita la compra de este tipo de productos a un muy reducido número de personas. Por esta razón, normalmente se orienta al sector empresarial.

Ilustración 40: Sistema de seguridad con detección de rostro y lector de huellas [17]

3.2 ALGORITMOS DE DETECCIÓN

Con el avance de la tecnología de los últimos años y la mejora de la capacidad computacional de los equipos, se han podido desarrollar algoritmos con capacidad de detectar objetos y caras que hace años era impensable. Por ello, han ido surgiendo algoritmos que emplean ideas distintas a la hora de diseñar la arquitectura, siendo unos más eficaces a la hora de detectar y permitiendo realizar detecciones en tiempo real. Puesto que la Raspberry Pi dispone de una capacidad de cálculo limitada, se ha tenido que buscar unos algoritmos que tengan un equilibrio entre precisión y velocidad de detección. En esta sección se va a comentar aquellos más representativos.

En [20] se realiza una comparativa de rendimiento entre los algoritmos más conocidos en el ámbito de la detección de objetos, como son Faster R-CNN, YOLO v3, RetinaNet y R-FCN. Para realizar este estudio, se ha basado en papers individuales de cada algoritmo y concluye que los algoritmos basados en división de regiones (R-CNN) muestran una leve mejora de precisión frente a la desventaja de presentar menor velocidad. En cambio, aquellos algoritmos basados en una imagen (YOLO) tienen una precisión aceptable para la gran velocidad que presentan a la hora de procesar las imágenes.

Según [22], en la ilustración 39 muestra cuál es la gran ventaja de YOLO con respecto a otros algoritmos y queda perfectamente reflejado en el que se muestra a continuación.

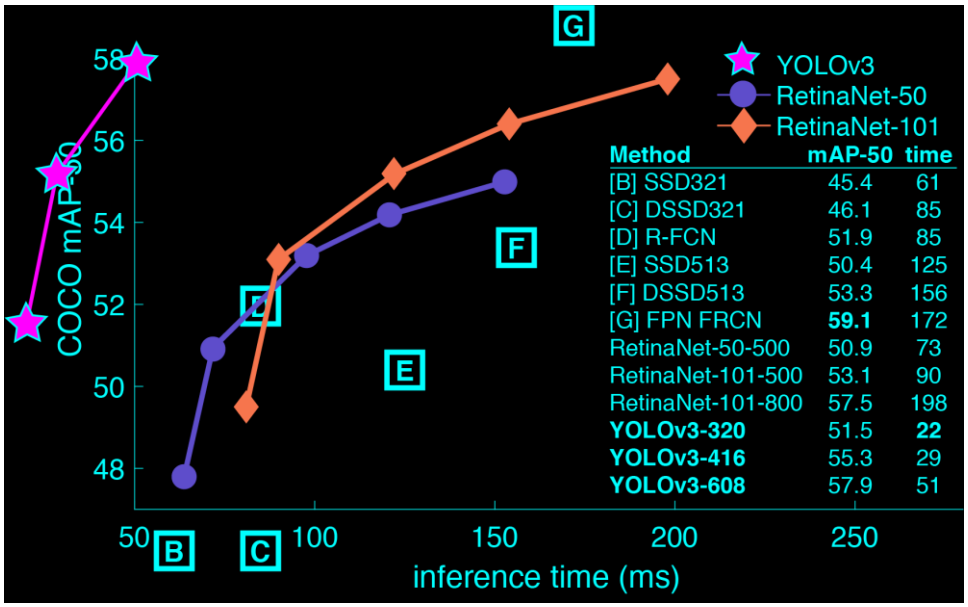


Ilustración 41: Comparativa de rendimiento de diferentes algoritmos [5]

Se observa que el algoritmo YOLO v3 presenta un elevado rendimiento en comparación con otros algoritmos además de no perder precisión a la hora de detectar objetos.

Una variable a tener en cuenta cuando se implementan esto algoritmos es el framework que se va a usar. En [21] se realiza una comparativa entre Caffe, TensorFlow, OpenCV y Caffe2 y se concluye que con OpenCV, usando la red neuronal de SqueezeNet, se obtienen los mejores resultados. Adicionalmente, concluyen que OpenCV es una de las mejores opciones para sistemas con bajo consumo. Esto último, queda reflejado en el gráfico de la ilustración 40 en el cual se muestran los resultados obtenidos en FPS (Fotogramas por segundo). Esto permite cuantificar la velocidad de procesamiento de la imagen en tiempo real. En la imagen aparecen los resultados para cada red distinta para cada framework. Se puede observar que OpenCV obtiene los mejores resultados para dos redes distintas y el segundo mejor resultado para una tercera.

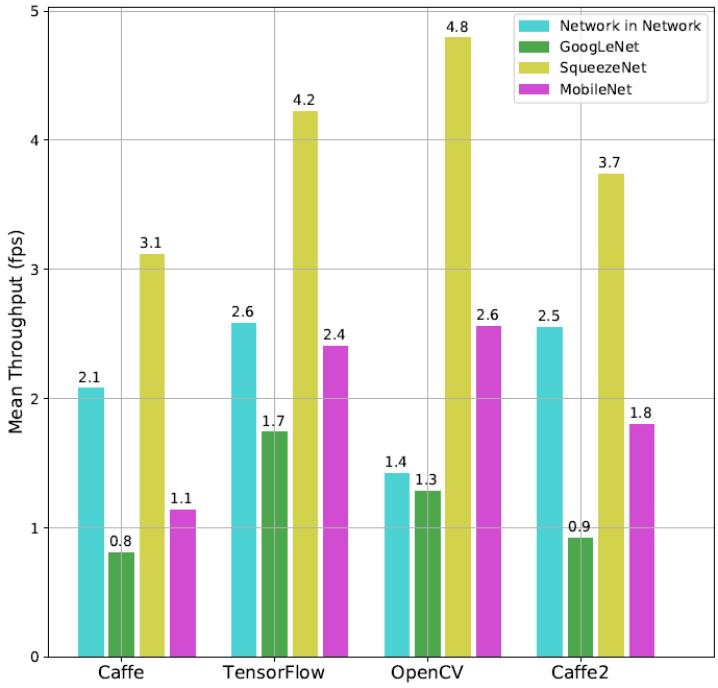


Ilustración 42: Rendimiento obtenido para distintos frameworks[21]

3.2.1 ARQUITECTURAS YOLO

El algoritmo YOLO tiene distintos modelos [5], que se diferencian en la estructura interna en las que se pueden encontrar más o menos capas CNN o se hacen distintos "trucos" para mejorar la velocidad del sistema. La arquitectura de la versión YOLO v3 más completa se encuentra explicada en [23] donde también se realiza una comparativa con los modelos anteriores de YOLO. Se obtienen 3 salidas con imágenes a 3 escalas distintas. Esto último ha sido una respuesta para mejorar la versión anterior a la hora de detectar objetos más pequeños. La salida con mayor número de celdas es capaz de detectar mejor los objetos pequeños mientras que la salida con menor número de celdas es más conveniente a la hora de detectar objetos grandes.

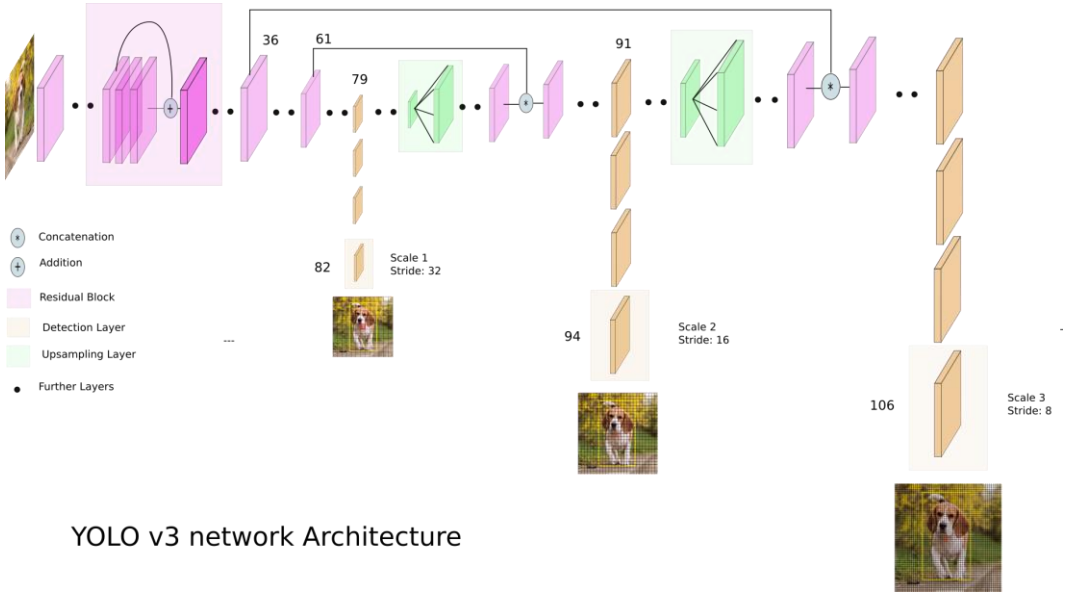


Ilustración 43: Arquitectura YOLO v3 [23]

A pesar de ser un algoritmo con un rendimiento muy elevado, existe una versión más reducida que elimina muchas capas y reduce el número de salidas a dos. La velocidad de detección aumenta sustancialmente, pero a costa de una precisión mucho menor.

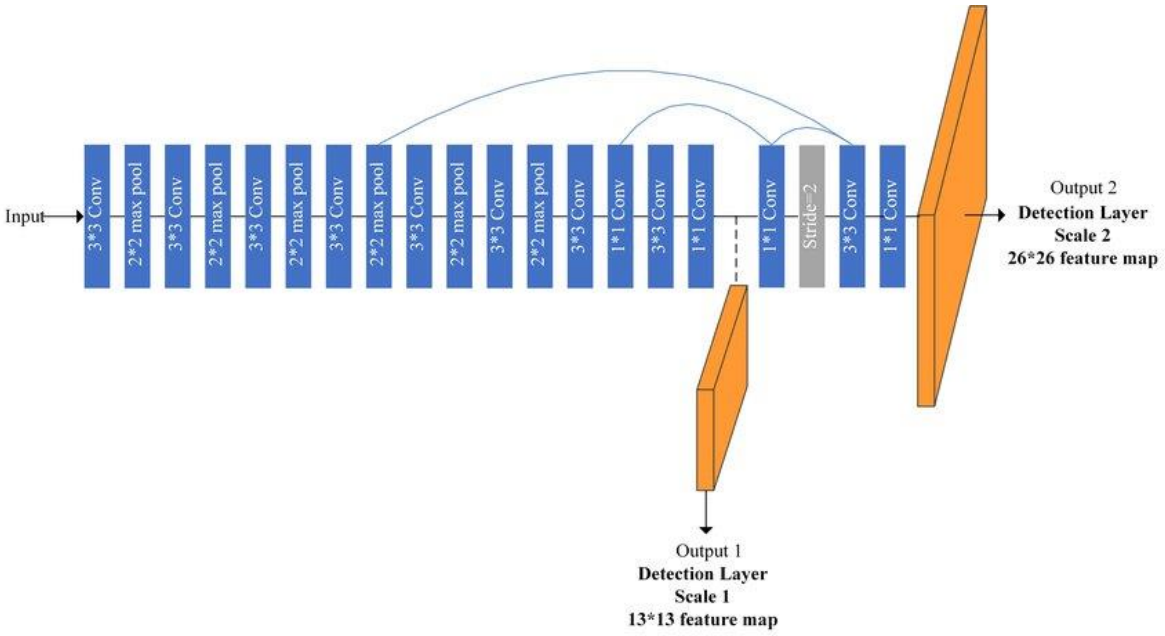


Ilustración 44: Arquitectura Tiny YOLO v3 [28]

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Una vez realizado el análisis de la situación, se puede observar que hoy en día existen soluciones similares a este proyecto pero que se quedan cortos a nivel de seguridad o de funcionalidades además de ser muy costosos. Se ha visto que varias empresas ofrecen soluciones con únicamente sensor de huellas o con solo cámara de seguridad. Además de contar únicamente con una barrera de seguridad, no cuentan con aplicaciones adicionales que se aprovechen de la tecnología que se está empleando. Por otro lado, en los casos en los que el producto cuenta con sensor de huellas y reconocimiento facial, los precios se vuelven desorbitados además de ofrecer pocas funcionalidades adicionales como los que se han implementado en este proyecto. Además, también se ha realizado una búsqueda exhaustiva de los algoritmos más eficientes a nivel computacional y a nivel de detección tanto de objetos como facial para ofrecer aquellos que son mejores en función del trabajo que esté realizando el sistema.

El objetivo de este proyecto es diseñar un producto que ofrezca seguridad gracias al uso de reconocimiento facial, de un lector de huellas junto con un sistema de alertas por correo a un muy reducido coste, ofreciendo de esta manera un producto barato y al alcance de toda persona que desee aumentar la seguridad de su domicilio. Además de contar con estas funcionalidades, también ofrece imagen en tiempo real a través de internet y de una plataforma donde monitorizar todo el estado del sistema de seguridad. Cabe destacar que los componentes que forman el sistema son fácilmente reemplazables y tal como se podrá ver en los siguientes apartados, todos sus componentes son baratos y fáciles de adquirir.

4.2 OBJETIVOS

Tal como se comentó anteriormente, se tiene como objetivo diseñar un sistema completo que ofrezca mucha seguridad y con una serie de funcionalidades que puedan resultar útiles para el usuario a un muy reducido coste.

4.2.1 SISTEMA DE SEGURIDAD COMPLETO

Tal como se ha mencionado anteriormente, el sistema cuenta con una cámara con visión nocturna para no reducir su utilidad únicamente a las horas con luz. Adicionalmente, cuenta con un lector de huellas. El objetivo es usar el lenguaje de programación que más facilite usar la información recibida por la cámara y por el sensor de huellas para ofrecer un sistema que ofrezca una gran seguridad. Además, hay que añadir que también se tiene como objetivo implementar los mejores algoritmos en función de la actividad que esté realizando el sistema. Por ejemplo, en el caso que se necesite detectar objetos ya sea para detectar personas, animales, cartas, etc. se implementa el algoritmo más eficiente a nivel computacional para realizar esta tarea y en los casos en los que se desee detectar la cara se usa otro algoritmo más eficaz para realizar esta función.

4.2.2 FUNCIONALIDADES ADICIONALES

Aprovechando la cámara y la tecnología actual para ofrecer más funcionalidades, también se tiene como objetivo enviar avisos por correo a la hora de detectar una persona durante cierto tiempo además de enviar avisos tras permitir el acceso de una persona en el domicilio.

Por otro lado, también se quiere ofrecer la posibilidad de ver en tiempo real las detecciones que se están realizando a través de una plataforma sencilla de acceder y a la cual se puede acceder desde cualquier sitio.

Además, se quiere ofrecer una plataforma en la cual se pueda monitorizar el estado del sistema de seguridad, ya sea si está detectando objetos, o leyendo la huella dactilar o reconociendo facialmente a una persona, además de recibir los parámetros de la temperatura y uso de la CPU del sistema de seguridad. Cabe destacar que también se tiene como objetivo

implementar en esta plataforma una serie de gráficos del número de accesos, de las temperaturas y de más parámetros.

Por último, se desea implementar un altavoz para facilitar la interacción del usuario final con el sistema de seguridad. Para ello, se va a programar de tal forma que indique al usuario qué debe realizar a la hora de configurar el sistema. Además, en los casos que el usuario desee acceder al domicilio se le va a indicar cómo debe realizarlo.

4.2.3 DISEÑO ÓPTIMO DEL SISTEMA

Por último, se tiene como objetivo incorporar todos los componentes necesarios para el correcto funcionamiento del sistema de la forma más óptima posible. Esto implica diseñar la carcasa de tal forma que los componentes estén situados en la mejor posición posible permitiendo la correcta ventilación del sistema. Además, se quiere facilitar el cambio de componentes si fuera necesario. Dentro de este apartado también se tiene como objetivo encontrar la mejor forma para refrigerar una placa como la Raspberry Pi 4 de la mejor forma posible.

Por otro lado, se desea ofrecer un sistema de seguridad eficiente a nivel energético. Se tiene como objetivo ofrecer un producto que consuma menos energía que los sistemas de seguridad similares existentes en el mercado.

4.3 METODOLOGÍA

Para la realización de este proyecto se ha recurrido especialmente a toda la información existente en internet y se han aplicado conocimientos obtenidos a lo largo de la carrera.

4.3.1 BÚSQUEDA DE UNA PLACA IDÓNEA

Para poder diseñar un sistema de seguridad que sea capaz de reconocer objetos y detectar caras es necesario disponer de un procesador potente. Por otro lado, se debe disponer de una placa que ofrezca múltiples puertos a los que conectar los diferentes periféricos con los que cuenta este sistema de seguridad. Además, se tiene que encontrar una placa que permita la

programación y ejecución de los algoritmos de detección en un lenguaje de programación extendido y disponga de diversas librerías para poder implementar funcionalidades adicionales. Es por ello, que uno de los primeros pasos para la implementación de este sistema es encontrar la placa idónea que cumpla con todos estos requisitos, ya que va a ser la parte central del sistema.

4.3.2 APRENDIZAJE DE PYTHON

Puesto que el lenguaje de programación que más facilidades ofrece para implementar todo este sistema es Python, se ha tenido que aprender a manejarlo. A pesar de tener conocimientos de programación previos, se ha hecho necesario aprender cómo funciona Python para poder implementar de la mejor forma posible el sistema de seguridad. Para tener unas nociones básicas de cómo es el lenguaje, se ha aprendido con una página web que dispone de cursos de varios lenguajes de programación llamada SoloLearn.

Se ha optado por el curso más completo que iba de lo más básico hasta lo más avanzado como podría ser la programación orientada en objetos en Python. A pesar de haberse realizado el curso completo, a lo largo del desarrollo del proyecto, se ha hecho necesario aprender por otros medios ciertos aspectos específicos que no se llegaban a tocar. Por lo general se ha recurrido a la página web Stackoverflow y en algunos casos en los que se requería entender mejor ciertos conceptos, se ha optado por tutoriales de YouTube. También a medida que se iban implementando librerías para ofrecer ciertas funcionalidades al proyecto, se ha recurrido a estas últimas dos páginas mencionadas y a la página oficial de Python en las secciones correspondientes de cada librería.

4.3.3 BÚSQUEDA DE ALGORITMOS DE DETECCIÓN DE OBJETOS

Uno de los aspectos más importantes del sistema de seguridad ha sido encontrar el algoritmo más eficiente a nivel computacional sin perder precisión a la hora de detectar tanto objetos como personas. Esto es debido a que la placa finalmente empleada para implementar todo este sistema es una Raspberry Pi 4, que, a pesar de haber mejorado a lo largo de los años,

sigue teniendo una capacidad limitada para realizar tareas como detección debido a la gran capacidad de procesamiento que requiere.

Tal como se ha comentado en el estado de la cuestión, existen muchos algoritmos que son capaces de detectar objetos, pero unos son más eficientes y permiten realizar estas tareas. Además, también se ha debido tener en cuenta la plataforma en la que se iba a implementar el algoritmo, lo cual aumentaba el rango de posibilidades.

Cabe destacar, que se ha implementado de primera mano varios de los algoritmos disponibles para comprobar su funcionamiento en PyCharm y tener ya una primera noción de cómo sería su ejecución en la Raspberry.

4.3.4 BÚSQUEDA DE ALGORITMOS DE RECONOCIMIENTO FACIAL

Al igual que en el caso anterior, se ha tenido que realizar una búsqueda exhaustiva de los algoritmos más convenientes para este sistema tan limitado a nivel computacional y al final se ha acabado optando por uno muy sencillo de implementar, preciso y muy eficiente.

4.3.5 BÚSQUEDA DE COMPONENTES

Uno de los mayores retos de este proyecto ha sido encontrar los componentes idóneos para disponer de un producto barato y capaz de realizar todas las tareas propuestas. Además, debe contar con puertos accesibles para conectar los diferentes periféricos empleados en este sistema de seguridad (cámara, altavoz, sensor de huellas). El componente principal es el cerebro del sistema y se ha pensado en varias alternativas. Por otro lado, también se han tenido que buscar los distintos componentes necesarios.

4.3.6 IMPLEMENTACIÓN DEL SISTEMA

Una vez recibidos todos los componentes mencionados anteriormente se ha tenido que montar todo el sistema resultando bastante sencillo en todos los casos salvo el de las huellas dactilares.

Por otro lado, cabe destacar que se ha tenido que aprender a manejar varios programas para poder establecer la comunicación entre la Raspberry Pi y el ordenador además de la puesta en marcha de la misma.

4.3.7 APRENDIZAJE DE OPENCV

Para la implementación del algoritmo de detección de objetos se ha hecho necesario aprender a usar la librería OpenCV la cual resultaba ser la más sencilla, la más eficiente y la más completa. Para ello, se ha usado la documentación proporcionada en la web de OpenCV además de varios tutoriales de YouTube.

4.3.8 IMPLEMENTACIÓN DE ALGORITMOS

Una vez aprendido a manejar OpenCV y entendido el funcionamiento de los algoritmos, se ha programado a través de Pycharm probando, en primer lugar, cada algoritmo de forma individual y posteriormente uniendo ambos para ejecutarlos en función de la tarea que tenga que realizar en cada caso.

4.3.9 IMPLEMENTACIÓN DE FUNCIONALIDADES ADICIONALES

Una vez conseguido implementar de forma satisfactoria los algoritmos, se han buscado las librerías que mejores soluciones ofrecían para implementar el streaming de vídeo, la reproducción de audio a través del altavoz y el aviso por correo.

4.3.10 PROGRAMACIÓN DE LA CONFIGURACIÓN DEL SISTEMA

Una vez conseguido programar y resolver todos los problemas que han ido surgiendo a lo largo de todo el desarrollo, se ha tenido que pensar en facilitar la configuración para el usuario que adquiera este sistema.

4.3.11 SMARTWORKS

Para aprender a usar esta plataforma se ha contado con dos sesiones de training en las cuales se han aprendido todos los aspectos básicos de la plataforma y su funcionamiento además

de aprender sobre los protocolos HTTP y MQTT. Una vez aprendido esto, se han ido implementando las funcionalidades que ofrece esta plataforma.

4.3.12 DISEÑO CARCASA 3D

Por último, una vez implementado de forma satisfactoria todos los componentes mencionados anteriormente, se ha procedido a diseñar la carcasa con el uso de SolidWorks de tal forma que el sistema pudiera estar lo mejor refrigerado posible y se puedan integrar los componentes de la forma más sencilla posible.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

El tiempo empleado para el desarrollo del proyecto se muestra en Ilustración 45: Diagrama de Gantt del cronograma. En él se puede observar que varias actividades se superponen con otras debido a que requieren un mayor aprendizaje a lo largo del tiempo. Muchas de las herramientas necesarias son específicas del proyecto y no se han aprendido en el curso de Python realizado, por lo que se requiere seguir aprendiendo a lo largo del desarrollo del proyecto.

Para la programación del código ocurre de forma similar. A medida que se van implementando nuevos componentes al sistema y comprobando que funcionan correctamente de forma independiente, resulta necesario reprogramarlos para que funcionen en sintonía con el sistema de seguridad completo. Cabe destacar que también se ha tenido que ir depurando el código para que funcione de la forma más eficiente posible.

Por otro lado, el resto de las actividades se han ido realizando de forma secuencial. A medida que se conseguía implementar un nuevo periférico exitosamente, se ha pasado a implementar otro nuevo.

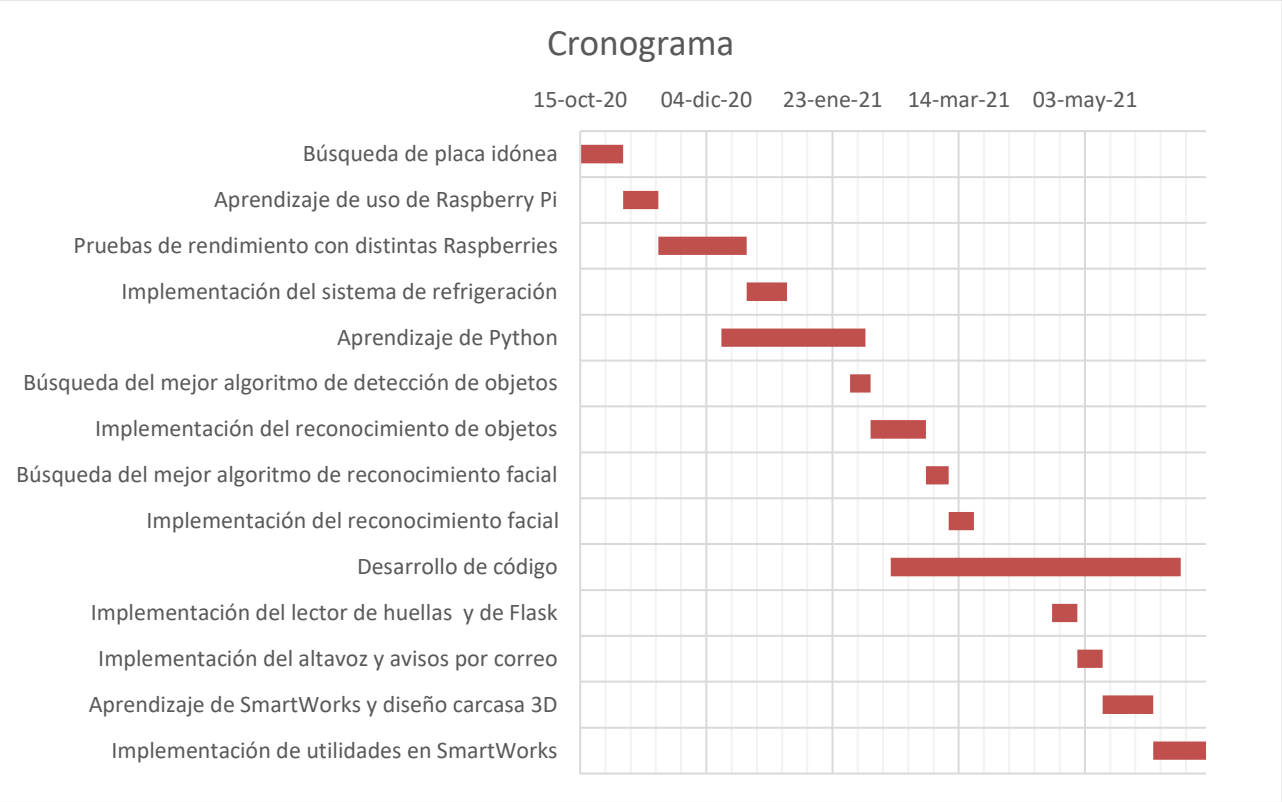


Ilustración 45: Diagrama de Gantt del cronograma

En Ilustración 46: Horas invertidas por actividad se pueden observar las horas aproximadas invertidas para cada actividad. Aproximadamente se han invertido unas 275 horas en el proyecto y la parte que se le ha dedicado más tiempo es el desarrollo del código y su depuración. Por otro lado, la implementación de cada nuevo periférico también ha supuesto bastante tiempo debido a las búsquedas que se tenían que realizar, aprender a usarlos y posteriormente implementarlos.

Por otro lado, en Tabla 2: Presupuesto del sistema de seguridad completo se muestra el coste estimado del desarrollo de este proyecto. Se ha tenido en cuenta que todos los programas empleados para el desarrollo de este sistema de seguridad son gratuitos. Cabe destacar que la licencia de SolidWorks ha sido gratuita, pero en el caso de no ser estudiante su coste se puede elevar hasta los 6600€ sin tener en cuenta su coste de mantenimiento. Hay que añadir que existen programas de diseño 3D gratuitos que permitir diseñar la carcasa de la misma manera. Por otro lado, se ha estimado un coste por hora de unos 15€ para el desarrollo del proyecto dando lugar a $Coste\ trabajador = 275h * \frac{15€}{h} = \mathbf{4125}€$.

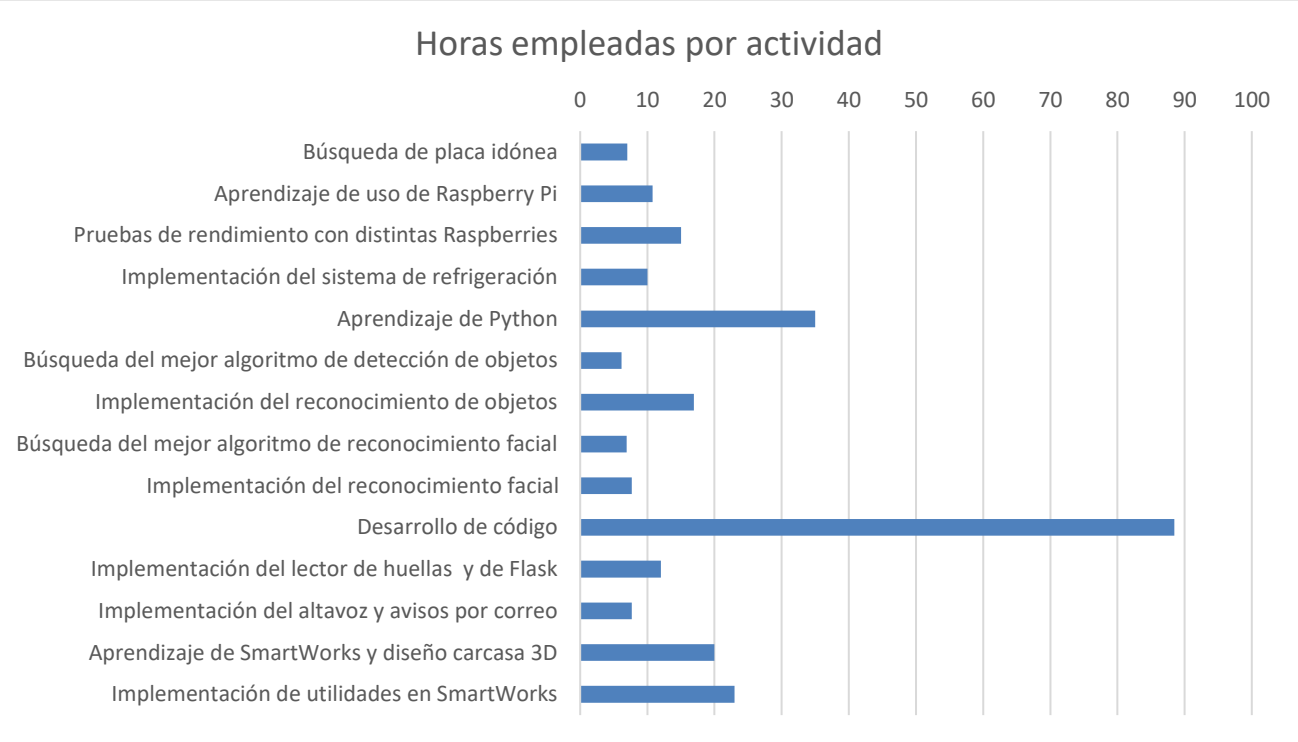


Ilustración 46: Horas invertidas por actividad

Componente	Coste
Coste trabajador	4125€
Raspberry Pi 4B 2Gb RAM	48,75€
Cámara de visión nocturna	12,78€
Lector de huellas	17,61€
Módulo TTL	2,2€
Sistema de refrigeración	8,18€
Altavoz	11,89€
Material y coste de impresión en 3D	30€
4256,41€	

Tabla 2: Presupuesto del sistema de seguridad completo

4.5 OBJETIVOS DE DESARROLLO SOSTENIBLE

Este proyecto está en alineación con el objetivo de desarrollo sostenible en Industria, innovación e infraestructura [27]. Es importante garantizar el uso sostenible de los recursos naturales además del desarrollo económico y social. Es por ello, que la integración en un mismo dispositivo de todos los componentes de este sistema de seguridad permite un uso eficiente de los recursos. Por un lado, reduce el cableado necesario para comunicar los distintos elementos del sistema consiguiendo reducir la cantidad de material empleado. Por otro lado, hay que tener en cuenta que la centralización en un mismo dispositivo de todo el procesamiento de la información consigue eliminar procesadores que estarían situados en dispositivos independientes. Con esto se logra reducir el número de procesadores y consecuentemente, eliminar chips que contienen materiales muy contaminantes. Adicionalmente, este proyecto está contribuyendo en la mejora de las infraestructuras aumentando la seguridad de ellas.

Por otro lado, este proyecto también contribuye a conseguir un mundo más sostenible. El consumo energético de este sistema de seguridad es muy inferior a los productos de la competencia. En concreto, reduce a más de la mitad la potencia necesitada para poner en marcha el sistema. Por otro lado, debido a la necesidad de menos material que dispositivos de la competencia, se consigue reducir la cantidad de desechos generados.

Capítulo 5. DESARROLLO

En este capítulo se va a explicar lo desarrollado en el proyecto. En primer lugar, se va a explicar la elección de los componentes empleados. Por otro lado, para tener una buena idea general del funcionamiento del sistema de seguridad, se va a explicar, en segundo lugar, el esquema general del sistema y a continuación se va a explicar la estructura de funcionamiento del código empleado para poner en marcha el sistema. Por otro lado, si se desea mirar con mayor profundidad el código empleado, se podrá encontrar en el ANEXO II.

5.1 ELECCIÓN DE COMPONENTES

Para la realización de este proyecto resulta necesario disponer de un dispositivo con una capacidad computacional elevada para ejecutar los algoritmos. Además, debe contar con varios puertos para poder conectar los periféricos empleados en este proyecto. Adicionalmente, debe tener un consumo energético reducido.

Se podría haber optado por una Arduino, pero su reducido rendimiento hace inviable su implementación. En el mercado existen otras placas menos populares pero el dispositivo que cumple con los requisitos mencionados anteriormente es una Raspberry Pi. Se ha optado por ella debido a su razonable rendimiento y la disponibilidad de diversos puertos a los que conectar los componentes necesarios facilitando la integración del conjunto del sistema de seguridad. Además, hay que destacar, que su sistema operativo permite la programación y ejecución en Python facilitando la implementación de este sistema de seguridad. Es por ello, que su implementación es idónea. Se aprovechan prácticamente todos los puertos que ofrece la placa, tiene un rendimiento razonable para ejecutar los algoritmos, permite ejecutar programas en Python, tiene un coste razonablemente reducido y tiene un reducido consumo energético.

En primer lugar, se realizaron pruebas con una Raspberry Pi Zero por su reducido tamaño, pero debido a su limitada capacidad computacional se ha tenido que optar por su hermano mayor para disponer de un sistema lo suficientemente rápido para ejecutar todo el sistema. Cabe destacar que también se han realizado pruebas con distintas Raspberry Pi 4 con más memoria RAM, que a pesar de observar que con 4Gb de RAM, el sistema se ejecutaba de forma más fluida, con 2Gb de RAM es más que suficiente.

En segundo lugar, se ha tenido que buscar una cámara con capacidad de visión nocturna con la resolución suficiente para poder detectar de forma correcta. Una vez realizado esto se ha buscado un altavoz sencillo que se pudiera conectar a través del puerto Jack de la Raspberry y también se ha buscado el sensor de huellas más sencillo de implementar.

Por otro lado, debido a los calentamientos que se producían a los pocos minutos de probar los algoritmos, se ha tenido que pensar en varias alternativas. La Raspberry llegaba a alcanzar temperaturas de 80º en 2 minutos y para un dispositivo que debe estar en funcionamiento de forma continua, se hacía inviable. En primer lugar, a través de código se he intentado reducir los tiempos de uso de los algoritmos. Para ello, se paralizaba el programa unos segundos mediante el uso de la función sleep. De esta manera se conseguía reducir la carga de trabajo de la CPU y, consecuentemente, se conseguía reducir la temperatura del sistema. Se ha conseguido implementar con éxito, pero se ha encontrado una mejor alternativa.

Inicialmente el sistema contaba con unos disipadores básicos sin ventilación. Se ha visto que la solución comentada anteriormente ayudaba a reducir la temperatura a corto plazo, pero a medio-largo plazo iba aumentando. Para eliminar este problema, se ha buscado un sistema de refrigeración que contara con disipadores y ventiladores. En el mercado existen varias opciones baratas y gracias al empleo de uno de ellos se ha conseguido estabilizar la temperatura a unos 50º siendo un valor muy razonable.

5.2 ESQUEMA GENERAL

El funcionamiento general del sistema puede simplificarse tal como se muestra en la ilustración que se muestra a continuación.

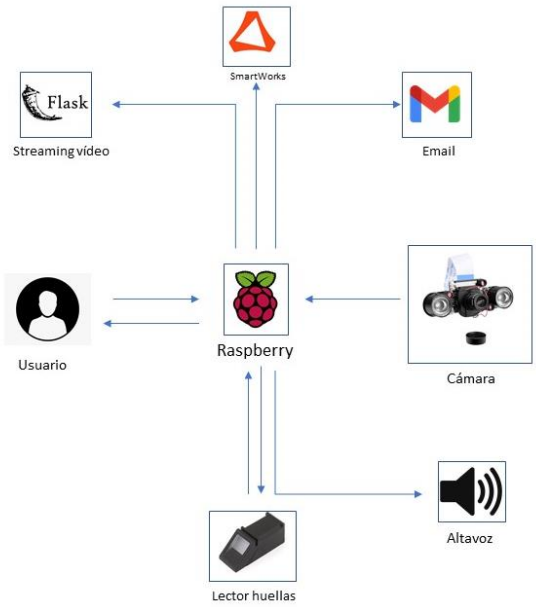


Ilustración 47: Esquema general del sistema de seguridad

La Raspberry Pi 4 es el cerebro del sistema y es la que realiza el procesamiento de los datos obtenidos de la cámara, del usuario y del lector de huellas. Además de procesar toda esta información, tiene la función de enviar a los diferentes periféricos del sistema la información necesaria para el correcto funcionamiento de los mismos.

A continuación, se va a explicar en detalle la función de cada elemento del sistema.

5.2.1 USUARIO

Para describir al usuario resulta necesario distinguirlo en los dos casos en los que puede interactuar con el sistema de seguridad: en la configuración del sistema y el caso en el que está en funcionamiento el sistema.

En primer lugar, cuando se adquiere el producto, resulta necesario configurar el sistema puesto que es necesario aportar una imagen de las personas a las que se les quiere permitir el acceso al domicilio. Además, también es necesario registrar las huellas dactilares y los correos electrónicos a los cuales se deben enviar avisos. Para poder realizar esta configuración, el usuario tiene que ejecutar el programa y una vez arrancado se mostrará un menú como el de la ilustración.

```
Bienvenido al asistente de configuración del sistema de seguridad:
 Elija la opción:
1.- Configurar persona con derecho de admisión
2.- Configurar correo para el envío de mensajes
3.- Poner en marcha sistema de seguridad
4.- Configurar nueva huella dactilar o eliminar huella registrada
5.- Mostrar perfiles configurados
Escriba en el teclado la opción que desee:4
```

Ilustración 48: Menú inicial

Una vez configurado el sistema, se pone en marcha el sistema de seguridad y empieza detectando objetos y personas. El usuario vuelve a interactuar con el sistema cuando vuelve a casa y el sistema tiene que detectarle como persona y posteriormente reconocerle facialmente. Una vez reconocido facialmente, el usuario tendrá que colocar la huella registrada en el lector de huellas para realizar la verificación completa.

5.2.2 RASPBERRY PI 4B

Tal como se comentó anteriormente, es el cerebro del sistema y será la encargada de procesar toda la información recibida por los periféricos del sistema y de intercambiar la información procesada.

En primer lugar, la Raspberry está continuamente recibiendo imagen de la cámara y mediante el uso del algoritmo de reconocimiento de objetos va a estar analizando los objetos que van apareciendo en la zona. Cuando se detecta una persona durante cierto tiempo, el sistema entiende que una persona desea acceder al domicilio y pasa al algoritmo de reconocimiento facial. Si la persona es reconocida con éxito, el usuario debe colocar la huella dactilar para permitir el acceso al domicilio.

Por otro lado, mientras está en funcionamiento todo el apartado de seguridad, la Raspberry está enviando de forma constante imagen en tiempo real a Flask permitiendo de esta forma observar el estado de la zona de forma remota. Además, también es la encargada de enviar los avisos por correo electrónico en los casos de detección de una persona y en aquellos casos en los que se permite el acceso de una persona al domicilio. Cabe destacar que la Raspberry está continuamente enviando información del estado del sistema a través del protocolo MQTT a la plataforma SmartWorks.

5.2.3 CÁMARA

Es la encargada de aportar la imagen necesaria para analizarla posteriormente en la Raspberry Pi. Además, para no limitar su funcionamiento a las horas de luz, cuenta con visión nocturna. Va conectada con un único cable al puerto de módulos de cámara de la Raspberry, la cual aporta corriente necesaria para hacerla funcionar y el intercambio de información entre Raspberry y cámara.

5.2.4 LECTOR DE HUELLAS

Tiene el objetivo de realizar la verificación por huellas dactilares aumentando de esta forma la seguridad del sistema. Se trata de un lector óptico y en los casos en los que se desea verificar una huella, se enciende una luz verde. Cabe destacar que la información de las huellas registradas está almacenada en el propio sensor y únicamente se envía la información de si la huella coincide o no con una de las registradas. El lector de huellas está conectado gracias a un módulo que se puede conectar a uno de los puertos USB de la Raspberry. Este módulo ofrece los pines necesarios para el correcto funcionamiento del lector. Cuenta con un pin para la alimentación del sensor, otro de tierra, otro RX para recibir los datos y otro TX para transmitir los datos.

5.2.5 ALTAVOZ

Tiene la función de indicar al usuario lo que tiene que realizar tanto en el proceso de configuración del sistema como en el proceso de verificación. En el caso que se quiera registrar una persona, se indicará al usuario que se coloque en frente de la cámara y la

introducción del nombre del usuario. Por otro lado, para el registro de huellas dactilares se indicará de forma similar al usuario cómo debe realizarlo. En los casos que una de las personas registradas desea acceder al domicilio, el altavoz indicará al usuario que se debe colocar frente a la cámara y una vez detectada a la persona se dirá su nombre y que debe colocar la huella en el sensor. Cabe destacar que está conectado a través del puerto Jack de la Raspberry y está alimentado mediante uno de los puertos USB.

5.2.6 FLASK

Es la plataforma que se emplea para visualizar en tiempo real la imagen que tiene el sistema de seguridad de la zona. Esta plataforma permite al usuario acceder, introduciendo en cualquier navegador internet la dirección IP generada por el programa, desde cualquier ubicación. Cabe destacar que la dirección IP generada se muestra al ejecutar por primera vez el programa y será la misma para todas las ejecuciones.

5.2.7 GMAIL

Los avisos que se vayan a notificar se realizan a través de Gmail. Cada vez que el algoritmo de reconocimiento de objetos detecta una persona durante cierto tiempo enviará una imagen por correo electrónico a aquellos usuarios registrados en el sistema. Esta notificación se emplea para saber quién pudiera querer acceder, para saber si ha llegado un paquete de mensajería o para saber si una persona desconocida está intentando acceder al domicilio. Por otro lado, como medida de seguridad adicional, se enviará otra notificación una vez que un usuario introduce la huella dactilar en el sensor tanto en el caso que la huella coincida con las registradas en el sistema como si no coincide.

5.2.8 SMARTWORKS

Además de mostrar en tiempo real la detección que está realizando el sistema, la Raspberry envía información del estado de la misma a Smart Works. La información que se envía es del estado en el que se encuentra el sistema de seguridad (detectando objetos, reconociendo facialmente una cara, verificando una huella dactilar, configurando el sistema…). Además,

se envían datos de la temperatura de la Raspberry y del uso de la CPU. Toda esta información es analizada y monitorizada en esta plataforma.

En esta plataforma se analizan los valores de la temperatura y del uso de la CPU para indicar estados. Estos valores son analizados por funciones que se ejecutan en los servidores de SmartWorks y de esta forma se reduce la carga de trabajo de la Raspberry. Por otro lado, se emplean estos datos para representar gráficos y facilitar la monitorización del sistema por parte del usuario.

5.3 ESQUEMA DEL PROGRAMA PRINCIPAL

Una vez entendidos las funciones de los periféricos del sistema de seguridad, se va a explicar el funcionamiento del código.

En primer lugar, hay que distinguir las distintas carpetas que forman el programa. Por un lado, se encuentra la carpeta de módulos en la cual se van a encontrar los distintos módulos (archivos de Python que realizan una función concreta). En la ilustración que se muestra a continuación se pueden observar todos los módulos.

Por otro lado, se encuentra la carpeta de Resources (Recursos), en la cual se encuentran varias carpetas con los archivos necesarios para la ejecución de los módulos. En ella se encuentran:

- Carpeta de audio: En ella se encuentran los archivos .mp3 que se van a generar cada vez que se inicie el programa y son los que se van a emplear para orientar al usuario en el proceso de configuración del sistema de seguridad y en el acceso al domicilio.
- Carpeta YOLO: En esta carpeta se encuentra el archivo del modelo entrenado del algoritmo YOLO. Junto a este archivo, se encuentra otro archivo empleado para configurar la red neuronal de YOLO. Cabe destacar que estos archivos han sido obtenidos de la página oficial de YOLO y se han elegido las versiones más simples puesto que las versiones con una arquitectura más compleja suponían una enorme carga de trabajo para la Raspberry Pi.

- Carpeta FaceRecognition: En ella se almacenan las imágenes con los nombres de las personas registradas en el sistema. El algoritmo emplea estas imágenes para realizar la verificación facial de las personas que desean acceder al domicilio.
- Carpeta Gmail: En ella se encuentra el archivo que almacena los datos de los usuarios que han registrado su correo electrónico.

Por último, lo que se encuentra en la carpeta principal es el archivo main.py. Este es el archivo Python que se ejecuta y pone en marcha todo el sistema de seguridad.

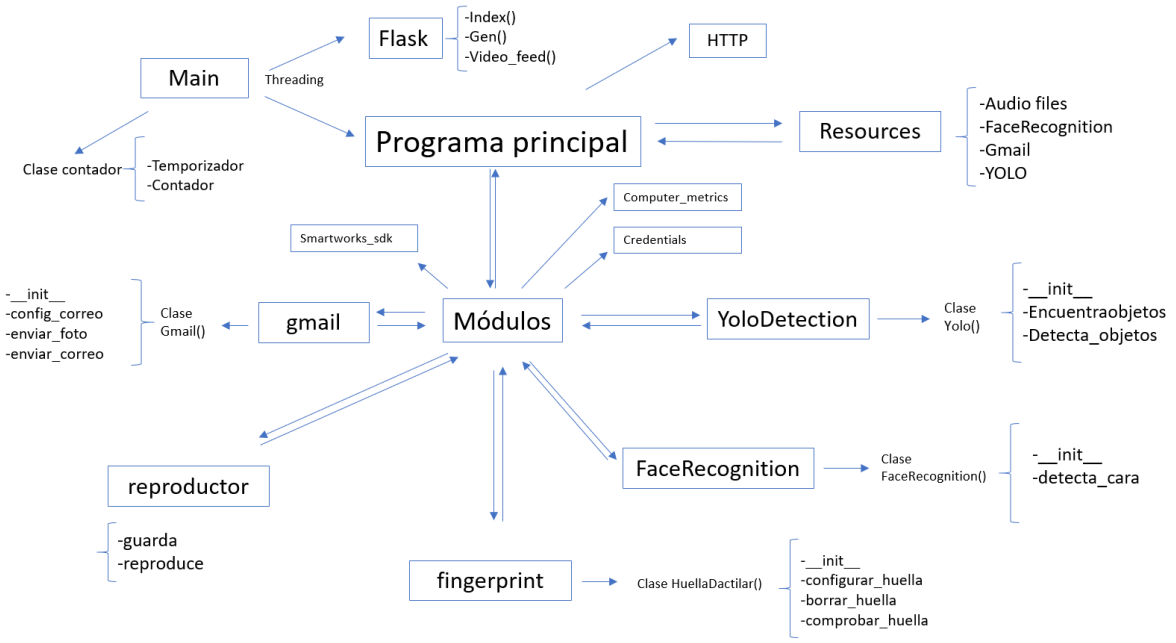


Ilustración 49: Esquema general del archivo main.py

Antes de comentar cada uno de los elementos del esquema anterior, se va a explicar el funcionamiento general del archivo main.py. Cabe destacar que en el Anexo II se encuentra el código del main.py junto con los de los distintos módulos y todo está comentado para entender mejor el funcionamiento.

El programa importa los archivos de todos los módulos que se van a emplear y está compuesto por una clase que se va a emplear como temporizador. Por otro lado, se generan los archivos de audio y se inicializan las variables necesarias para el protocolo MQTT. En

primer lugar, se va a ejecutar la librería Flask y mediante el uso de la librería threading se va a ejecutar de forma paralela el programa principal, el cual se inicia con el asistente de configuración inicial del sistema de seguridad. Se hace vital usar la librería Threading puesto que, si el programa únicamente funcionara en un hilo, se producirían interrupciones a la hora de mostrar el vídeo en tiempo real cuando el programa estuviera analizando la imagen recibida de la cámara. Cabe destacar que en Python existen dos opciones para ejecutar dos programas de forma paralela y son multiprocessing y threading. Se ha optado por la segunda opción porque resulta ser más sencillo y además comparte el mismo espacio de datos entre los distintos hilos. La librería threading permite generar distintos hilos en los que cada uno de ellos se realizan operaciones diferentes. En este caso se ejecuta en un hilo Flask y en otro hilo se ejecuta el programa principal. Por otro lado, debido a la ventaja añadida de compartir en un mismo espacio todos los datos, facilita la comunicación entre ambos hilos y de esta forma una vez que se ha analizado la imagen, se puede compartir esta misma a Flask para que éste la utilice para enviarla por internet.

Una vez que el programa principal está en marcha se muestra un menú en el que se pueden realizar las siguientes operaciones:

- Configurar persona con derecho de admisión
- Configurar el correo electrónico
- Puesta en marcha del sistema de seguridad
- Configurar el lector de huellas dactilares
- Mostrar los perfiles configurados

Salvo para el caso de la puesta en marcha del sistema de seguridad, el resto de los casos no tienen funciones de especial interés y se pueden observar de forma detallada en el Anexo II. El resto de las funciones van a ser comentadas más adelante para comprender su funcionamiento. Además, se comentarán las distintas clases empleadas.

5.4 CLASES

Tal como se ha podido observar en el esquema mostrado en el apartado anterior, el programa está compuesto por 5 clases. Cabe destacar que con el fin de simplificar el código se han distribuido las clases en los distintos módulos mencionados anteriormente. Con el objetivo de tener una mejor visión general, en la tabla que se muestra a continuación se pueden observar las distintas funciones que pertenecen a cada una de las 5 clases.

CLASE	FUNCIONES o MÉTODOS
Contador()	• __init__() • contador() • temporizador()
Yolo()	• __init__() • encuentra_objetos • detecta_objetos
gmail()	• __init__() • config_correo() • enviar_foto() • enviar_correo()
FaceRecognition()	• __init__() • detecta_cara()
HuellaDactilar()	• __init__ • configurar_huella() • borrar_huella() • comprobar_huella()

Tabla 3: Clases del sistema con sus respectivas funciones

La única clase que no se encuentra en un módulo es la clase Contador, la cual se encuentra en el main. Esta clase se emplea para contar tiempos. Cuenta con dos métodos y cada uno corresponde a un algoritmo.

Tal como cabe esperar, la clase Yolo se emplea para la detección de objetos mientras que la clase FaceRecognition se emplea para el reconocimiento facial. Por otro lado, la clase Gmail tiene como objetivo enviar los mensajes por correo electrónico y está compuesto por los métodos mostrados en la tabla. Por último, la clase HuellaDactilar se emplea para todo lo relacionado con el reconocimiento de huellas dactilares.

5.5 FUNCIONES

En este apartado se van a explicar por encima las funciones empleadas en el programa para poder comprender de forma básica el funcionamiento del sistema de seguridad.

5.5.1 CONTADOR

Esta función pertenece a la clase Contador y es la empleada cuando pasan más de 15 segundos desde que se inicia el reconocimiento facial. Solo se ejecuta cuando no se detecta la cara de la persona y únicamente cambia la variable encargada de hacer que el programa ejecute un algoritmo u otro.

5.5.2 TEMPORIZADOR

Esta función también pertenece a la clase Contador y se emplea cuando pasan más de 30 segundos desde que se inicia el reconocimiento de objetos. Esta función es necesaria porque reinicia el contador de detecciones de una persona. El contador de detecciones de persona se emplea para los casos en los que se detecta una persona un único instante y no salte directamente al reconocimiento facial. Resulta muy necesario para los casos en los que se vaya a instalar el sistema en la calle y estén continuamente pasando peatones. Además de reiniciar el contador, también se actualizan los datos de los parámetros del sistema que posteriormente se envían a SmartWorks. Se realiza esto cada 30 segundos para no ralentizar

el sistema de forma innecesaria ya que no resulta vital conocer el estado de la CPU en cada instante.

5.5.3 APILAR_IMAGENES

Es la única función perteneciente al programa principal y tal como su nombre indica tiene como objetivo unir las distintas imágenes registradas en el sistema. Para ello une de forma horizontal y vertical las imágenes de tal forma que se muestre en pantalla todas las imágenes de los usuarios registrados en una misma imagen. Esto ayuda al usuario a identificar su imagen y solicitar su eliminación o cambio.

5.5.4 DETECTA_OBJETOS

Se encuentra en la clase Yolo y con ella se obtienen las salidas de la red neuronal YOLO. En primer lugar, se encarga de convertir la imagen de la cámara a un formato legible para la red neuronal YOLO y para ello emplea la función blob de OpenCV. Una vez realizada la conversión, se introduce la imagen convertida en la red neuronal y se obtienen los nombres de las salidas de las capas de la red y estos son utilizados para obtener los datos analizados por la red (los bounding boxes).

5.5.5 ENCUENTRA_OBJETOS

Esta función también se encuentra en la clase Yolo y se encarga de usar los datos obtenidos de la función detecta_objetos. Además de recibir estos datos, también recibe la imagen (el frame del instante). Tiene como objetivo dibujar un rectángulo alrededor de los objetos detectados y comprobar si se ha detectado a una persona. Para ello recorre con dos bucles for los valores de las salidas para analizar los posibles objetos detectados y, si la confianza del bounding box supera el umbral límite establecido, se guardan los valores de la posición del objeto detectado. Esta posición se almacena para posteriormente dibujar el recuadro alrededor de él mediante el uso de OpenCV. Por otro lado, mediante otro bucle for y empleando los índices obtenidos de los objetos detectados, se comprueba si hay uno que coincide con la clase person para saber si se ha detectado una persona. Si es así, se actualiza

la variable que almacena la presencia de una persona y se contabiliza para sumar en la variable contador de personas.

5.5.6 CONFIG_CORREO

Esta función pertenece a la clase Gmail y tal como su nombre indica sirve para configurar el correo que desea registrar el usuario. Para ello, solicita el correo electrónico y la contraseña del usuario y almacena estos datos en un archivo .txt. Este archivo queda guardado en la carpeta Gmail y se emplea en los casos en los que se desea enviar un correo.

5.5.7 ENVIAR_FOTO

También pertenece a la clase Gmail y es la encargada de enviar una imagen de la persona detectada. Como es de esperar recibe la imagen desde el programa principal. En primer lugar, abre el archivo con los datos del usuario y mediante el uso de funciones pertenecientes a la librería MIME se realiza el envío del correo.

5.5.8 ENVIAR_CORREO

Tiene una función similar a la función enviar_foto. Se diferencia en que esta es capaz de indicar el nombre de la persona que ha accedido.

5.5.9 DETECTA_CARA

Esta función pertenece a la clase FaceRecognition y esta única función se encarga de realizar todo el proceso de reconocimiento facial. Para ello, en primer lugar, reescala el frame a analizar para obtener una mayor velocidad de procesamiento. A continuación, se convierte la imagen a RGB porque es el formato que es capaz de leer la librería de face_recognition y se buscan todas las caras que pueda haber en la imagen. Una vez detectada la cara, se compara con las registradas en el sistema y se almacena la posición de la misma para poder dibujar el rectángulo alrededor de ella. Al igual que se hace en la función encuentra_objetos, mediante el uso de OpenCV se dibuja el recuadro alrededor de la cara y también se muestra el nombre de la persona detectada. Puesto que posteriormente se va a usar la imagen

analizada para mostrarla en streaming, se convierte a un formato compatible para Flask y se devuelve al programa principal el resultado.

5.5.10 CONFIGURAR_HUELLA

Perteneciente a la clase HuellaDactilar, se emplea para registrar una nueva huella en el sistema. Para ello, el sensor activa la luz para tomar una imagen de la huella y en la función se convierte la imagen a un formato legible. En primer lugar, comprueba si la huella ya está registrada y si no lo está, registra la nueva huella en el sistema.

5.5.11 BORRAR_HUELLA

Tal como su nombre indica, esta función perteneciente a la clase HuellaDactilar, tiene el objetivo de eliminar una huella previamente registrada. Para ello, la función solicita al usuario qué huella desea eliminar y la función se encarga de borrarla.

5.5.12 COMPROBAR_HUELLA

Esta es la función que se emplea cuando se ha detectado la cara de una persona de forma satisfactoria. Tiene la finalidad de comprobar que la huella colocada en el sensor coincide con una de las registradas en el sistema. A diferencia de lo que se realiza en las funciones de los algoritmos, se almacena el resultado en un atributo al que tiene acceso el programa principal evitando de este modo poner un return.

5.5.13 GUARDA

Esta función pertenece a la clase reproductor y se encarga de almacenar un archivo .mp3 en la carpeta de audios. Para conseguir este resultado, emplea la librería gtts de Google, la cual convierte líneas de código en audio. Cabe destacar que se han probado otras alternativas, pero las voces sonaban muy robóticas.

5.5.14 REPRODUCE

Tal como su nombre indica, se encarga de reproducir los archivos .mp3 generados con la función guarda. Para ello se emplea la librería pygame, la cual está realmente pensada para

la creación de videojuegos en Python. No se ha escogido otras alternativas más sencillas como playsound debido a que no funcionan en la Raspberry Pi. Cabe destacar que se ha realizado una profunda búsqueda por si hubiera alguna otra alternativa, pero ninguna ha funcionado.

5.5.15 PUBLISH

Esta función pertenece a la librería paho-mqtt y se emplea para enviar los datos de interés a la plataforma SmartWorks. Para ello, la función recibe el topic a la cual se desea enviar y el payload. El topic es la dirección a la que el bróker debe enviar la información de la Raspberry. Por otro lado, el payload es la información que se desea enviar.

5.6 SMARTWORKS

La plataforma SmartWorks se emplea para monitorizar el estado de la Raspberry y mediante la creación de gráficas facilitar esta monitorización. Por un lado, recibe datos de la temperatura, uso de CPU, memoria RAM disponible de la Raspberry y estado del sistema de seguridad e indica el estado mediante funciones que se ejecutan en los servidores de Altair.

Inicialmente se empleó el protocolo HTTP, pero realizando pruebas con la Raspberry se ha observado que el protocolo MQTT es más rápido. Con HTTP se producían serias ralentizaciones de la detección de objetos y se ha optado prescindir de él.

5.6.1 FUNCIÓN DATAUPDATE

Esta función recibe la información de la temperatura y del uso de la CPU de la Raspberry Pi. En función de los valores de la temperatura, se indica el estado (crítica, normal y baja). Por otro lado, los datos de uso de la CPU se emplean para indicar si se encuentra en estado crítico, normal o bajo uso. Esta información es enviada al topic de estado correspondiente y se puede observar en el menú principal de la plataforma.

5.7 INTERFAZ

Puesto que este producto está dirigido a un usuario final que no dispone de conocimientos de programación, se ha tenido que pensar en facilitar el trabajo de configuración del sistema de seguridad. Para ello, se va a mostrar en los siguientes apartados, las distintas plataformas con las que puede interactuar el usuario.

5.7.1 CONFIGURACIÓN DEL SISTEMA DE SEGURIDAD

Para poder configurar el sistema de seguridad, el usuario debe ejecutar el archivo Python con el programa principal. Este archivo, en primer lugar, ejecuta Flask y, en segundo lugar, ejecuta paralelamente el programa principal.

Nada más iniciar el programa aparecerá lo siguiente en la terminal.

```
pi@raspberrypi:~/TFG/WebRecognition $ python3 mqttsmartworks.py
pygame 1.9.4.post1
Hello from the pygame community. https://www.pygame.org/contribute.html
37.972
Objeto yolo creado:
Objeto video creado:
 * Serving Flask app "mqttsmartworks" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.31.35:5005/ (Press CTRL+C to quit)
2
Bienvenido al asistente de configuración del sistema de seguridad:
 Elija la opción:
1.- Configurar persona con derecho de admisión
2.- Configurar correo para el envío de mensajes
3.- Poner en marcha sistema de seguridad
4.- Configurar nueva huella dactilar o eliminar huella registrada
5.- Mostrar perfiles configurados
Escriba en el teclado la opción que desee:
```

Ilustración 50: Ventana inicial al ejecutar el programa

Tal como se puede observar, se muestra la dirección IP a la cual se puede acceder para observar en tiempo real la detección de objetos y caras. Por otro lado, se despliega un menú

con varias opciones de configuración. Para acceder a una opción, se escribe en el teclado el número de la opción.

Si se selecciona la opción 1, se configura una persona con derecho de admisión. Para ello, se pide, en primer lugar, el nombre de la persona. Este nombre se emplea para guardar la foto con ese nombre. A continuación, se abre una ventana con la imagen en tiempo real de la cámara para facilitar al usuario colocarse de forma adecuada y realizar la foto de forma satisfactoria. Una vez colocado, se tiene que pulsar la tecla f y se almacena la imagen.

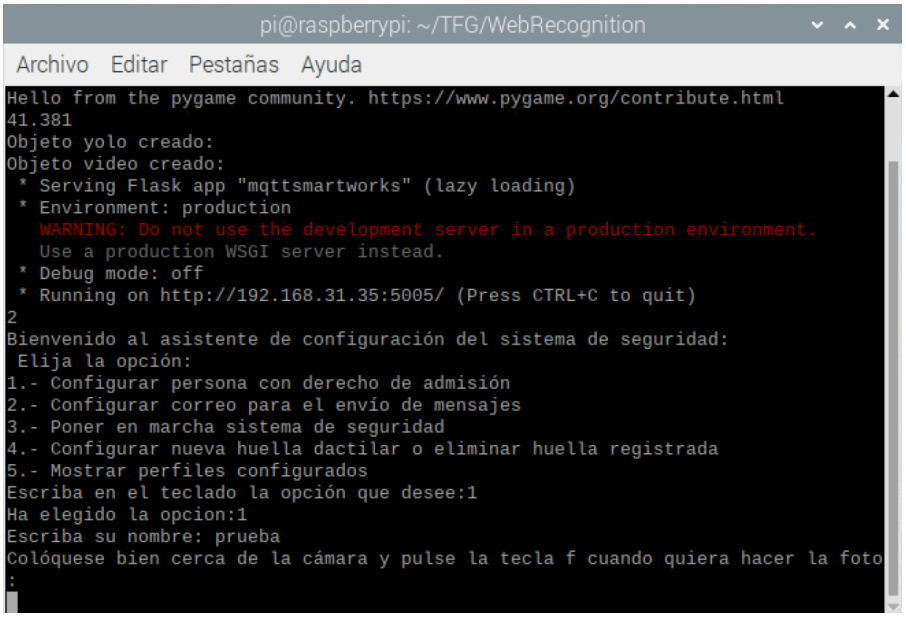


Ilustración 51: Configuración de una nueva persona en el sistema

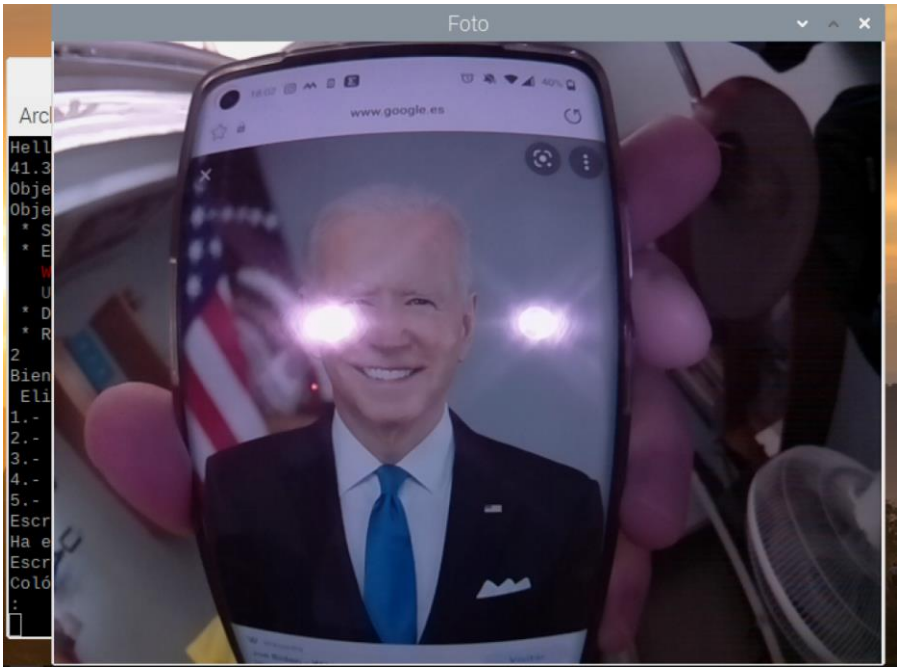


Ilustración 52: Ventana que se abre para facilitar al usuario colocarse de forma adecuada

Si se pulsa la opción 2 se configura el correo que se desea utilizar para enviar los avisos. Para ello, solicita, en primer lugar, el correo electrónico, y a continuación, la contraseña del correo. Una vez introducidos estos datos, el sistema los almacena en un archivo al cual se puede acceder en los casos que se requiera.

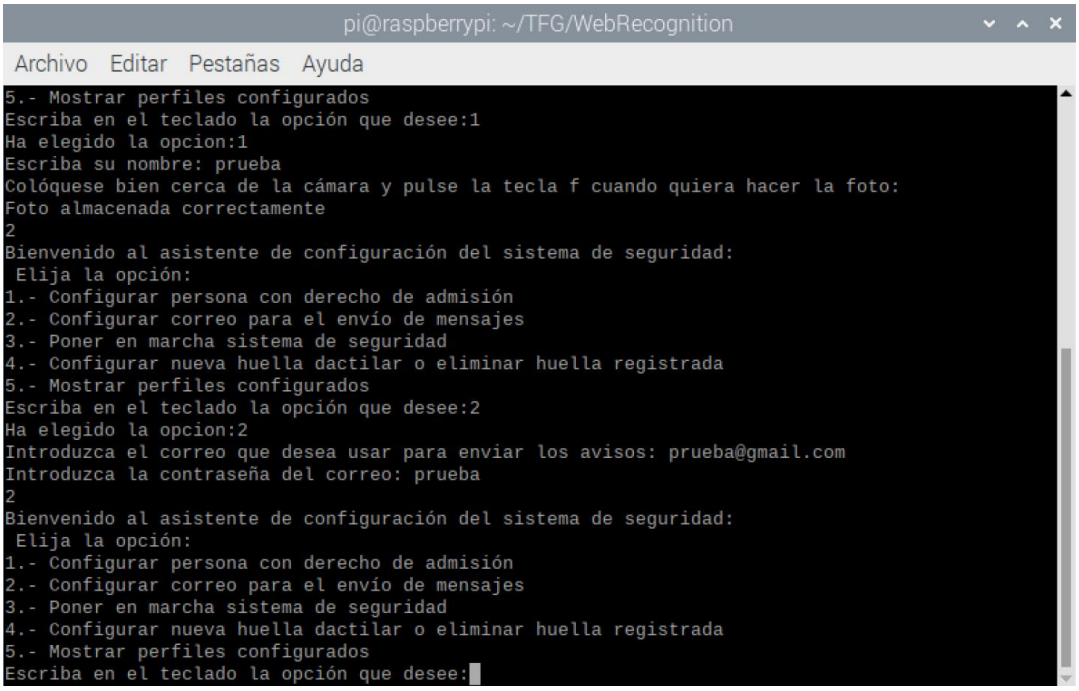


Ilustración 53: Configuración del correo electrónico

Si se selecciona la tercera opción, se pone en marcha el sistema de seguridad. Para ello se ponen en funcionamiento los distintos algoritmos del sistema. En la interfaz de usuario de Raspberry Pi OS no se abre ninguna ventana para no ralentizar el sistema y la única forma de observar lo que ve la cámara es acceder a la dirección IP generada por Flask al inicio de la ejecución del programa.

Si se selecciona la cuarta opción, el programa entra en la configuración de las huellas dactilares. Una vez se accede, se tiene que indicar si se quiere configurar una nueva huella o eliminar una registrada en el sistema. Esta configuración se puede observar en la imagen se muestra a continuación.

Ilustración 54: Registro de una nueva huella dactilar

5.7.2 FLASK

Es la plataforma donde se puede observar en tiempo real lo que observa la cámara y el análisis realizado por los algoritmos. Para poder acceder hay que conocer la dirección IP generada por Flask. Esta dirección IP es la que aparecía al iniciar el programa. Para ello, se introduce en cualquier navegador internet la dirección IP y se mostrará una ventana como las de las ilustraciones siguientes.

Cabe destacar que la carpeta de archivos del programa cuenta con un archivo HTML en el que se ha programado para que la pestaña en la cual se va mostrar la imagen, se llame “Sistema de seguridad Raspberry Pi 4” y también para que por encima de la ventana se muestre esto también. La programación de este archivo se peude encontrar al final del anexo II de este documento.

Ilustración 55: Interfaz de Flask con la detección de objetos activa

Ilustración 56: Interfaz en Flask con el reconocimiento facial activo

Tal como se puede observar, para la detección de objetos se indica el tipo de objeto y la confianza que tiene el algoritmo que sea de ese tipo. Por otro lado, cuando la detección facial está activa, solo se muestra el nombre de la persona registrada y en el caso que no lo esté, se muestra que es Desconocida.

5.7.3 GMAIL

Por otro lado, cuando el algoritmo de detección de objetos detecta una persona, automáticamente envía un correo electrónico. Cabe destacar que se ha programado de tal forma que se envían correos de una persona cuando se detecta un número de veces en un intervalo de tiempo. Esto se realiza para evitar correos innecesarios de personas que pasen unos segundos.

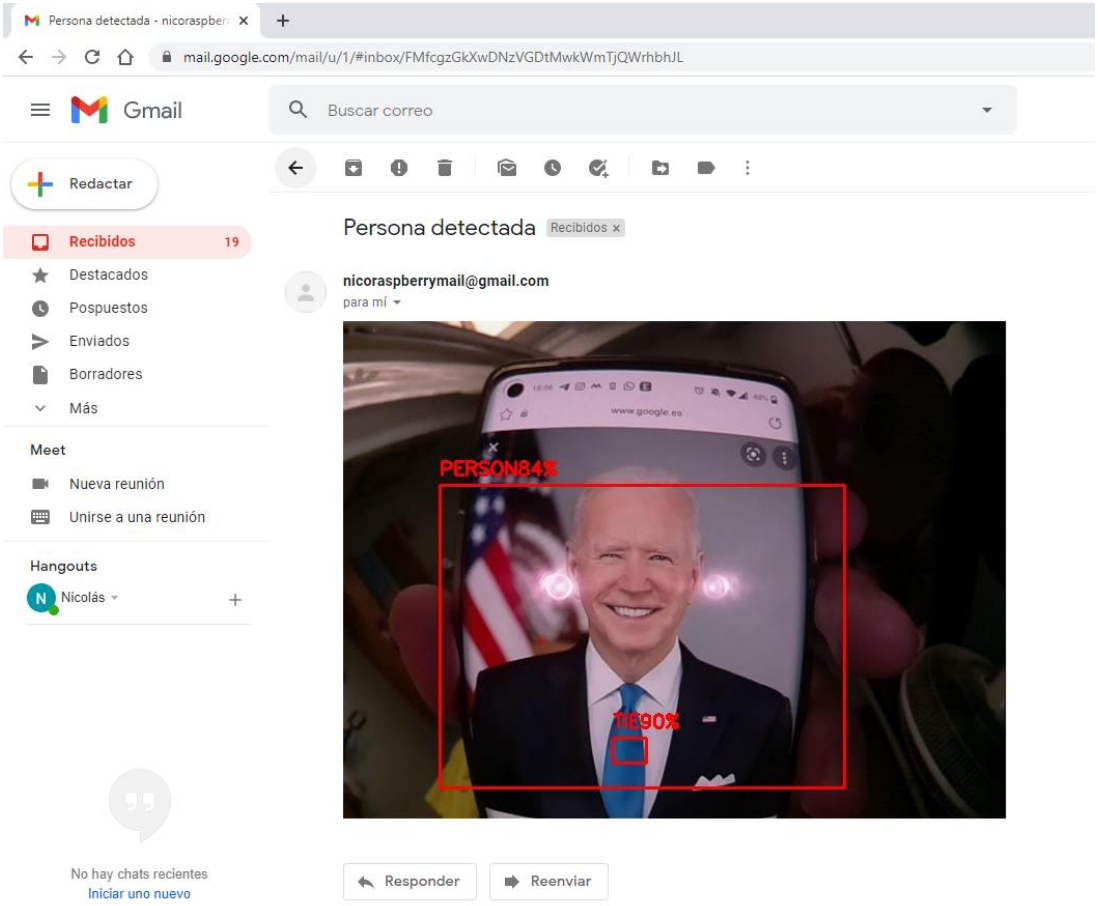


Ilustración 57: Aviso por correo de la detección de una persona

Cuando la persona reconocida introduce la huella dactilar y coincide con una de las registradas en el sistema, el programa envía por correo electrónico una imagen de la persona y en el asunto se indica quién ha accedido. Cabe destacar que el altavoz indica al usuario las acciones que debe realizar. Cuando se detecta una persona, el altavoz se activa e indica a la persona que se acerque a la cámara. Una vez reconocida la persona, el altavoz indica al usuario que debe colocar el dedo en el sensor de huellas.

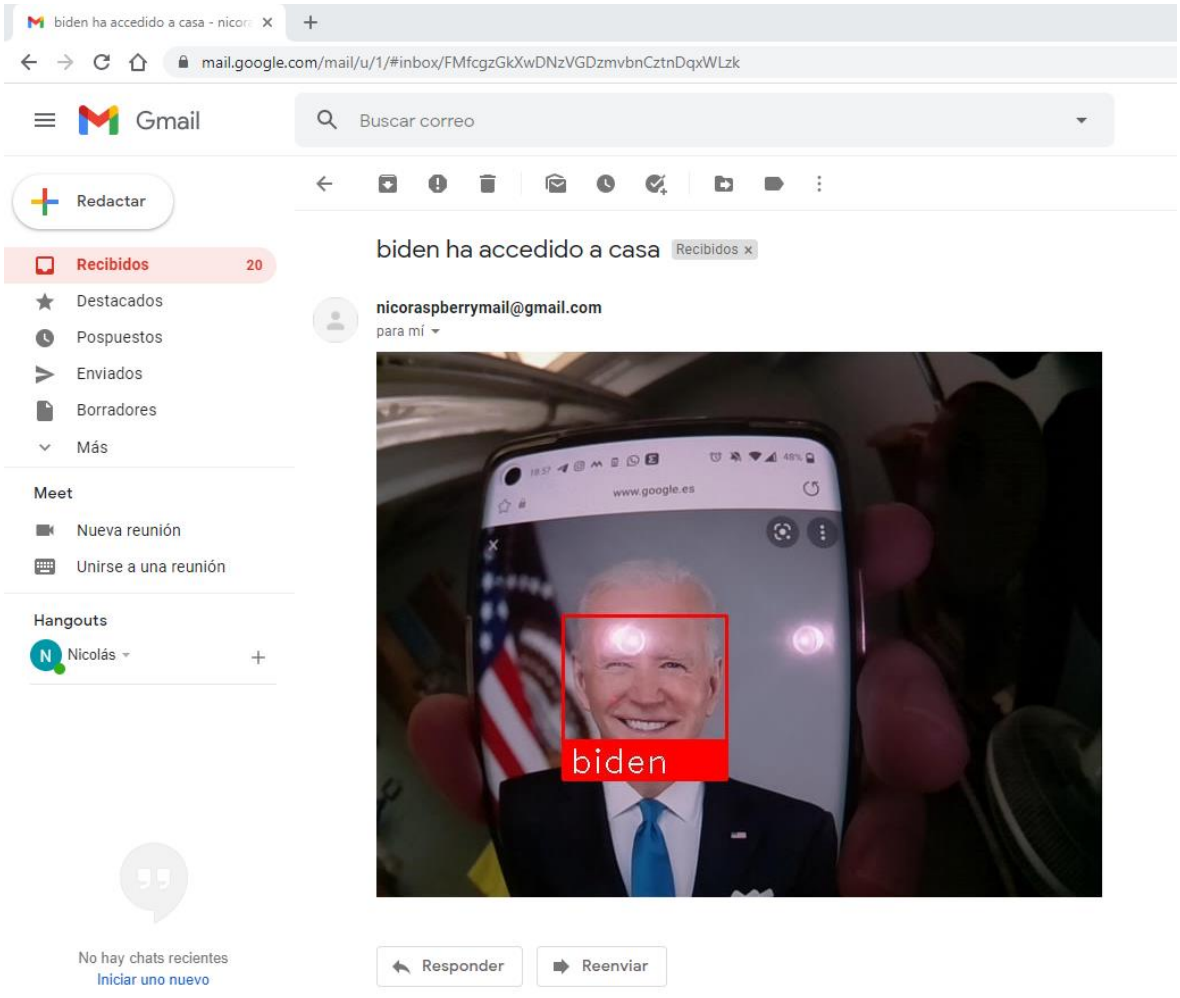


Ilustración 58: Correo recibido al permitir el acceso de una persona

5.7.4 SMARTWORKS

En esta plataforma el usuario puede observar los valores de distintas propiedades del sistema de seguridad. Además, puede observar distintos gráficos.

En primer lugar, cuando el usuario accede a la plataforma se encuentra una ventana como la de la Ilustración 59: Interfaz de SmartWorks.

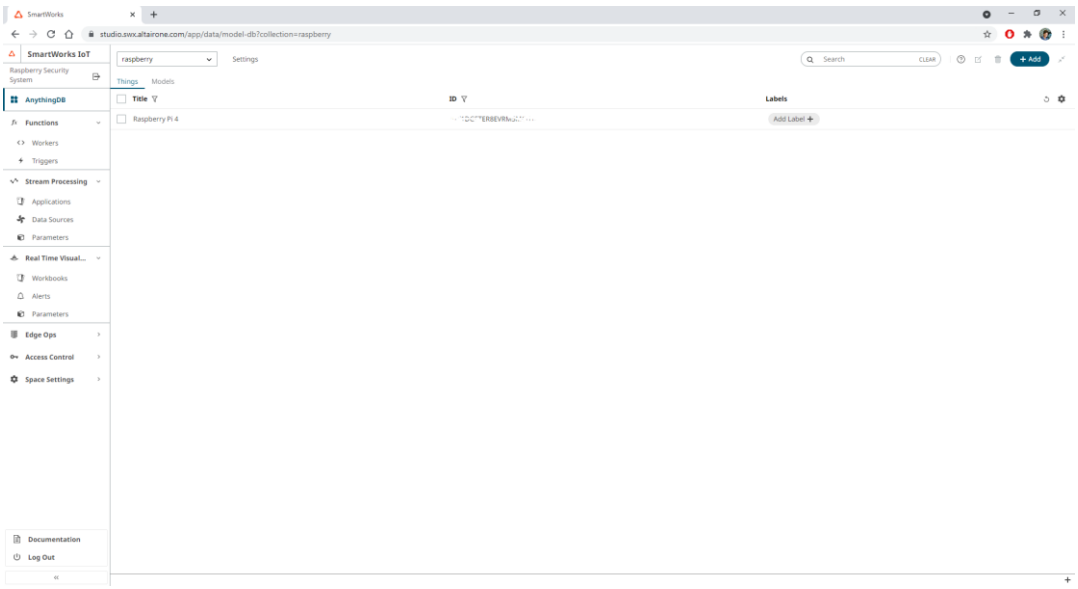


Ilustración 59: Interfaz de SmartWorks

En esta primera ventana se muestran todos los dispositivos configurados y seleccionando un dispositivo se puede observar la información que se muestra en la Ilustración 60: Propiedades del sistema de seguridad. En ella, se pueden observar los recursos consumidos por el sistema, la temperatura y el estado del sistema. Cabe destacar que estas propiedades se van actualizando de forma automática y ofrecen una vista general de la situación del sistema de seguridad. Por otro lado, hay que mencionar que estas propiedades se pueden configurar. Si se desea aportar más información del sistema, se pueden añadir propiedades y programar en el sistema de seguridad para enviar estos datos adicionales. En el anexo II se puede ver el código necesario para añadir estas propiedades adicionales.

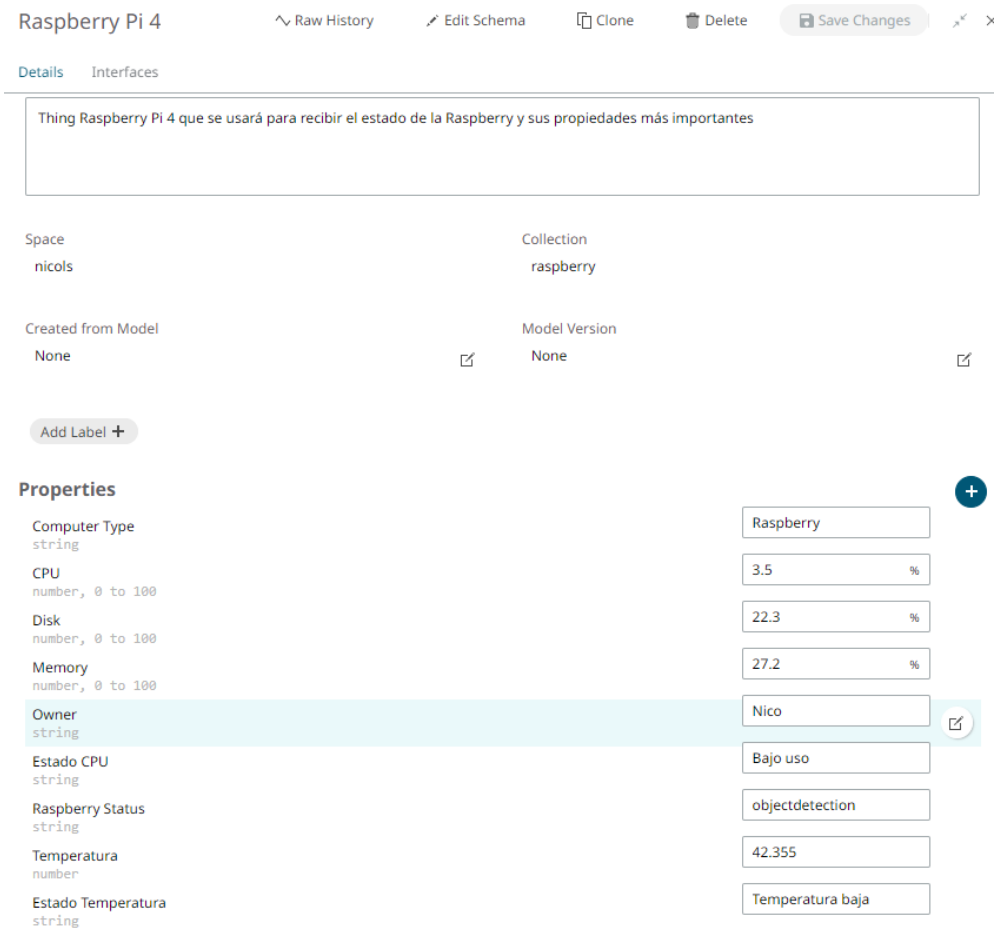


Ilustración 60: Propiedades del sistema de seguridad

Por otro lado, también se aprovecha la plataforma para generar gráficos con la información más relevante del sistema de seguridad. En Ilustración 61: Visualización general de propiedades de forma gráfica se pueden observar cuatro gráficos distintos. En el gráfico superior se muestran el número de accesos por día al domicilio. Para ello, se muestra por día el número de accesos y se recopila la información recibida por el sistema de seguridad para agruparlo. Para ello, se han tenido que obtener los datos de interés, generar unas tablas, seleccionar los datos de interés y establecer el tiempo. En el gráfico que se encuentra en medio del panel, se muestra el uso de la CPU y se ha realizado el mismo procedimiento de filtrado de datos mencionado anteriormente. Por otro lado, el gráfico que se encuentra en la parte inferior muestra la temperatura de la Raspberry Pi y para observar de forma más clara la temperatura en ese instante, se muestra el gráfico de la esquina inferior derecha.

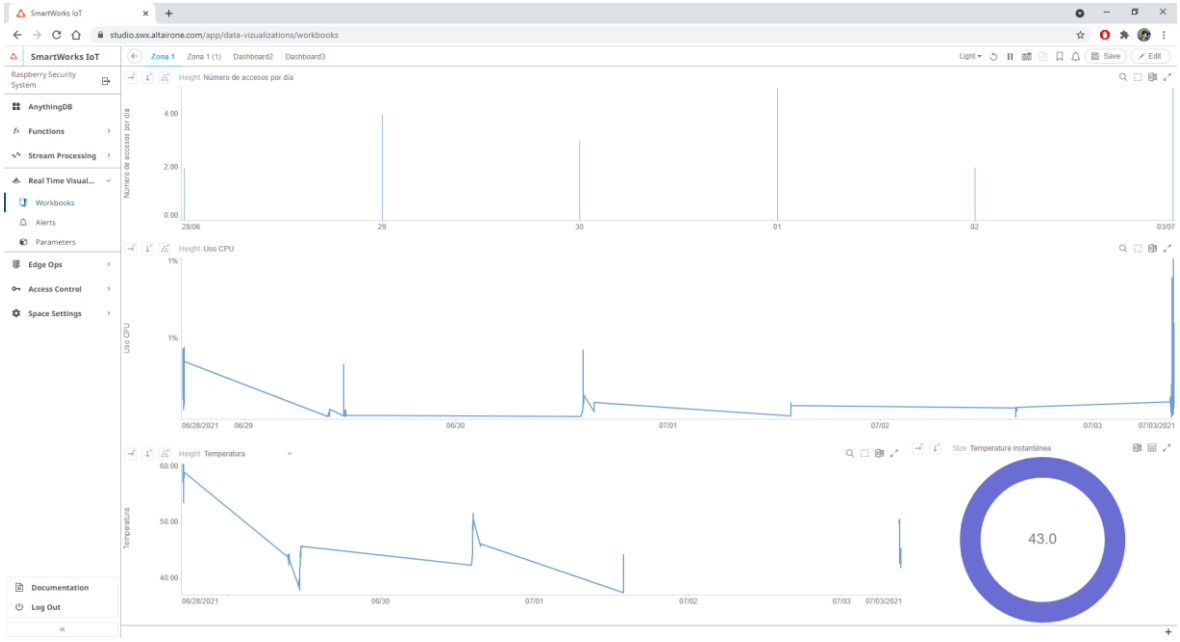


Ilustración 61: Visualización general de propiedades de forma gráfica

5.8 CARCASA 3D

Una vez programado el código y puesto en marcha cada uno de los periféricos que componen el sistema de seguridad, se ha procedido a diseñar una carcasa que pudiera integrar cada uno de estos componentes. Primero se realizó una búsqueda por Internet de posibles diseños, pero el uso del sistema de refrigeración, del altavoz y del lector de huellas hacen que este proyecto sea único y no haya ningún diseño previo similar. Es por ello que se ha tenido que diseñar desde cero toda la carcasa. Para el diseño de la carcasa se ha empleado SolidWorks y se han tenido en cuenta los siguientes factores:

- Ventilación
- Sustituibilidad de los componentes
- Posición de los componentes

Uno de los factores más importantes es la ventilación. Detectar objetos y caras de forma continua supone una gran carga computacional y provoca sobrecalentamientos. Debido a que el sistema de refrigeración cuenta con dos ventiladores, resulta necesario extraer

el aire caliente generado por la Raspberry Pi. Es por ello, que se han implementado agujeros en los laterales para permitir el flujo de aire. Por otro lado, debido a que la Raspberry está completamente cubierta de disipadores, para permitir la disipación de manera óptima se ha intentado dejar espacio libre entre el disipador y las paredes que sujetan la Raspberry.

Puesto que uno de los objetivos es integrar el máximo número de componentes en el menor espacio posible, se han colocado todos los componentes tal como se muestran en la imagen de a continuación. El altavoz va directamente conectado al puerto Jack y hay espacio para la disipación de la Raspberry. Aprovechando el espacio que genera el altavoz, se coloca en la parte inferior el lector de huellas.

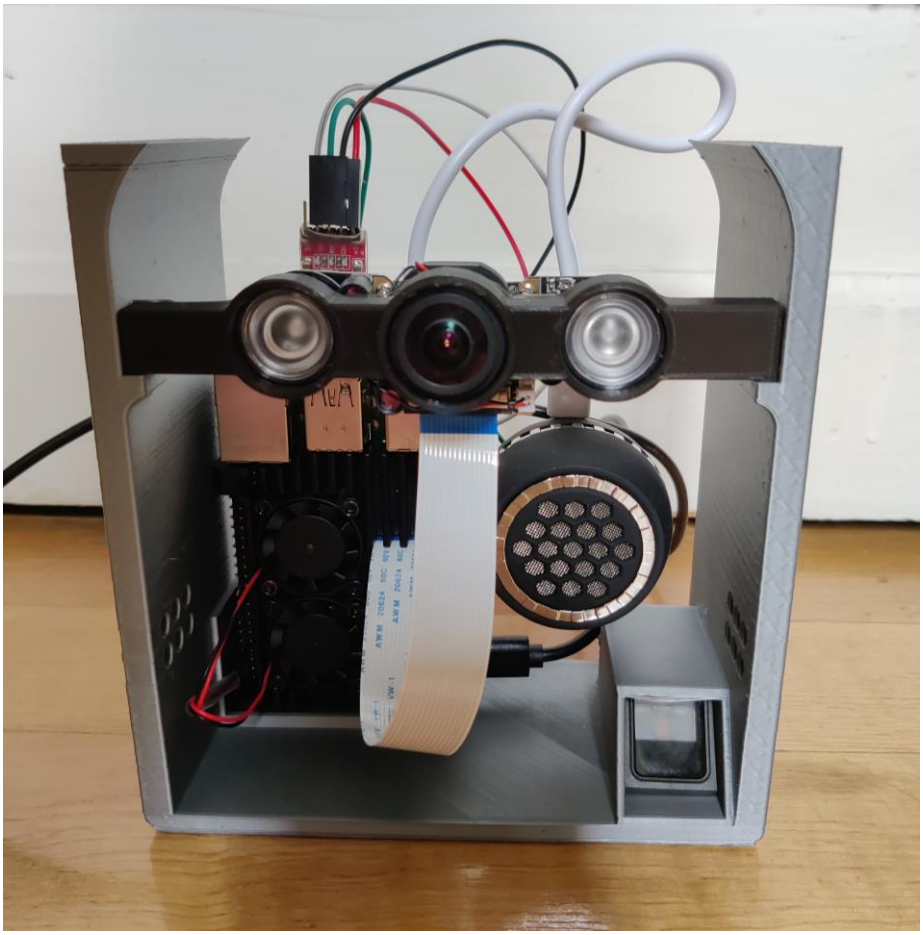


Ilustración 62: Sistema de seguridad montado

Cabe destacar que la carcasa anterior tiene el propósito de mostrar la colocación de los componentes en el interior de la carcasa, pero el diseño real es el que se muestra en Ilustración 64: Diseño final de la carcasa.

Por último, se ha querido pensar en la comodidad de los clientes a la hora de sustituir ciertos componentes del sistema. Para ello, todos los componentes se pueden extraer de forma sencilla.

Cabe destacar, que se han tenido que realizar distintos prototipos para poder ajustar al máximo posible todos los componentes. Por un lado, se han tomado todas las medidas necesarias y se han ido ajustando a la calidad de impresión de la impresora 3D. Además, se ha personalizado el diseño para tener en cuenta la microSD y los distintos cables de cada componente. A continuación, se muestra un diseño con la parte delantera transparente para poder apreciar el interior de la carcasa 3D.

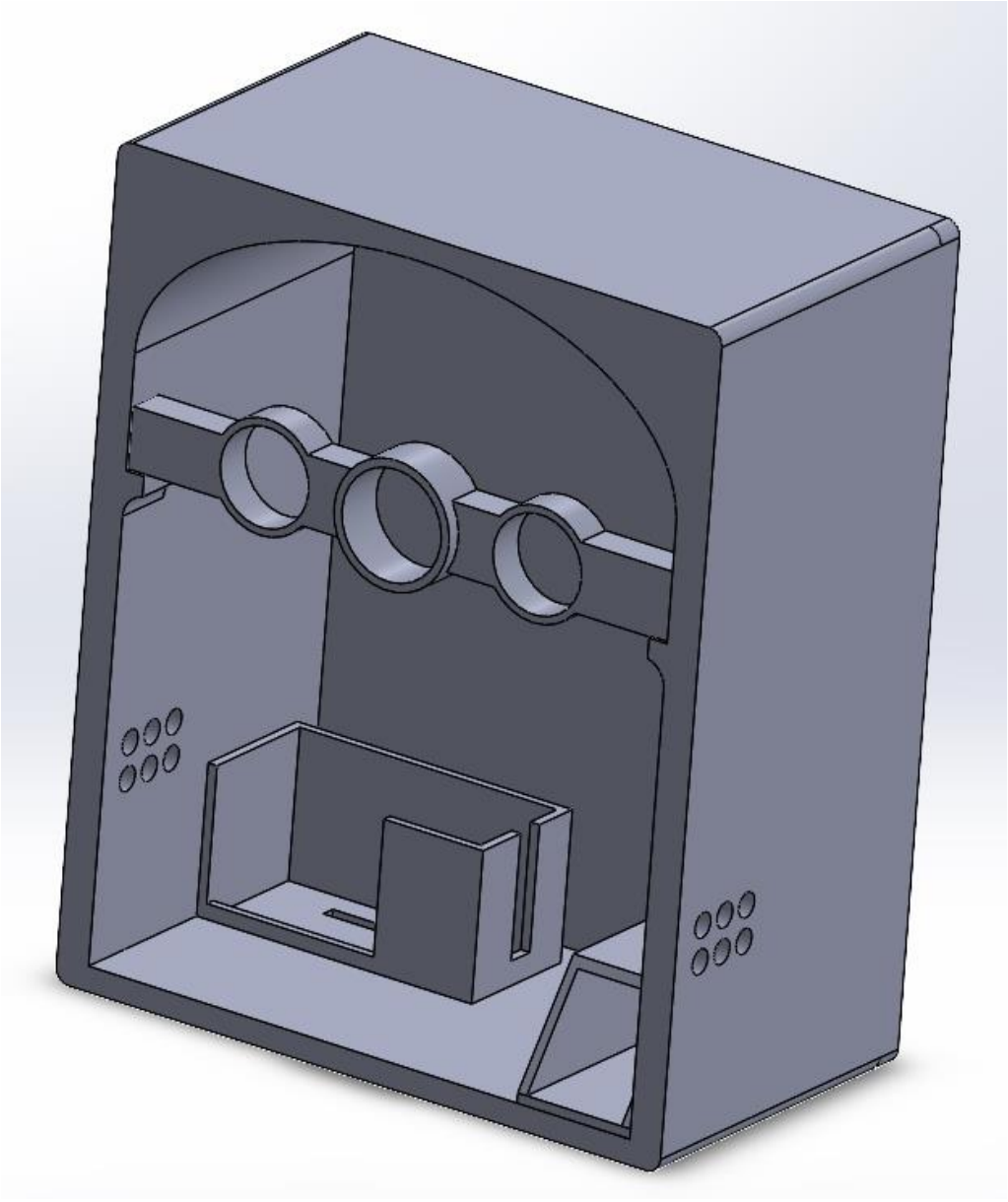


Ilustración 63: Diseño de la carcasa mostrada anteiormente en SolidWorks

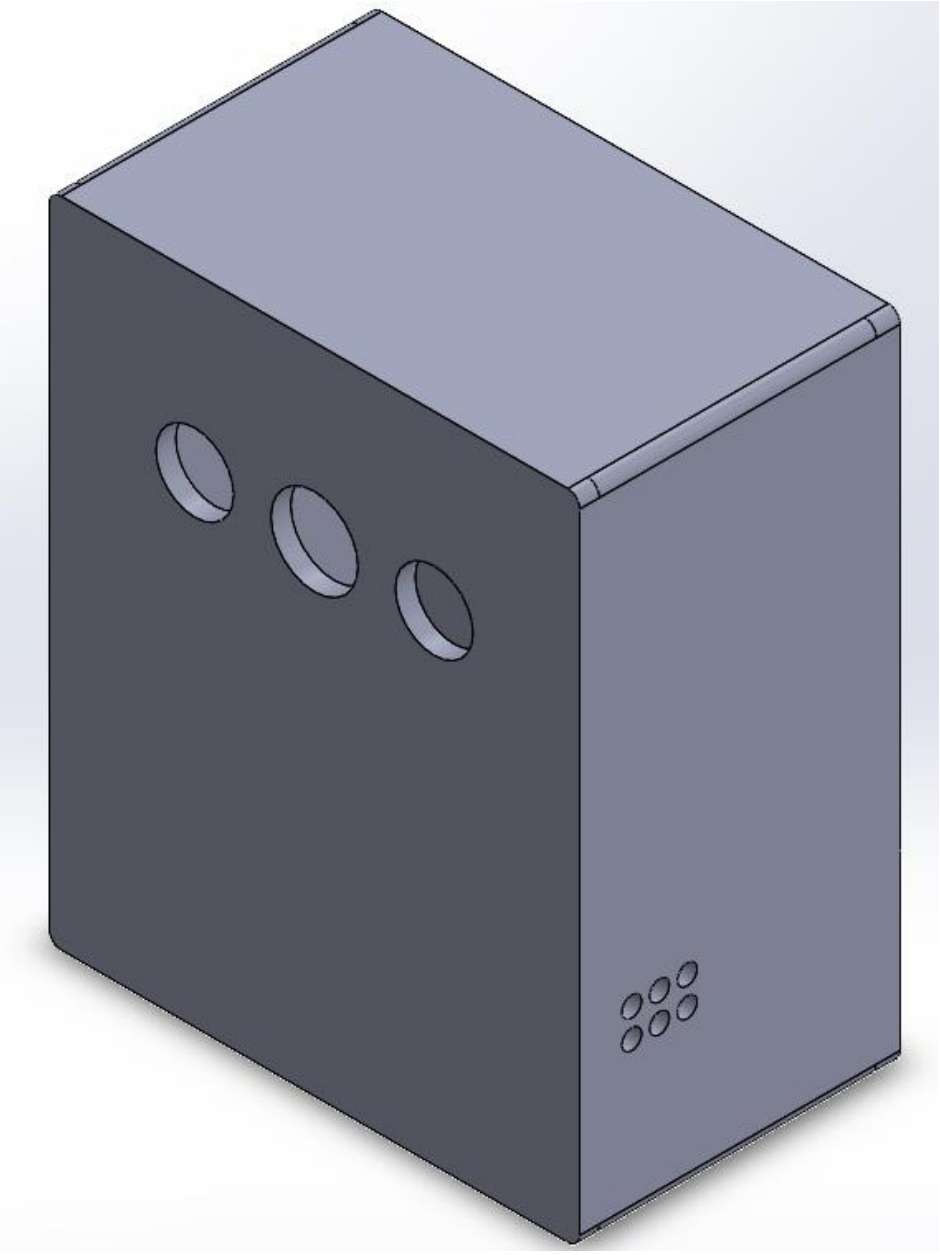


Ilustración 64: Diseño final de la carcasa

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se van a comentar los resultados más relevantes del proyecto.

6.1 REFRIGERACIÓN

Se trata de uno de los apartados que más ha mejorado a lo largo del proyecto. Las tareas requeridas por el sistema de seguridad son muy exigentes y requieren una gran capacidad de cálculo. Debido a ello, se producen sobrecalentamientos de la Raspberry Pi.

Inicialmente, la placa contaba con diversos disipadores situados sobre el procesador, el módulo de memoria RAM y otros chips de la Raspberry Pi. En reposo conseguían mantener la temperatura de la Raspberry Pi constante y una temperatura más reducida que sin disipadores. Con el sistema de seguridad en funcionamiento sufría serios sobrecalentamientos provocando la paralización del sistema siendo necesario apagar el dispositivo y volver a iniciarlo. La evolución de la temperatura se puede observar en la Ilustración 65: Evolución de la temperatura de la Raspberry Pi con disipadores.

Cabe destacar que estas medidas están realizadas con una herramienta de software que dispone Raspberry Pi OS e indica en tiempo real la temperatura del procesador de la Raspberry. También hay que mencionar que no se disponen de más datos debido al auto apagado de seguridad del sistema para evitar daños en el procesador.

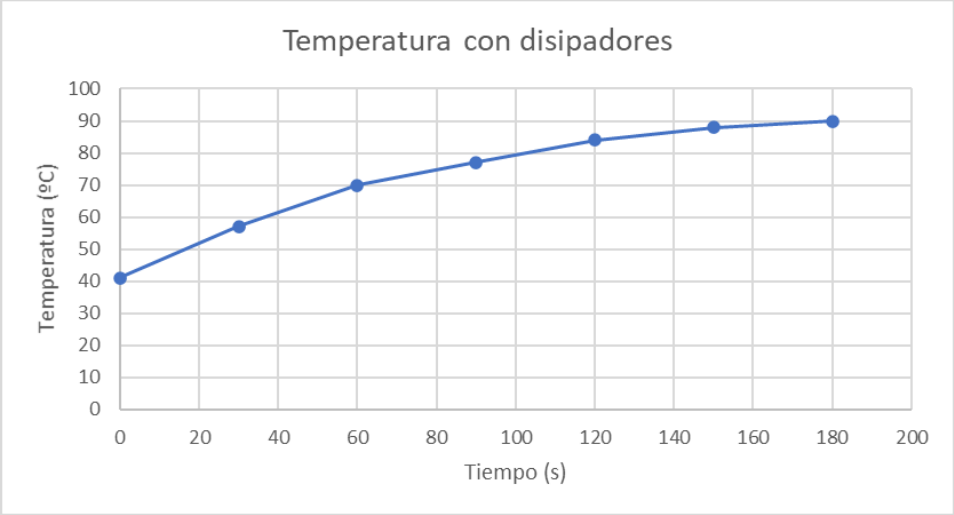


Ilustración 65: Evolución de la temperatura de la Raspberry Pi con disipadores

Tal como se comentó en Desarrollo, se pensó en reducir la carga de trabajo de la Raspberry añadiendo descansos entre análisis de imágenes mediante el uso de la función sleep. Esto permitía reducir levemente la temperatura en los "descansos", pero la temperatura seguía siendo muy elevada y se mantenía en torno a los 80º. Este valor de temperatura se hace inviable para un producto que debe estar funcionando de forma continua y reduce la durabilidad de este.

Es por ello, que se ha recurrido a un sistema de refrigeración como el de Ilustración 66: Sistema de refrigeración de la Raspberry.

Ilustración 66: Sistema de refrigeración de la Raspberry [36]

Este sistema, además de ser barato, permite distribuir el calor generado alrededor de toda la placa y disiparlo por aire. Por otro lado, los dos ventiladores facilitan la extracción del aire caliente consiguiendo los resultados que se muestran a continuación.

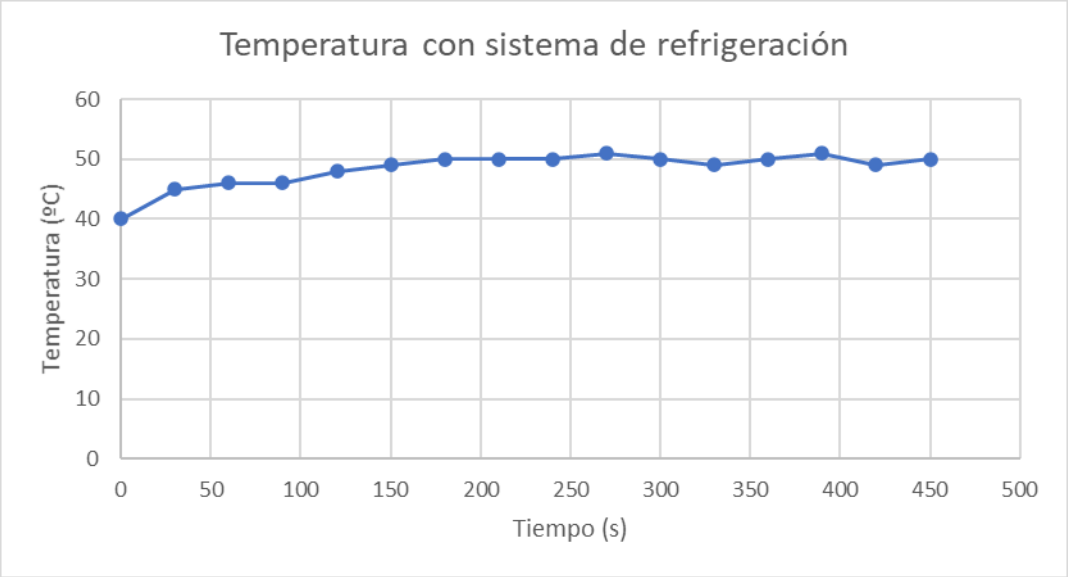


Ilustración 67: Evolución de la temperatura con sistema de refrigeración

Tal como se puede observar, la temperatura se mantiene estable en torno a los 50ºC y se concluye que es una solución eficaz. Esta solución permite aumentar la durabilidad del dispositivo y evitar posibles cuelgues por sobrecalentamientos.

6.2 CONSUMO ENERGÉTICO

En esta parte se van a analizar varios aspectos relacionados con el consumo energético del sistema. En primer lugar, se va a analizar el consumo del sistema de seguridad y se va a comparar con alternativas del mercado.

Una de las grandes ventajas de este sistema, es que todos los componentes están integrados y centralizados en torno a la Raspberry Pi. Esto reduce el número de aparatos de alimentación necesarios y conlleva un gran ahorro energético. En la tabla que se muestra a continuación se muestran los consumos medios en función del estado del sistema de seguridad. Hay que

resaltar que todos los periféricos están alimentados por la Raspberry Pi y no necesitan de alimentación externa.

Consumo medio	Detección de objetos	Detección facial	Reconocimiento de huellas dactilares
9,6W	9,7W	9,5W	9,4W

Tabla 4: Consumo medio en cada estado del sistema de seguridad

Cabe destacar que los consumos mostrados en la tabla 4 son del conjunto del sistema de seguridad. Se tratan de los consumos medios en cada uno de los estados en los que se puede encontrar el sistema de seguridad. Cuando el sistema se encuentra detectando objetos puede presentar picos de 10,4W, pero su consumo medio se encuentra en torno a los 9,7W. Por otro lado, el consumo en la detección facial es ligeramente inferior y presenta un consumo medio de 9,5W teniendo picos de 10,1W. Por último, cuando el sistema se encuentra reconociendo huellas dactilares tiene un consumo medio de 9,4W.

Fijándose en sistemas de seguridad que cuentan con detección facial y reconocimiento de huellas dactilares de la competencia como es el caso de Hikvision [24], se observa en su ficha técnica que presentan consumos de 20-24W. Mirando productos que únicamente cuentan con detección facial, presentan consumos similares. Se puede concluir que este sistema de seguridad reduce a más de la mitad el consumo energético y supone un ahorro muy importante tanto a nivel energético como a nivel económico.

Si se supone un consumo medio del sistema de seguridad de unos 10W y del resto de sistemas de la competencia de unos 20W se pueden obtener unos ahorros muy importantes. A pesar de que el precio de la electricidad está subiendo notablemente en los últimos años, se va a estimar un coste medio de 0,3€/KWh y haciendo el siguiente cálculo:

$$Ahorro\ anual = \frac{0{,}3€}{KW * h} * \frac{1KW}{1000W} * 10W * 24h * 365días = 26{,}28€$$

Otro aspecto que se ha tenido en cuenta a la hora de programar el envío de datos a SmartWorks, es el tipo de protocolo empleado. Se ha observado que el envío de información por HTTP requiere de una mayor capacidad de cálculo y que reduce drásticamente la velocidad de detección de objetos. Por otro lado, se ha visto que presenta unos picos de consumo mayores. Debido a ello, se emplea el protocolo de red MQTT, el cual presenta un rendimiento mayor.

6.3 RECONOCIMIENTO DE OBJETOS

Se trata de uno de los ejes principales del sistema de seguridad y se han obtenido unos resultados muy satisfactorios. No presenta ningún fallo en la Raspberry, aunque en PyCharm aparece un warning que no se ha conseguido eliminar. A pesar de ello, funciona perfectamente en PyCharm y ofrece los mismos resultados que en la Raspberry Pi.

A pesar de emplear la arquitectura más básica de YOLO, el sistema es capaz de reconocer a la perfección los objetos cuando se ven con suficiente claridad y a una distancia razonable. Por otro lado, con las pruebas que se han realizado, no se han visto errores a la hora de señalar un objeto u otro. Cabe destacar que también reconoce varios objetos a la vez y no se producen conflictos entre ellos. Adicionalmente, se ha probado con la luz apagada para probar la visión nocturna de la cámara y se ha visto que no afecta en el rendimiento a la hora de reconocer objetos.

Una de las grandes ventajas de disponer de este algoritmo en el sistema, es que es capaz de detectar varios objetos a la vez sin afectar su rendimiento. Dispositivos como los comentados en secciones anteriores solo son capaces de detectar facialmente. Es por ello, que este sistema ofrece un valor añadido porque es capaz de detectar animales u objetos ofreciendo de esta forma un sistema más completo.

6.4 *RECONOCIMIENTO FACIAL*

Este sistema de reconocimiento no funciona tan bien como YOLO, pero ofrece unos buenos resultados. Cabe destacar que, para detectar a la persona, ésta se tiene que situar cerca de la cámara debido a que se ha programado de tal forma que reduce la resolución de la imagen para aumentar la velocidad del análisis. Por otro lado, sí se ha visto que en algunos casos reconoce una persona que no es, pero por norma general funciona bastante bien. Hay que destacar que es más rápido reconociendo a una persona que YOLO cuando solo aparece la cara en la imagen. Por último, hay que comentar que en algunos casos muy concretos sufre un error recibiendo la imagen y se paraliza la detección. Se ha intentado indagar sobre ello, pero debido a que es un problema que principalmente ocurre en PyCharm y en la Raspberry ha ocurrido en contadas ocasiones, se ha visto que funciona correctamente. Al igual que con la detección de objetos, se ha probado bajo condiciones de escasa luminosidad y se ha comprobado que el rendimiento no se ve afectado y detecta de la misma manera que con buenas condiciones de luz.

6.5 *DETECCIÓN DE HUELLAS DACTILARES*

Se trata de uno de los elementos esenciales del sistema de seguridad debido a que ofrece una capa de seguridad adicional. Es por ello, que su fiabilidad debe ser alta y se ha comprobado que no sufre ningún tipo de error. Se ha visto que no confunde huellas de otras personas y tampoco dedos de la misma persona. Cabe destacar, que, en algunos casos, cuando no se coloca el dedo de la misma forma que como se registró, no reconoce la huella y se tiene que colocar el dedo nuevamente. Por otro lado, hay que mencionar que la detección es rápida y emplea un segundo aproximadamente.

6.6 *ELEMENTOS ADICIONALES DEL SISTEMA*

El altavoz funciona correctamente y no ha supuesto un gran quebradero de cabeza debido a que la Raspberry cuenta con un puerto Jack. Lo que ha resultado más complicado ha sido

reproducir los archivos de audio con una librería, pero tras indagar se ha conseguido emplear uno que funcione correctamente.

Por otro lado, debido a que la Raspberry cuenta con un puerto específicamente diseñado para una cámara, no ha supuesto ninguna complicación. Este puerto ofrece la alimentación de la cámara y la conexión con la Raspberry. En todos estos meses de pruebas no ha fallado en ningún momento.

Por último, la librería de envío por correo electrónico funciona a la perfección y nunca ha tenido un problema a la hora de enviar un correo.

6.7 SMARTWORKS

La conexión entre el sistema de seguridad y la plataforma SmartWorks funciona correctamente. SmartWorks recibe todos los datos enviados por el sistema y muestra la información relevante en la sección de propiedades y en los gráficos. Por otro lado, las funciones que analizan los datos recibidos también funcionan correctamente y muestran en la sección de propiedades las conclusiones obtenidas. Por último, los gráficos con el uso de la CPU, número de accesos diarios y temperatura también se han implementado de forma satisfactoria y muestran los datos correctamente. De esta forma, se ha conseguido facilitar aún más al usuario final la visualización de la información más relevante del estado del sistema de seguridad.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

En este proyecto se ha conseguido alcanzar todos los objetivos planteados. Además, se han podido añadir ciertas funcionalidades que no estaban pensadas en el inicio del proyecto. Se ha logrado diseñar un producto funcional con un reducido coste, con capacidad de detección de objetos y reconocimiento de caras y con una capa de seguridad adicional mediante el uso de un lector de huellas. Además de estos, cuenta con varias funciones adicionales y es mucho más eficiente a nivel energético que productos de la competencia.

7.1.1 FUNCIONAMIENTO DE LOS ALGORITMOS

Uno de los ejes principales del proyecto han sido la detección de objetos y el reconocimiento facial. Tal como se ha podido ver en las secciones anteriores, se ha logrado alcanzar estos objetivos. Ha sido una tarea complicada lograr que funcionen en un dispositivo como la Raspberry Pi debido a su limitada capacidad de procesamiento, pero gracias a la búsqueda de los algoritmos más eficientes y la optimización de algunos de ellos, se ha conseguido que funcionen de forma satisfactoria en este dispositivo. Esto último ha supuesto la búsqueda de los mejores algoritmos y plataformas donde ejecutarlos además de implementar los dos juntos. Por otro lado, la programación de estos algoritmos ha requerido el aprendizaje del lenguaje de programación Python y de los frameworks en los que se ejecutan.

7.1.2 LECTOR DE HUELLAS

Otro aspecto muy importante del proyecto es el reconocimiento de huellas dactilares. Este elemento supone una capa adicional de seguridad en el sistema. Ha supuesto aprender qué software usar para ponerlo en marcha y también cómo conectarlo a la Raspberry Pi. Adicionalmente, se ha tenido que programar para que funcione en momentos concretos. Se ha logrado que funcione correctamente y con una gran precisión.

7.1.3 AVISOS POR CORREO ELECTRÓNICO

Se ha logrado que el sistema de seguridad envíe correos electrónicos con los nombres de las personas que acceden e imágenes de las personas detectadas de forma satisfactoria. Para ello se ha tenido que aprender a manejar dos librerías. Por otro lado, se ha tenido que pensar en qué momentos se deben enviar los correos y qué contenido deben tener para no enviar información irrelevante.

7.1.4 ALTAVOZ

A pesar de ser teóricamente uno de los elementos más sencillos de implementar gracias a la presencia de un puerto Jack en la Raspberry, ha dado ciertos quebraderos de cabeza. Se ha tenido que encontrar una librería con capacidad de generar archivos de audio con voz de alta calidad. Por otro lado, se ha tenido que buscar una librería capaz de reproducir estos archivos en el momento idóneo. Esto último ha sido complicado porque la mayoría de las librerías probadas no funcionaban en la Raspberry. Por último, también se ha tenido que encontrar una forma de reproducir estos archivos de audio sin paralizar el sistema de seguridad. Esto último también se ha conseguido solucionar gracias al empleo de multithreading. A pesar de todos los problemas surgidos se ha conseguido implementar de forma satisfactoria este componente del sistema y permite ayudar al usuario final cómo configurar el sistema e indicar los pasos a seguir a la hora de permitir el acceso a la vivienda.

7.1.5 FLASK

Se trata de un elemento del sistema que aporta un gran valor añadido ya que permite visualizar en tiempo real, a través de Internet, la imagen de la cámara de seguridad. Ha sido una tarea complicada implementar este elemento debido a que tiene que funcionar paralelamente. Gracias al uso de multithreading se ha logrado implementar sin paralizar el análisis de la detección de objetos y caras. Además de mostrarse la imagen de la cámara, se ha conseguido mostrar lo analizado por la Raspberry, pudiéndose ver los recuadros de las personas y objetos detectados.

7.1.6 INTEGRACIÓN DE LOS DISTINTOS COMPONENTES DEL SISTEMA

Uno de los objetivos principales de este proyecto era integrar en un mismo sistema distintas capas de seguridad debido a que en el mercado no suelen venir detección facial y reconocimiento de huellas juntos. El uso de una Raspberry Pi ha permitido integrar ambos elementos, además de ofrecer diversas funcionalidades adicionales.

Por otro lado, tal como se comentó en la sección anterior, se tenía como objetivo implementar un sistema durable en el tiempo. Para ello, ha resultado necesario encontrar la forma de mantener las temperaturas del sistema de seguridad a unos valores razonables. Para ello, se ha tenido que pensar en distintas alternativas tanto a nivel de software como de hardware. Finalmente se ha optado por un sistema de refrigeración capaz de mantener la temperatura a unos niveles razonables eliminando la necesidad de reducir el rendimiento a nivel de software. Cabe destacar, que en el diseño de la carcasa se ha tenido en cuenta este último aspecto además de pensar en la mejor forma para poder reemplazar ciertos componentes en caso de fallo.

7.1.7 FACILITAR LA CONFIGURACIÓN DEL SISTEMA

Debido a que el sistema está destinado a usuarios de todo tipo entre los que se pueden encontrar personas sin conocimientos de programación, se ha intentado facilitar el proceso de configuración del sistema de seguridad. Es por ello, que al ejecutar el código se muestran menús claros y con la ayuda del altavoz se facilita aún más el proceso.

7.1.8 SMARTWORKS

Con el objetivo de añadir funcionalidades adicionales al usuario final, se ha logrado conectar la Raspberry Pi con la plataforma SmartWorks. Se ha conseguido que el usuario disponga de una plataforma donde visualizar la información más relevante del estado del sistema de seguridad. Además de observar los datos, se pueden mirar gráficamente gracias al uso de gráficos que representan la información más importante.

7.1.9 AHORRO ENERGÉTICO

Tal como se pudo ver en el Análisis de Resultados se ha logrado ofrecer un producto barato que también es más eficiente. Se ha logrado diseñar un producto que consume la mitad de energía que productos similares de la competencia.

7.2 TRABAJOS FUTUROS Y MEJORAS AL PROYECTO

7.2.1 OPTIMIZACIÓN DE LOS ALGORITMOS

A pesar de ya usar los algoritmos más eficaces a nivel computacional, estos se pueden optimizar aún más entrenando las redes neuronales. El algoritmo YOLO se puede entrenar con varias herramientas, pero diseñar un programa para que el usuario final pueda hacerlo de forma sencilla es bastante complicado. Al igual que con el algoritmo YOLO, esto también es aplicable a los algoritmos de reconocimiento facial. La idea sería solicitar, en lugar de pedir una imagen de la persona a detectar tal como se realiza ahora, varias imágenes del usuario para posteriormente enviarlas a una herramienta de software que entrene las redes neuronales que se emplean para el reconocimiento. Sería realizar todo este proceso de forma automática sin que el usuario necesite realizar ninguna acción adicional.

Por otro lado, también se ha pensado en reducir el número de clases del algoritmo YOLO para reducir el tamaño de la red neuronal y de esta forma poder usar arquitecturas YOLO más complejas y de esta forma conseguir una mayor precisión en la detección de objetos.

7.2.2 AVISOS POR OTROS MEDIOS

Los avisos por correo electrónico son una buena forma de comunicar el acceso de personas al domicilio, pero podría ser más interesante recibir un SMS. Existen ya algunas librerías que permiten realizar esto y sería interesante implementar esta función en futuros proyectos. Además, también se ha pensado en notificar a través de llamadas o aplicaciones específicamente diseñadas para este sistema de seguridad.

7.2.3 FACILITAR LA INTERACCIÓN DEL USUARIO CON EL SISTEMA

En este proyecto se ha intentado facilitar al máximo posible la interacción del usuario con el sistema de seguridad. Para ello, se ha empleado un altavoz para indicar los pasos a seguir, además de programar el código de tal forma que el usuario pueda realizar todas las acciones de forma sencilla. En futuros proyectos, el objetivo sería crear un archivo ejecutable para evitar al usuario entrar en una ventana de comandos y ejecutar el programa. Además, debería presentar una interfaz más sencilla de usar.

Por otro lado, la conexión de la Raspberry Pi con la red Wifi de la vivienda es vital y supone la mayor barrera para poder configurar el sistema. La idea sería encontrar alguna forma de facilitar el proceso de conexión con el sistema de seguridad.

7.2.4 CREACIÓN DE UNA APLICACIÓN MÓVIL

Teniendo en mente el incremento de uso de los smartphones en los últimos años, sería interesante crear una aplicación móvil tanto para iOS como para Android. Esta aplicación debería ofrecer la posibilidad de ver en tiempo real el estado del sistema de seguridad además de imagen en tiempo real de la cámara. Por otro lado, también podría ofrecer estadísticas de accesos, personas que han accedido y más propiedades interesantes.

Adicionalmente, también sería interesante realizar la configuración del sistema de seguridad a través de la aplicación, conectándose primero a la Raspberry para ofrecer los datos de la red Wifi a la que conectarse y de esta forma eliminar el difícil proceso inicial de conexión a la Raspberry Pi. Por otro lado, también debería permitir configurar las imágenes empleadas de los usuarios que se van a registrar junto con las demás configuraciones que se tienen que realizar para poner en marcha el sistema. Por otro lado, también se podría emplear para recibir los avisos por móvil de accesos de una persona. Por un lado, una notificación de la persona que ha accedido y, por otro, una imagen de la persona. Adicionalmente, ofrecer la posibilidad de acceder a distintos sistemas de seguridad que pueda poseer el usuario y ver de forma centralizada el estado de todos ellos.

7.2.5 APERTURA DE LA PUERTA

Uno de los mayores problemas para implementar de forma satisfactoria este proyecto es conseguir tener la puerta del domicilio conectada al sistema de seguridad. El objetivo sería diseñar una forma sencilla de conectar la puerta con el sistema. Por otro lado, también pensar en cómo realizar esta implementación. Si se debiesen emplear relés u otros métodos.

7.2.6 LECTOR DE TARJETAS

Este proyecto cuenta ya con una buena capa de seguridad que hace prácticamente imposible el acceso de una persona desconocida, pero sería interesante añadir una capa adicional mediante el uso de un lector de tarjetas. Esta tarjeta sería única, difícil de hackear y tendría un funcionamiento similar al de las llaves utilizadas para los coches.

7.2.7 MEJORA DEL DISEÑO DE LA CARCASA

Por último, sería interesante optimizar el diseño de la carcasa. Por un lado, se plantearía el uso de ventiladores adicionales para mejorar el flujo de aire en el interior de la carcasa. Por otro lado, facilitar aún más el reemplazo de los componentes del sistema de seguridad mediante la implementación de pestañas ingeniosamente diseñadas. Adicionalmente, diseñar una carcasa más atractiva con la capacidad de almacenar el correo y de esta forma sustituir una caja por otra y ocupar menos espacio.

Capítulo 8. BIBLIOGRAFÍA

[1] Geitgey, A. Machine learning is fun. Part 4. Último acceso: 02/02/2021. Mayo, 2014. https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471 Weng, L. Object Detection for Dummies Part 1: Gradient Vector, HOG, and SS. Último acceso: 02/02/2021. Octubre, 2017 https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html/

[2] Mallick, S. Histogram of Oriented Gradients explained using OpenCV. Último acceso: 02/02/2021. Diciembre, 2016. https://learnopencv.com/histogram-of-oriented-gradients/

[3] Rosenbrock, A. Facial landmarks with dlib, OpenCV and Python. Último acceso: 03/02/2021. Abril, 2017. https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[4] Manishgupta. YOLO – You only look once, real time object detection explained. Último acceso: 04/02/2021. Agosto, 2017. https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4

[5] Yolo Official Webpage. Último acceso 14/04/2021. https://pjreddie.com/darknet/yolo/

[6] Redmon, J. An incremental Improvement. Último acceso 17/02/2021. https://pjreddie.com/media/files/papers/YOLOv3.pdf

[7] Sadli, R. The beginner´s guide to implementing YOLOv3 in TensorFlow 2.0 (part 1). Último acceso: 09/02/2021. Diciembre, 2019. https://machinelearningspace.com/yolov3-tensorflow-2-part-1/

[8] Mantripragada, M. Digging deep into YOLO V3. Último acceso: 10/02/2021. Agosto, 2020. https://towardsdatascience.com/digging-deep-into-yolo-v3-a-hands-on-guide-part-1-78681f2c7e29

[9] Brownlee, J. A gentle introduction to Pooling layers for convolutional neural networks. Último acceso: 12/02/2021. Abril, 2019. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

[10] Rosenbrock, A. Face detection with OpenCV and deep learning. Último acceso: 12/02/2021. Febrero, 2018 https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/

[11] Geitgey, A. Face recognition algorithm. Último acceso: 08/02/2021. Abril, 2018. https://github.com/ageitgey/face_recognition

[12] Dlib Official Webpage. Último acceso 20/05/2021. http://dlib.net/

[13] Barrios, J. Redes Neuronales Convolucionales. Último acceso 21/03/2021. https://www.juanbarrios.com/redes-neurales-convolucionales/

[14] Leroy Merlin Cámaras IP. Último acceso: 25/06/2021. https://www.leroymerlin.es/electricidad/alarmas-camaras-ip-detectores/vigilancia-camaras-ip?facet=%5B%7B%22c%22%3A%2214f7540e-d4b8-4857-b3c5-0c25982b26cc_Opus_Criterion%22%2C%22ft%22%3A%22s%22%2C%22f%22%3A%5B%22aa1dbd2b-2a4e-417d-90bd-bbfa6ea65fdf_Opus_Segment%22%5D%7D%5D&sort=default&gt=4-col&offset=0

[15] Hikvision Official page. Hikvision DS-K1T331. Último acceso: 25/06/2021. https://www.hikvision.com/es-la/products/Access-Control-Products/Face-Recognition-Terminals/Value-Series/ds-k1t331/

[16] Reolink Official page. Reolink RLK8-420D4. Último acceso: 25/06/2021. https://reolink.com/es/product/rlk8420d4/?attribute_pa_version=4mp&gclid=Cj0KCQjw6-SDBhCMARIsAGbI7UjZqpGr1t5Ptf79C4s4AbxSiOUT9ZkLZ3FhYdJtQHt6cdK29wdWQaAgWWEALw_wcB

[17] Hikvision Official Page. Hikvision DS-K1T607MFW. Último acceso: 25/06/2021. https://www.hikvision.com/en/products/Access-Control-Products/Face-Recognition-Terminals/Pro-Series/ds-k1t607mfw/

[18] Axis Official Page. Axis Commucations. Último acceso: 25/06/2021. https://www.axis.com/es-es/products/network-cameras

[19] Marshall Oficial page. Marshall Electronics. Último acceso: 25/06/2021. http://www.marshall-usa.com/

[20] Hui, J. Object Detection: speed and accuracy comparison. Último acceso: 18/02/2021. Mayo, 2018. https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

[21] Universidad de Sevilla. Performance Analysis of Real-Time DNN Inference on Raspberry Pi. Último acceso: 19/03/2021. https://digital.csic.es/bitstream/10261/163973/1/Performance_Analysis_of_Real_Time_DNN_on_RPi.pdf

[22] Redmon, J. pjreddie webpage. Último acceso: 14/04/2021. https://pjreddie.com/darknet/yolo/

[23] Kathuria, A. What´s new in YOLO v3? Último acceso: 14/02/2021. Abril, 2018. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

[24] Hikvision Official Page. Último acceso: 27/06/2021. https://www.hikvision.com/es/products/Access-Control-Products/Face-Recognition-Terminals/Pro-Series/ds-k1t671mf/

[25] Wikipedia. Histogram of oriented gradients. Último acceso: 04/03/2021. https://de.wikipedia.org/wiki/Histogram_of_oriented_gradients

[26] Llamas, L. ¿Qué es MQTT? Su importancia como protocolo IoT. Último acceso: 27/06/2021. Abril, 2019. https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[27] Naciones Unidas. Objetivos de Desarrollo sostenible. Último acceso: 28/06/2021. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[28] Fang, W. The network structure of Tiny-YOLO-V3. Último acceso: 20/03/2021. Diciembre, 2019. https://www.researchgate.net/figure/The-network-structure-of-Tiny-YOLO-V3_fig1_338162578

[29] AliOS. Raspberry Pi 4 GPIO Pinout. Último acceso: 04/03/2021 Noviembre 2019. https://keytosmart.com/single-board-computers/raspberry-pi-4-gpio-pinout/

[30] Raspberry Pi Official Webpage. Documentation. Último acceso: 19/01/2021 https://www.raspberrypi.org/documentation/usage/gpio/

[31] Aliexpress. Cámara de visión nocturna. Último acceso: 20/01/2021. https://es.aliexpress.com/item/32880128406.html?spm=a2g0o.productlist.0.0.33903c69hTPjQ7&algo_pvid=c21d7111-b273-4b8b-8584-a91351395d48&algo_exp_id=c21d7111-b273-4b8b-8584-a91351395d48-3

[32] Ebay. CP2101. Módulo USB a TTL UART. Último acceso: 20/03/2021. https://www.ebay.es/itm/232516584694

[33] Ianik. CP2102 usb to ttl copy. Último acceso: 20/03/2021. https://easyeda.com/lanik/CP2102_usb_to_ttl-7Ttrp82ke

[34] ZFM-70 User Manual. Último acceso: 09/04/2021. https://www.velleman.eu/downloads/29/infosheets/vma329_datasheet.pdf

[35] Amazon. ZFM-70 Sensor de huellas. Último acceso: 09/04/2021

https://www.amazon.com/VELLEMAN-VMA329-FINGERPRINT-SENSOR-ZFM-708/dp/B07CSD8J32

[36] Aliexpress. Para Raspberry Pi 4/3 caja de aluminio con ventilador de refrigeración Dual Carcasa de metal. Último acceso: 20/03/2021. https://es.aliexpress.com/item/1005002912919705.html?spm=a2g0o.productlist.0.0.4d22f51bbdAYMj&algo_pvid=13d2631e-3d58-4f5e-97a2-f322e59796c3&algo_exp_id=13d2631e-3d58-4f5e-97a2-f322e59796c3-9

[37] Amazon. Color You Altavoz portátil. Último acceso: 01/04/2021. https://www.amazon.es/gp/product/B07QYFFNK5/ref=ppx_yo_dt_b_search_asin_image?ie=UTF8&psc=1

[38] Pccomponentes. Joy It RB Heatsink Set De Disipador de calor para raspberry Pi Modell 2 Banana Pi PcDuino. Último acceso: 02/02/2021. https://www.pccomponentes.com/joy-it-rb-heatsink-set-de-disipador-de-calor-para-raspberry-pi-modell-2-banana-pi-pcduino

ANEXO I: INSTALACIÓN DE PROGRAMAS

PYTHON 3.7

Python 3.7 es la versión que se emplea durante todo el proyecto. Supone una de las piezas centrales del proyecto porque es el lenguaje de programación que va a emplear tanto a la hora de programar en PyCharm como en la Raspberry Pi. Se usa la versión 3.7 porque es la que menos conflictos genera con OpenCV. A pesar de usar la versión 3.7, también se han realizado pruebas con versiones posteriores y aparentemente no han causado problemas, pero para asegurar el correcto funcionamiento del programa es mejor instalar esta versión. Tal como se verá más adelante, Python es multiplataforma y se puede instalar en diferentes sistemas operativos. Entre ellos se encuentran Windows, Mac y Linux.

Cabe destacar que se debe tener instalado Python tanto en la Raspberry Pi como en el ordenador. Puesto que en la versión de Raspberry Pi OS viene instalado por defecto, no resulta necesario realizar la instalación de Python. Sin embargo, para poder usar PyCharm es necesario instalarlo y es lo que se va a mostrar a continuación.

Para poder instalar Python, se accede a la página oficial y se accede a la sección de descargas (https://www.python.org/downloads/). Cabe destacar que por defecto las descargas serán para instalar Python en Windows y en el caso que se requiera instalar en Linux o Mac se tendrá que pinchar, en el mismo enlace mostrado anteriormente, sobre Linux/Unix o Mac OS X. Para descargar la versión 3.7 se hace scroll hacia abajo donde se verá una sección de versiones específicas de Python y se pinchará en download. En el caso que se quiera instalar la última versión de Python simplemente hay que pinchar sobre Download the latest version of Windows y se obtendrá un archivo ejecutable .exe.

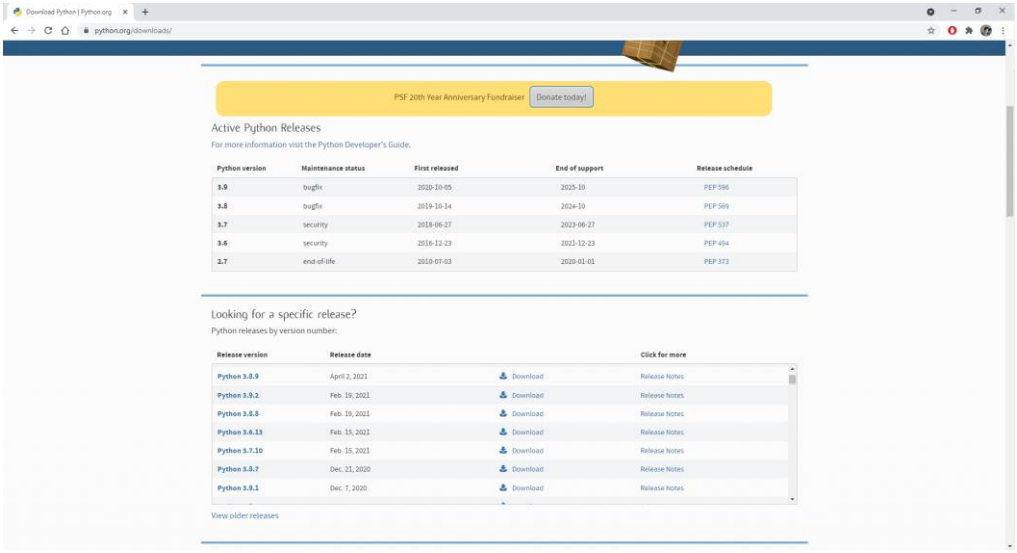


Ilustración 68: Página de descargas de Python

Una vez descargado el archivo ejecutable, se ejecuta y se elige la instalación por defecto, se elige la ruta en la que se desea instalar Python y aceptando algunos términos y condiciones se consigue instalar rápidamente.

PYCHARM COMMNUNITY EDITION

Es el entorno de desarrollo de Python empleado para programar en el ordenador. Se podría haber programado directamente en la Raspberry, pero resulta más cómodo hacerlo desde el ordenador. En este caso, no es necesario instalar una versión específica y se instala la última versión disponible. Para ello hay que dirigirse a la página oficial de JetBrains (empresa que ha creado el IDE) mediante este enlace https://www.jetbrains.com/es-es/pycharm/download/#section=windows. Cuando se accede a este enlace hay que pulsar sobre el botón de descarga para la versión Community. Se trata de la versión gratuita y ofrece todas las herramientas necesarias para programar en Python.

En los casos que se requiera instalar en otro sistema operativo, simplemente se pulsa sobre la pestaña correspondiente al sistema operativo de interés y se procede a la descarga del programa.

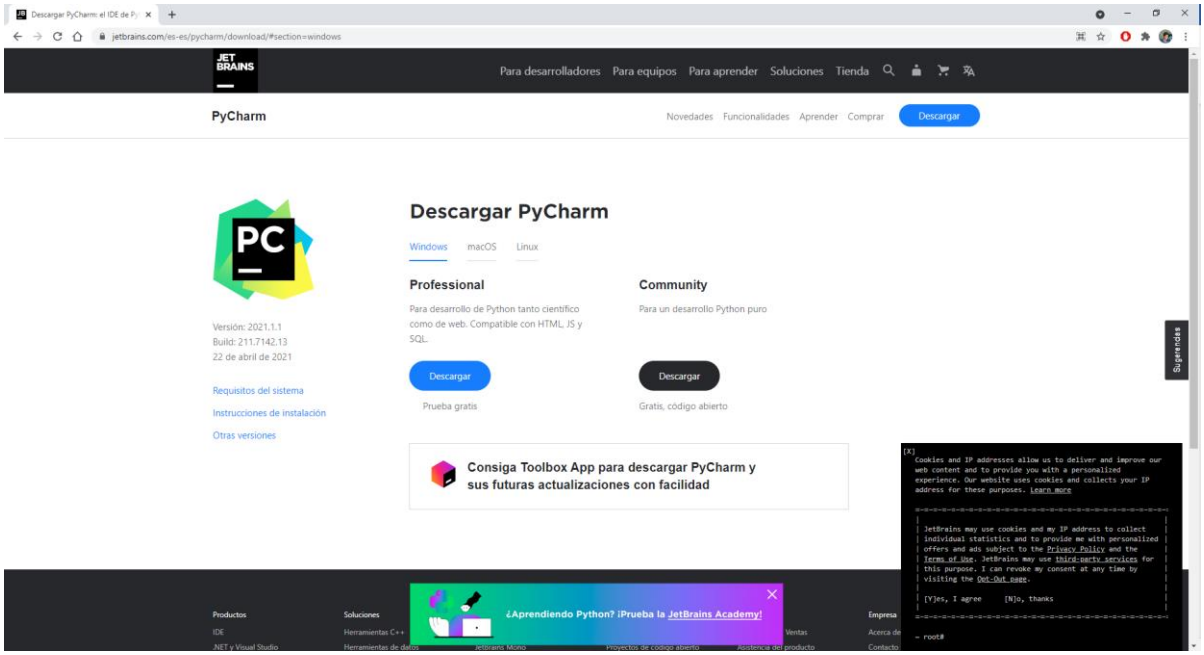


Ilustración 69: Descarga de PyCharm

Una vez realizada la descarga se ejecuta el archivo y se procede a la instalación sin necesidad de realizar o marcar algún elemento en concreto en el proceso.

RASPBERRY PI IMAGER

Es el programa que se va a emplear para crear, en la microSD que se inserta en la Raspberry Pi, la imagen para instalar el sistema operativo oficial de Raspberry: Raspberry Pi OS. Para ello, hay que dirigirse a la página oficial de Raspberry y acceder a este enlace: https://www.raspberrypi.org/software/. Se elige el sistema operativo que se esté empleando y se hace clic en Download. Se obtiene un archivo ejecutable y se instala el programa. Cabe destacar que existen otros programas que también permiten realizar la misma instalación.

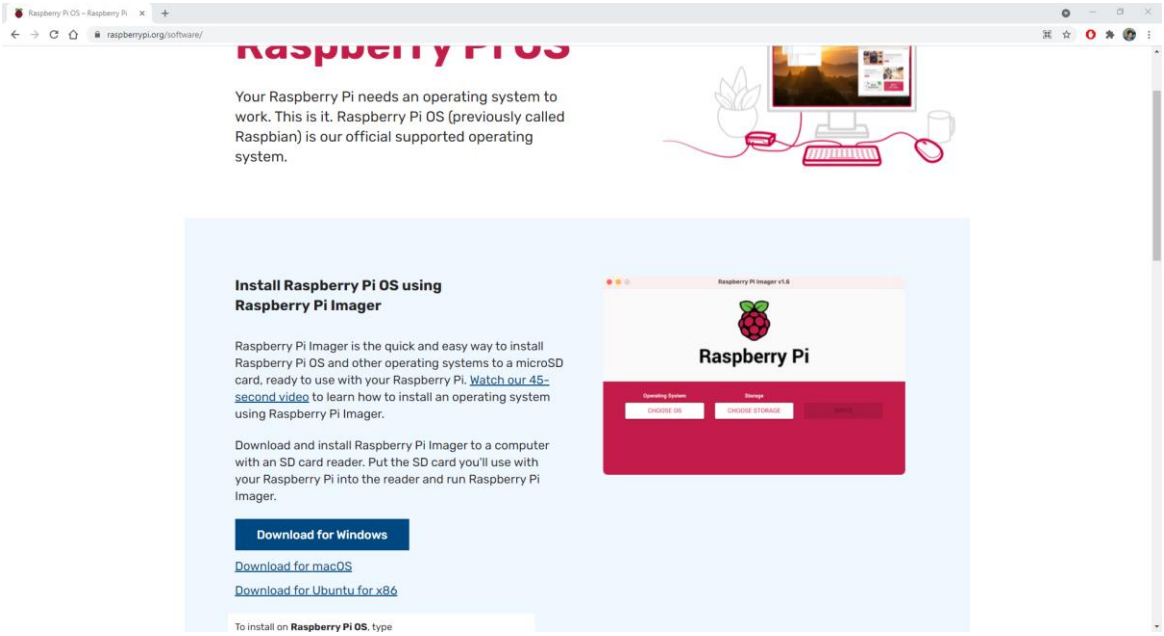


Ilustración 70: Descarga de Raspberry Pi Imager

RASPBERRY PI OS

Se trata del sistema operativo oficial para la Raspberry Pi. Cabe destacar que existen multitud de versiones compatibles con la Raspberry Pi y hay algunas que están específicamente diseñadas para ciertas finalidades como por ejemplo crear un servidor.

Una vez instalado el programa Raspberry Pi Imager se puede proceder a la instalación de Raspberry Pi OS. Resulta necesario resaltar qué versión de Raspberry Pi OS instalar puesto que existen tres versiones diferentes: la versión normal de Raspberry Pi OS, Raspberry Pi OS Lite y Raspberry Pi OS Full.

La versión Full trae instalada varios programas por defecto y cuenta con interfaz gráfica. Por otro lado, la versión Lite no trae ningún programa por defecto salvo los básicos como Python y no dispone de interfaz gráfica. Esto implica que para interactuar con el sistema operativo se tiene que hacer a través de una terminal. Presenta la ventaja de ocupar menos espacio y consumir menos recursos del sistema. Sin embargo, para el proyecto se ha empleado la versión normal para facilitar la interacción de los nuevos usuarios con el sistema de seguridad. Cabe destacar que para el proceso de configuración de las fotos es necesario poder

visualizar la imagen que se ha obtenido y para ello resulta imprescindible contar con interfaz gráfica. Además, se ha visto que la Raspberry es capaz de ejecutar sin cortes el programa y se puede trabajar de forma cómoda.

Por tanto, a la hora de elegir la versión de Raspberry Pi OS se elige la que aparece nada más hacer clic sobre choose Operating Sytem (Raspberry Pi OS (32-bit)) tal como se muestra en la imagen de a continuación. Una vez realizado esto, se elige la microSD que se va a insertar en la Raspberry Pi y se procede a la creación de la imagen.

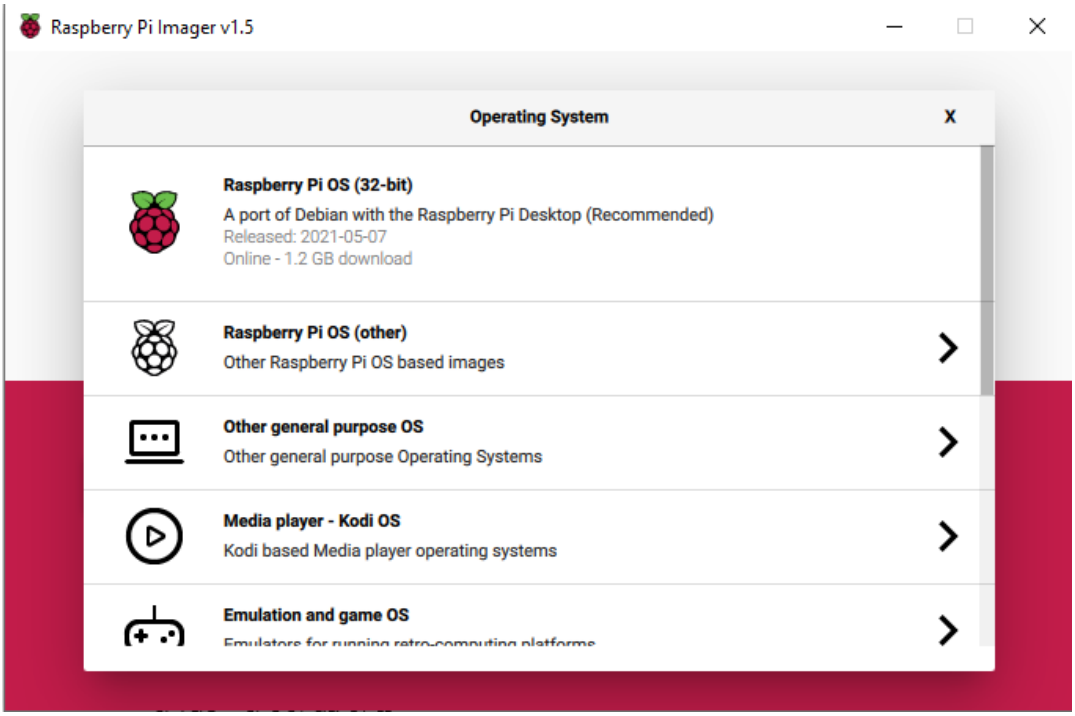


Ilustración 71: Elección de la versión de Raspberry Pi OS

Cabe destacar que, una vez creada la partición, para poder acceder posteriormente a la Raspberry sin necesidad de ratón, teclado y conexión a la pantalla, hay que añadir en la primera carpeta dos archivos “ssh” y “wpa_supplicant”. Con el uso de estos archivos se va a activar el modo ssh de la Raspberry y, por otro lado, se va a proporcionar a la Raspberry los datos de la red a la que se tiene que conectar para posteriormente poder acceder a ella de forma remota. Esto último se explicará en el apartado siguiente.

Para la obtención del archivo ssh simplemente se crea un archivo sin formato que tenga el nombre ssh. Por otro lado, para la obtención del archivo wpa_supplicant es necesario abrir el bloc de notas y escribir lo mostrado en la imagen de a continuación.

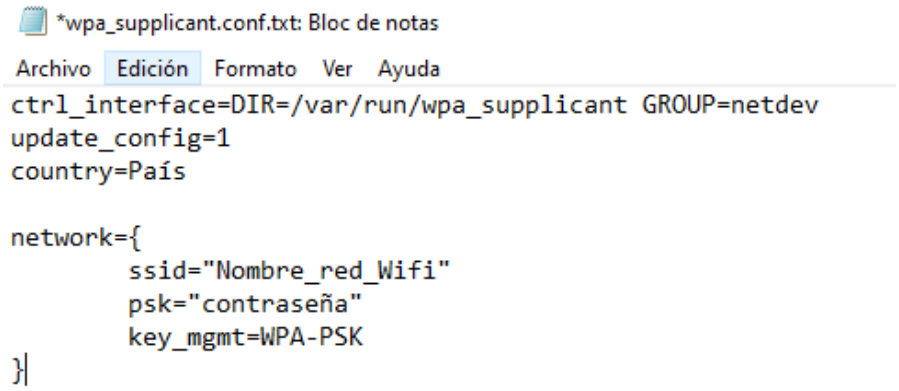


Ilustración 72: Creación del archivo wpa_supplicant

Tal como se puede observar, solo resulta necesario modificar el país en el que el usuario se encuentra. Si fuera España habría que escribir ES. Por otro lado, hay que proporcionar el nombre de la red wifi en la ssid teniendo en cuenta que debe estar entre comillas. A continuación, se escribe la contraseña. Por último, se debe proporcionar el tipo de seguridad de la red la cual se puede obtener mirando el router del domicilio, en el teléfono móvil en el apartado wifi o en el propio ordenador. Una vez realizado esto, se guarda el archivo con el nombre de wpa_supplicant.conf y puesto que se va a guardar como archivo .txt se tiene que cambiar el nombre del archivo eliminando el .txt final.

PUTTY

PuTTY es un programa de código libre que emplea el protocolo SSH a través del cual es posible conectarse a la Raspberry Pi. Hoy en día solo está disponible en Windows y Unix, pero se están desarrollando versiones para Mac OS X.

Para instalarlo hay que acceder a la página oficial de PuTTY y dirigirse al siguiente enlace https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html donde se podrá observar lo mostrado en la imagen de a continuación.

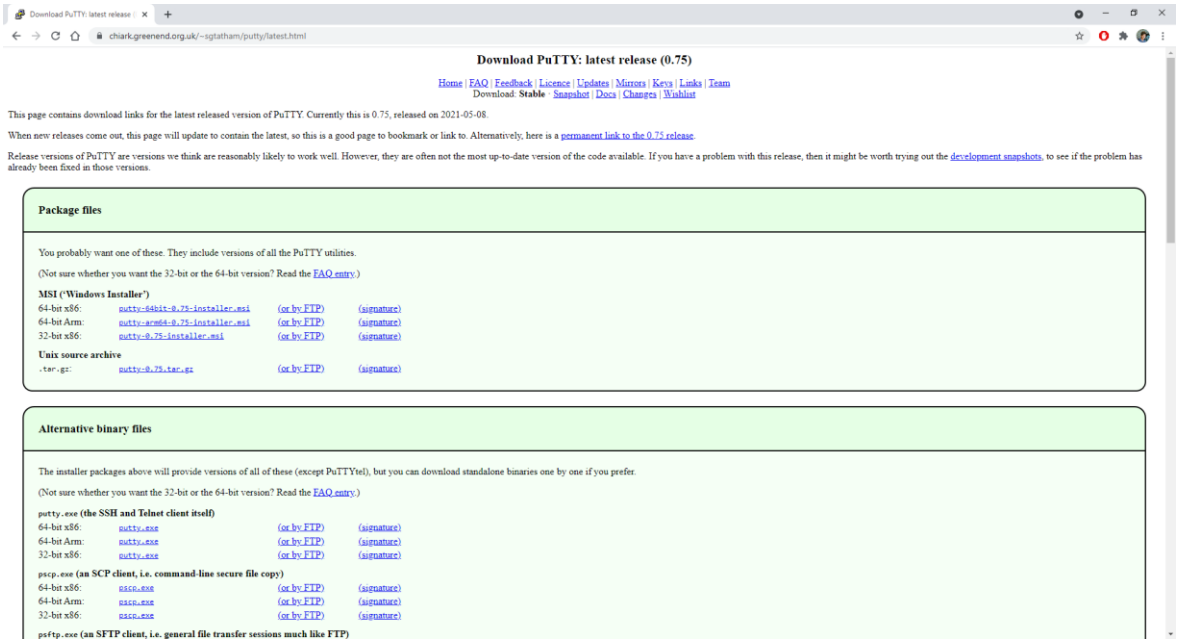


Ilustración 73: Descarga de puTTY

En función del tipo de sistema operativo Windows del que se disponga, se elige la versión de 64 o 32 bits y se procede a la descarga y su posterior instalación.

Tal como se comentó en el apartado anterior, antes de encender la Raspberry es necesario introducir dos archivos en la partición del sistema operativo. Uno de los archivos se encargará de activar la conexión SSH de la Raspberry. El otro archivo contendrá todos los datos de la red para que la Raspberry se pueda conectar sin necesidad de tener que usar un teclado, ratón y pantalla para conectarse a la red. Una vez encendida la Raspberry hay que acceder al router para encontrar la dirección IP que este le asigna a la Raspberry ya que será uno de los datos necesarios para poder conectarse a través de PuTTY. Una cosa importante que también hay que hacer es configurar el router para que siempre asigne la misma dirección IP a la Raspberry (IP estática) para no tener que estar constantemente mirando la dirección IP cada vez que encendamos la Raspberry. Cabe destacar que este último paso

depende del router del que disponga el usuario y puede variar en función del fabricante. Una vez que se ha realizado todo lo anterior se procede a acceder a la Raspberry.

Cuando se pone en marcha puTTY aparece el menú que se muestra a continuación. Se selecciona SSH y en el apartado de Host Name se introduce la dirección IP obtenida del router. Una vez introducidos estos datos se pulsa sobre open y se abre una ventana de una terminal en la que se pide introducir el usuario y la contraseña de la Raspberry. En el primer arranque es necesario introducir el usuario: pi y la contraseña: raspberry ya que se tratan de los datos por defecto.

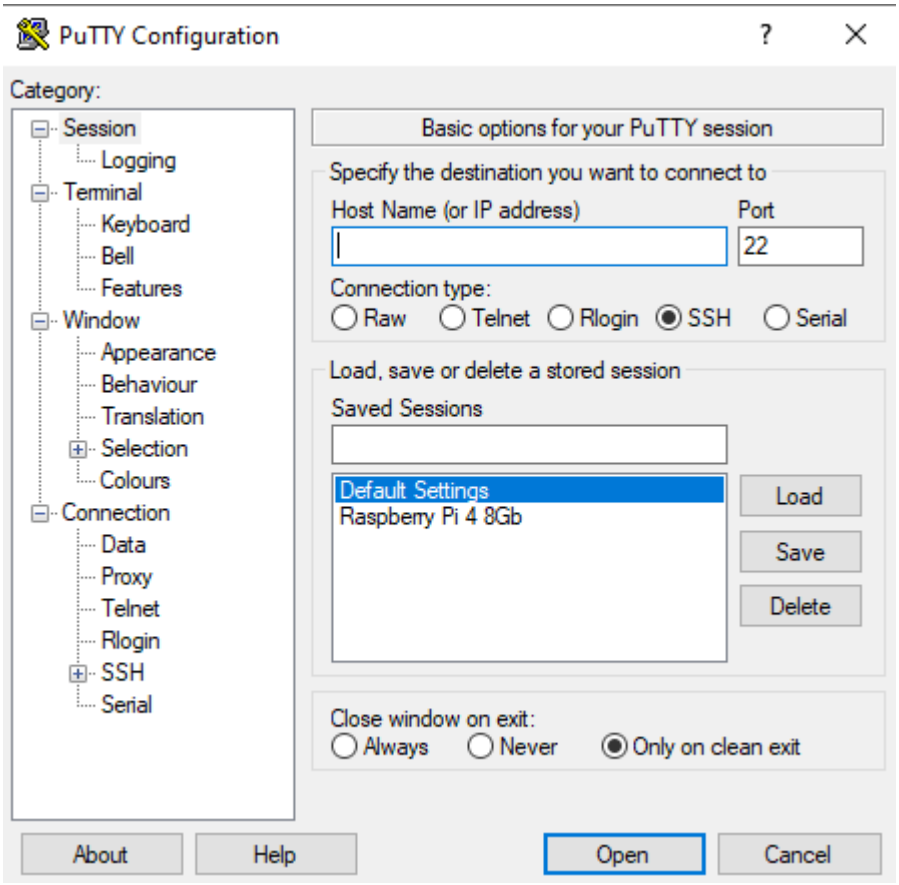


Ilustración 74: Acceso a la Raspberry

VNC VIEWER

Se trata de un programa perteneciente a la empresa REALVNC, la cual ha creado la tecnología VNC. Este programa permite conectarse a la Raspberry Pi de forma remota y presenta la gran ventaja de poder trabajar con la interfaz gráfica de Raspberry Pi OS resultando mucho más cómodo que estar trabajando con una terminal. Además, es un programa que es compatible con múltiples plataformas.

Para instalar el programa hay que acceder a la página oficial de REALVNC y dirigirse al apartado de descargas. https://www.realvnc.com/es/connect/download/viewer/ En él se muestran varias pestañas correspondientes a cada uno de los sistemas operativos que se puede instalar el programa. Se elige el sistema operativo, se procede a la descarga y se instala el programa.

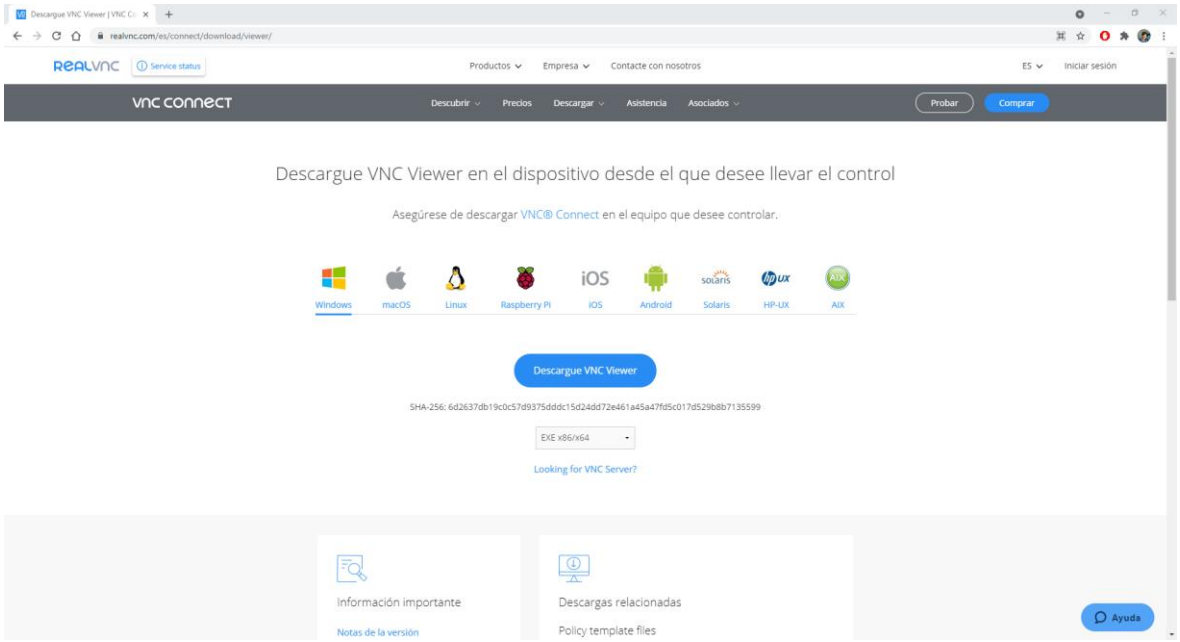


Ilustración 75: Descarga de RealVNC

Un aspecto muy importante a tener en cuenta es que la Raspberry Pi no trae la función VNC activada por defecto. Por este motivo es necesario instalar y configurar puTTY en primer

lugar. Es por ello que antes de conectarse a través de VNC Viewer resulta necesario seguir los pasos que se van a comentar a continuación.

En primer lugar, accedemos a la Raspberry usando PuTTY y empleando el usuario y contraseña por defecto. Una vez realizado esto se podrá observar lo siguiente en la terminal.

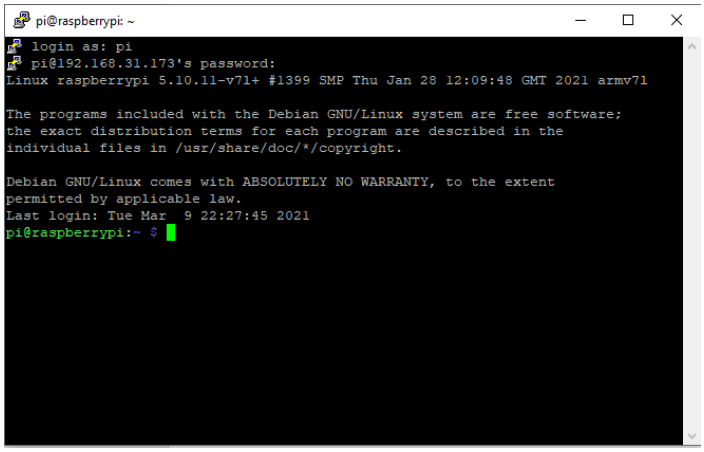


Ilustración 76: Acceso a la Raspberry Pi a través de la terminal de PuTTY

A continuación, se escribe el siguiente comando en la terminal: sudo raspi-config. Introduciendo este comando se accede al menú de configuración general de la Raspberry Pi observando lo que se muestra en la imagen siguiente.

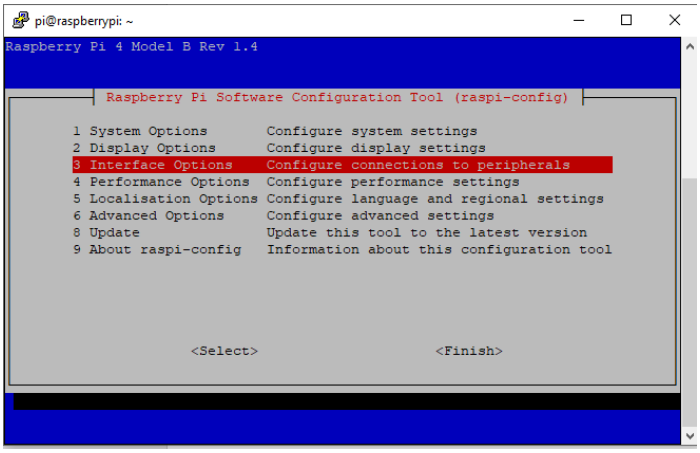


Ilustración 77: Menú principal de la Raspberry Pi

Tal como está indicado en la ilustración anterior, se selecciona la opción interface options y se despliega un nuevo menú en el que en la tercera opción se observa VNC. Simplemente se pulsa sobre él y se requerirá confirmar la activación de VNC tal como se muestra en la imagen de a continuación.

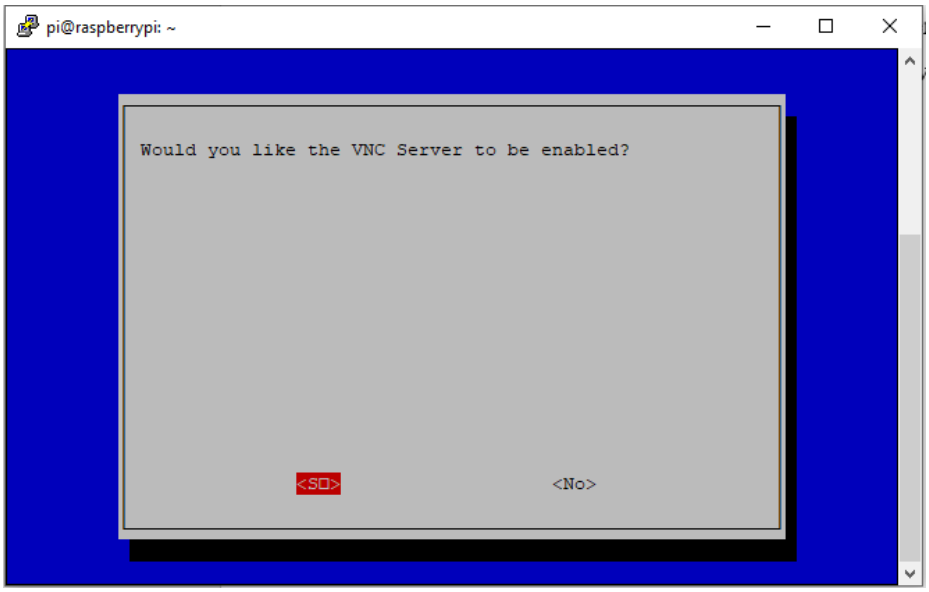


Ilustración 78: Mensaje de confirmación para activar VNC

Una vez realizado esto último, aparece una nueva ventana en la que se confirma la activación de VNC y se retrocede al menú principal.

Ahora que VNC está activado se puede acceder a la Raspberry usando REALVNC y únicamente se requerirá la dirección IP de la Raspberry.

Ilustración 79: Acceso a la Raspberry a través de VNC

Tal como se puede observar en la ilustración anterior, la dirección IP se introduce en el menú de "Especifique una dirección…". Cabe destacar que, tras realizar algún acceso, el programa almacena la dirección IP y es por ello que se pueden observar dos opciones para acceder a una de las Raspberry.

Cuando se consigue acceder a la Raspberry se muestra una interfaz similar a la de la imagen de a continuación y se puede trabajar sobre ella como si se tuviera el ratón o el teclado conectado a uno de los puertos de la Raspberry.

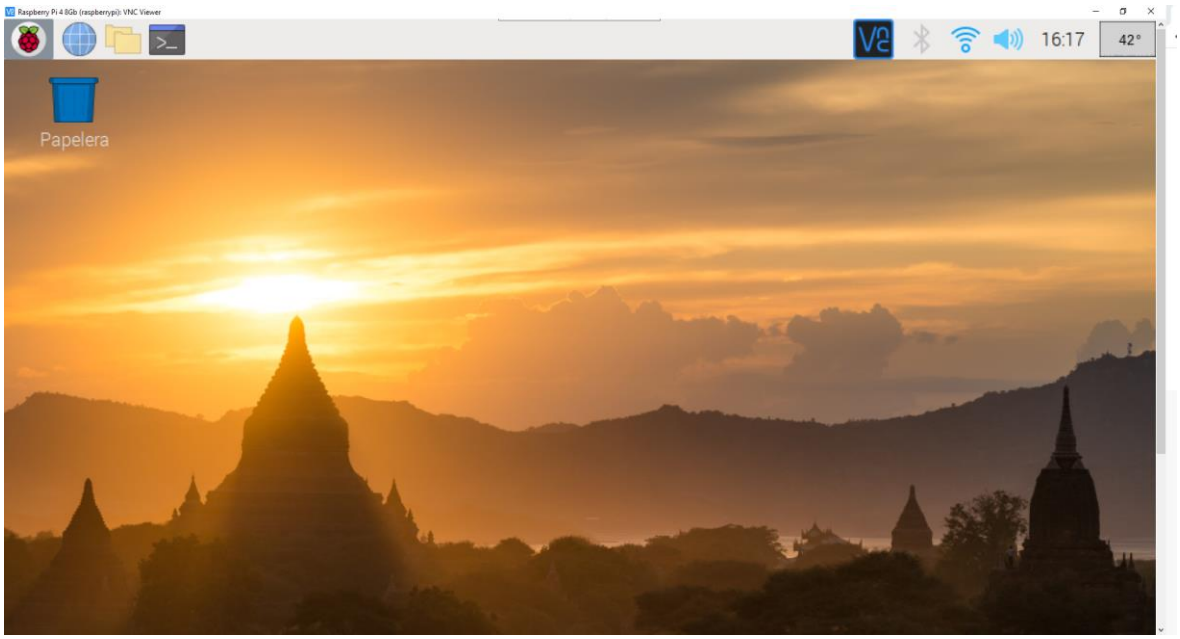


Ilustración 80: Interfaz de Raspberry Pi OS usando VNC Viewer

WINSCP

Se trata del último programa que se va a emplear para interactuar con la Raspberry Pi. WinSCP es un programa de código abierto que se va a usar para intercambiar archivos entre la Raspberry y el ordenador. Va a resultar especialmente útil para transferir los nuevos códigos a la Raspberry y de esta forma probarlos. Tiene el inconveniente de ser únicamente compatible con Windows, pero en los casos en los que no se disponga de Windows existen otros programas que realizan la misma función o mediante el uso del correo electrónico desde la Raspberry se puede auto enviar los nuevos códigos.

Para instalar el programa se va a acceder a la página oficial y en el apartado de descargas https://winscp.net/eng/download.php se podrá observar lo siguiente.

Ilustración 81: Descarga de WinSCP

Se hace clic sobre Download WINSCP y una vez descargado se ejecuta el archivo ejecutable para instalar el programa.

Al igual que con puTTY y VNC Viewer va a ser necesario conocer la dirección IP de la Raspberry. Además, se tiene que introducir el usuario y la contraseña configurada en la Raspberry.

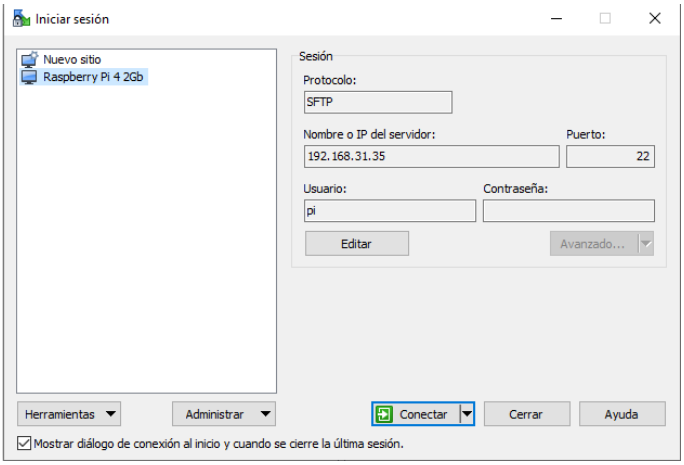


Ilustración 82: Acceso a la Raspberry Pi con WinSCP

Una vez obtenido el acceso, se despliega un menú mostrando carpetas pertenecientes al ordenador y otro de las carpetas de la Raspberry. Se puede arrastrar los archivos o bien usar los comandos de copiar y pegar que se muestran en el menú principal del programa.

SOLIDWORKS

Es un programa CAD empleado para el diseño de la carcasa del sistema de seguridad. Es necesario disponer de una licencia para poder usarlo, pero por ser estudiante se puede disponer de la licencia de forma gratuita. Por otro lado, solo es compatible con Windows.

Cabe destacar que existen múltiples programas CAD que se pueden adquirir de forma gratuita y permiten diseñar la carcasa de forma similar.

Para su instalación hay que acceder a la página oficial de SolidWorks a través del enlace https://www.solidworks.com/es/support/community-download#no-back y rellenar los datos que piden. En el proceso de instalación se siguen las instrucciones y se introduce la licencia de la que se tiene que disponer.

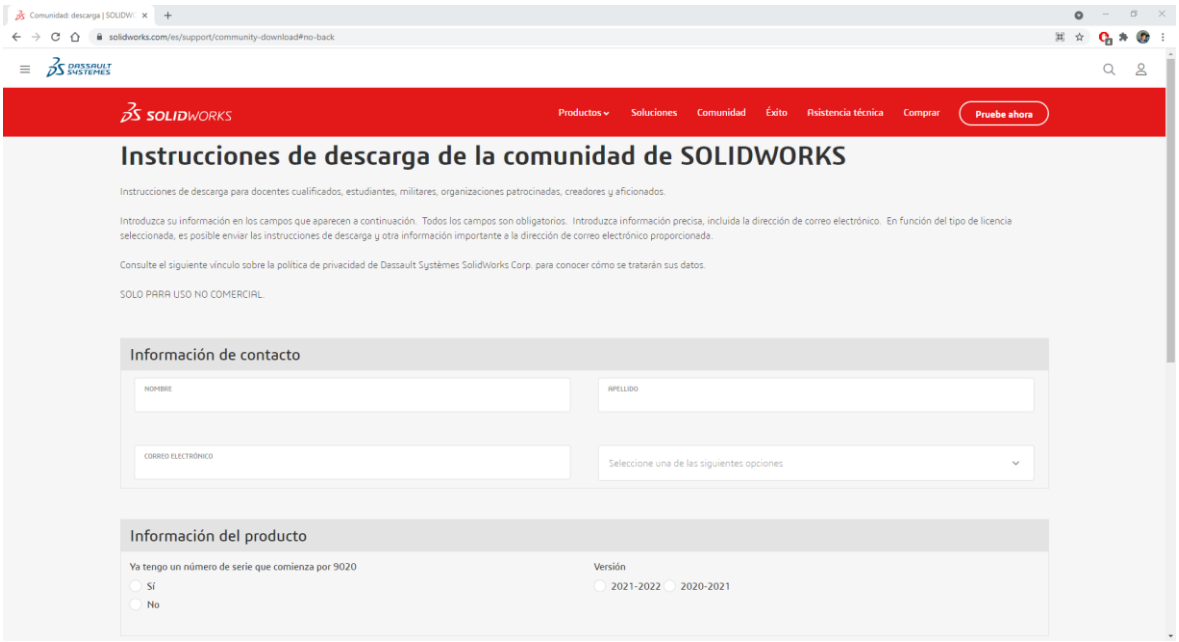


Ilustración 83: Descarga de SolidWorks desde la página oficial

ULTIMAKER CURA

Es un programa de impresión 3D con el que se van a configurar los parámetros necesarios para una correcta impresión. Es multiplataforma y es compatible con Windows, Linux y Mac OS.

Para su instalación es necesario acceder a la página oficial de Ultimaker y usando el enlace siguiente https://ultimaker.com/es/software/ultimaker-cura se llega al apartado de descargas. Haciendo clic sobre descarga gratuita se despliega un menú en el que se tiene que elegir el sistema operativo y una vez seleccionado se descarga el archivo de instalación.

El proceso de instalación es sencillo y en la primera ejecución del programa se tiene que seleccionar la impresora 3D que se va a emplear y ciertos parámetros.

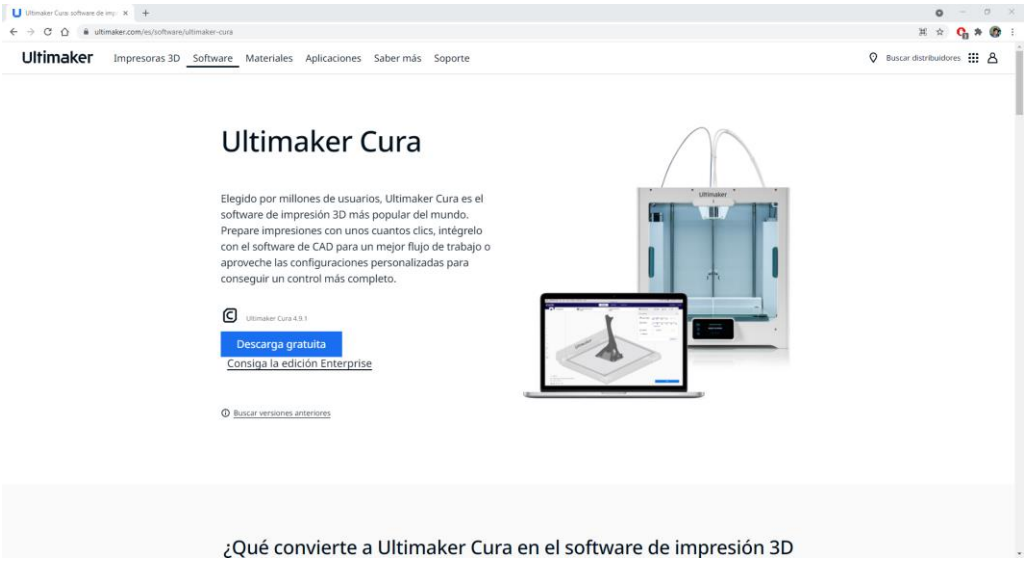


Ilustración 84: Descarga de Ultimaker Cura

ANEXO II: CÓDIGO

RASPBERRY

En este apartado se va a encontrar todo el código ejecutado en la Raspberry Pi y se va a dividir entre los distintos módulos que componen el programa.

main.py

```
# -*- coding: utf-8 -*-
# Importación de librerías
import cv2
import numpy as np
from os.path import isfile, join
from os import listdir
import os
import time  # Librería que usaremos para reducir el número de envíos entre
correos
from FaceRecognition import FaceRecognition  # Módulo del reconocimiento facial
from YoloDetection import Yolo  # Módulo del algoritmo Yolo
from gmail import Gmail  # Módulo del correo
from threading import Timer  # Importamos la librería timer que es una especie de
periférico (Interrupciones)
from flask import Flask, render_template, Response
import multiprocessing
import threading  # Librería para multihilo (varios programas funcionando a la
vez)
from reproductor import *  # Importamos todas las funciones del reproductor
from fingerprint import HuellaDactilar  # Módulo del reconocimiento de huellas
dactilares

# Librerías necesarias para trabajar con SmartWorks
import json
import paho.mqtt.client as mqtt           # Librería para trabajar con MQTT
import Modules.datos_dispositivo as raspberry # Módulo para actualización de
variables del dispositivo
import Modules.smartworks_sdk as swx
import Modules.credentials as credenciales    # Módulo con los datos de acceso a
SmartWorks

# Guardamos los audios que nos interesan y no son variables
# Configuración inicial:
texto = 'Bienvenido al asistente de configuración del sistema de seguridad:'
indice = 'Configuration assistant'
```

```
guarda(texto, indice)
# Detección de una persona:
texto = 'Acérquese a la cámara para detectar su cara:'
indice = 'Personadetectada'
guarda(texto, indice)

######################## MQTT ######################
nombre_dispositivo = credenciales.computers[0] # Accedemos a los datos del
dispositivo de interés

# CONFIGURACIÓN del broker MQTT
# Variables MQTT
host = "mqtt.swx.altairone.com"    # url del MQTT
user, password = credenciales.mqtt_credentials(nombre_dispositivo)

# Variables del espacio de trabajo
space = "nicols"                    # Nombre del espacio de trabajo de Smart Works
collection = "raspberry"            # Nombre de la colección que almacena los
things (los objetos conectados a internet)
thing_id = credenciales.thing_ids(nombre_dispositivo)

# Conexión al broker
client = mqtt.Client()
client.username_pw_set(username=user, password=password)
client.connect(host, port=1883, keepalive=120, bind_address="")

######################   CREACIÓN DE TOPICS  ############################
data_topic = "set/{}/collections/{}/things/{}/data".format(space, collection,
thing_id)
property_topic_cpu =
"set/{}/collections/{}/things/{}/properties/{}".format(space, collection,
thing_id, "cpu")         # Uso de la cpu
property_topic_temp =
"set/{}/collections/{}/things/{}/properties/{}".format(space, collection,
thing_id, "temp")       # Temperatura
property_topic_disk =
"set/{}/collections/{}/things/{}/properties/{}".format(space, collection,
thing_id, "disk")       # Uso del disco
property_topic_memory =
"set/{}/collections/{}/things/{}/properties/{}".format(space, collection,
thing_id, "memory")   # Memoria libre
property_topic_status =
"set/{}/collections/{}/things/{}/properties/{}".format(space, collection,
thing_id, "status")   # Estado de  Raspberry
property_topic_access =
"set/{}/collections/{}/things/{}/properties/{}".format(space, collection,
thing_id, "access")   # Indica el acceso de una persona

reconocimiento_huellas = False
# Actualizamos los valores de los parámetros del sistema
raspberry.actualiza_datos(reconocimiento_huellas)
```

```
# ENVÍO DE DATOS INICIALES
client.publish(topic=data_topic,
payload=str(json.dumps(raspberry.datos_dispositivo)))

# El envío de datos se hace con el formato json string
# Actualización de las propiedades
property_cpu = {"cpu": raspberry.info_CPU, }
property_temp = {"temp": raspberry.temperaturaCPU.temperature, }
property_disk = {"disk": raspberry.disco_porcentaje, }
property_memoria = {"memory": raspberry.memoria_porcentaje, }
property_status = {"status": 'Menu inicial', }

# Se envían los datos actualizados a sus respectivos topics
client.publish(topic=property_topic_cpu, payload=str(json.dumps(property_cpu)))
client.publish(topic=property_topic_temp, payload=str(json.dumps(property_temp)))
client.publish(topic=property_topic_disk, payload=str(json.dumps(property_disk)))
client.publish(topic=property_topic_memory,
payload=str(json.dumps(property_memoria)))
client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))

# LIMITAR NÚMERO DE HILOS
from concurrent.futures import ThreadPoolExecutor
threadpool = ThreadPoolExecutor(10)  # 10 threads máximo

def threaded(fn):
    def wrapper(*args, **kwargs):
        return threadpool.submit(fn, *args, **kwargs)
    return wrapper

# Constantes necesarias
intervalo_tiempo = 60  # Segundos porque time funciona en segundos
intervalo_dactilar = 60  # Intervalo para el reconocimiento de huellas dactilares
tiempo = intervalo_tiempo
contador_yolo = 0
max_caras = 7
max_yolo = 7
reconocimiento_facial = False
reconocimiento_huellas = False


# Clase contador
class Contador():
    # Métodos
    # Constructor
    def __init__(self):
        self.reconocimiento_facial = False
        self.tiempoacabado_yolo = False
        self.tiempoacabado_face = False
```

```
    # Función para cuando se alcanzan los 15 segundos
    def contador(self):
        print('Han pasado 15 segundos')
        self.tiempoacabado_face = True

    # Función para Yolo cuando se alcanzan los 30 segundos
    def temporizador(self):
        print('Han pasado 30 segundos y no se ha detectado a una persona el
suficiente número de veces')
        self.tiempoacabado_yolo = True

        ##### MQTT #######  Cada 30 segundos se envían datos actualizados
        # Actualizamos los parámetros
        raspberry.actualiza_datos(reconocimiento_huellas)

        # Actualización de valores de las propiedades para enviar a los topics
        property_cpu = {"cpu": raspberry.info_CPU, }
        property_disk = {"disk": raspberry.disco_porcentaje, }
        property_memory = {"memory": raspberry.memoria_porcentaje, }
        property_temp = {"temp": raspberry.temperaturaCPU.temperature, }

        client.publish(topic=property_topic_cpu,
payload=str(json.dumps(property_cpu)))
        client.publish(topic=property_topic_disk,
payload=str(json.dumps(property_disk)))
        client.publish(topic=property_topic_memory,
payload=str(json.dumps(property_memory)))
        client.publish(topic=property_topic_temp,
payload=str(json.dumps(property_temp)))

        # ENVÍO DE DATOS
        client.publish(topic=data_topic,
payload=str(json.dumps(raspberry.datos_dispositivo)))

# Creamos el objeto yolo
yolo = Yolo()
# Creamos objeto vídeo para el reconocimiento facial
video = FaceRecognition()

# Constructor de la aplicación Flask
app = Flask(__name__)


# Programa principal
def programa_principal():
    # Hacemos que en el primer arranque tarde un poco más para que no solapen los
mensajes de los dos programas:
    time.sleep(5)
    while True:
        # ASISTENTE DE CONFIGURACIÓN INICIAL
        indice = 'Configuration assistant'
```

```
        reproduce(indice)
        print(threading.activeCount())  # Para saber el número de hilos en marcha

        print('Bienvenido al asistente de configuración del sistema de
seguridad:\n Elija la opción:')
        # print('Elija la opción:')
        print('1.- Configurar persona con derecho de admisión')
        print('2.- Configurar correo para el envío de mensajes')
        print('3.- Poner en marcha sistema de seguridad')
        print('4.- Configurar nueva huella dactilar o eliminar huella
registrada')
        print('5.- Mostrar perfiles configurados')

        opcion = int(input(
            'Escriba en el teclado la opción que desee:'))  # Para que nos
almacene un valor de tipo entero sino estaría almacenando un string
        print('Ha elegido la opcion:' + str(opcion))
        # Otras formas de mostrar por pantalla un valor:
        # print('Ha elegido la opcion: {}'.format(opcion))
        # print('Ha elegido la opcion:', opcion)

        # Configuraración persona con derecho de admisión
        if opcion == 1:
            ####### MQTT #######
            property_status = {"status": 'Configurando nueva persona en el
sistema', }
            # ENVÍO DE DATOS
            client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))

            nombre_persona = input('Escriba su nombre: ')
            print('Colóquese bien cerca de la cámara y pulse la tecla f cuando
quiera hacer la foto:')

            # Audio de configuración
            texto = 'Colóquese bien cerca de la cámara y pulse la tecla F cuando
quiera hacer la foto:'
            indice = 'configurarnuevapersona'
            guarda(texto, indice)
            reproduce(indice)

            while True:
                captura = cv2.VideoCapture(0)
                if not captura.isOpened():
                    print("No se puede abrir la cámara")
                    exit()
                ret, image = captura.read()
                if ret == False:
                    print('No se ha podido leer correctamente el frame')

                cv2.imshow('Foto', image)
```

```
                if cv2.waitKey(1) & 0xFF == ord(
                        'f'):  # M antenemos la ventana abierta hasta pulsar la
tecla f y así tomar una foto
                    # Almacenamos la foto realizada con el nombre de la persona
                    cv2.imwrite("Resources/FaceRecognition/ " +
str(nombre_persona) + '.jpg', image)
                    # cv2.imwrite(str(nombre_persona)+'.jpg', img)
                    print('Foto almacenada correctamente')
                    cv2.destroyAllWindows()
                    captura.release()
                    break  # Para salir del bucle while
                captura.release()

        # Configuración correo para el envío de mensajes
        if opcion == 2:
            ####### MQTT #######
            property_status = {"status": 'Configurando correo', }
            # ENVÍO DE DATOS
            client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))

            # Audio de configuración:
            texto = 'Introduzca los datos de su correo'
            indice = 'configurarcorreo'
            guarda(texto, indice)

            # Creamos el objeto
            datos = Gmail()
            datos.config_correo()

        # Funcionamiento del sistema de seguridad
        if opcion == 3:
            ############ MQTT #############
            property_status = {"status": 'objectdetection', }
            # ENVÍO DE DATOS
            client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))

            # Creamos el objeto temporizador_yolo:
            temporizador_yolo = Contador()

            # Creamos el objeto temporizador_facerecognition:
            temporizador_face_recognition = Contador()

            # Configuramos e inicializamos el temporizador que usaremos para
yolo:
            temporizador = Timer(30, temporizador_yolo.temporizador)  # Nuevo
objeto timer
            temporizador.start()

            global reconocimiento_facial
```

```
            reconocimiento_huellas = False
            # Ponemos las variables de tiempo a 0
            tiempo_dactilar = 0
            tiempo = 0

            while True:
                # Leemos la cámara del dispositivo:
                # captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)   # Se resuelve el
warning en Pycharm. En la Raspberry no funciona
                captura = cv2.VideoCapture(0)
                if not captura.isOpened():
                    print("No se puede abrir la cámara")
                    exit()

                # -----------------DETECCIÓN DE OBJETOS-------------------------
                if reconocimiento_facial == False:
                    print('Entramos en el algoritmo YOLO')
                    # Ejecutamos el algoritmo Yolo
                    # Almacenamos en un atributo del objeto video la captura:
                    yolo.capture = captura
                    yolo.yolo_outputs = yolo.detecta_objetos()
                    yolo.encuentraObjetos(yolo.yolo_outputs, yolo.img)
                    # Abrimos una ventana mostrando el resultado obtenido:
                    # cv2.imshow('Analisis YOLO', yolo.img)
                    # Cogemos la imagen actualizada para convertirla a un formato
manejable para Flask
                    exito, foto = cv2.imencode('.jpg', yolo.img)
                    # Convertimos la imagen en bits
                    yolo.converted_image = foto.tobytes()

                    if yolo.thereisperson:  # Si detectamos a una persona,
sumamos
                        yolo.contador_yolo += 1
                        print(yolo.contador_yolo)  # Mostramos por pantalla la
cuenta

                        # -------------------------ENVÍO POR CORREO--------------
                        if (time.time() - tiempo) > intervalo_tiempo:
                            # Creamos el objeto correo
                            correo = Gmail()
                            # Convertimos la imagen en vectores
                            exito, jpeg = cv2.imencode('.jpg', yolo.img)
                            # Convertimos la imagen en bits
                            imagen_convertida = jpeg.tobytes()

                            correo.enviar_foto(imagen_convertida)
                            tiempo = time.time()

                    if yolo.contador_yolo > max_yolo:  # Cuando alcanzamos el
máximo de personas detectadas pasamos al reconocimiento facial
                        yolo.contador_yolo = 0  # Reiniciamos el contador
```

```
                        reconocimiento_facial = True

                        ####### MQTT #######
                        property_status = {"status": 'facerecognition', }
                        # ENVÍO DE DATOS
                        client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))

                        # Ponemos en marcha en otro hilo el audio de que se ha
detectado a una persona
                        indice = 'Personadetectada'
                        reproduce(indice)

                        # Configuramos e inicializamos el contador de
face_recognition de 15 segundos:
                        # global contador_facerecognition
                        contador_facerecognition = Timer(15,
temporizador_face_recognition.contador)  # Nuevo objeto timer
                        contador_facerecognition.start()

                    # Si en 30 segundos no se consiguen 7 dectecciones de persona
se vuelve a empezar a contar:
                    if (yolo.contador_yolo < max_yolo) &
(temporizador_yolo.tiempoacabado_yolo == True):
                        temporizador_yolo.tiempoacabado_yolo = False  # Para que
no vuelva a entrar en el bucle hasta que pasen otros 30 segundos
                        temporizador = Timer(30, temporizador_yolo.temporizador)
# Nuevo objeto timer
                        temporizador.start()
                        print('Iniciamos una nueva cuenta de 30 segundos:')
                        yolo.contador_yolo = 0  # Reseteamos la cuenta que
llevábamos

                # ------------RECONOCIMIENTO FACIAL-------------
                if reconocimiento_facial == True:
                    print('Entramos en el reconocimiento facial:')
                    # Cancelamos el temporizador de yolo
                    temporizador.cancel()

                    # Almacenamos en un atributo del objeto video la captura:
                    video.capture = captura  # Lo guardamos aquí para no ir
consumiento memoria cuando no se ejecuta
                    print(video.cuentacaras)

                    # Si en 15 segundos no se consiguen 7 dectecciones de cara se
vuelve a empezar a contar y se sale del reconocimiento facial:
                    if (video.cuentacaras < max_caras) and
(temporizador_face_recognition.tiempoacabado_face == True):
                        print('Nos salimos del reconocimiento facial')
                        video.cuentacaras = 0  # Reseteamos la cuenta de las
caras
```

```
                        temporizador_face_recognition.tiempoacabado_face = False
# Para que no vuelva a entrar en el bucle hasta que pasen otros 15 segundos
                        # Nos salimos del reconocimiento facial
                        reconocimiento_facial = False

                        # Volvemos a poner en marcha el temporizador del detector
de objetos
                        temporizador = Timer(30, temporizador_yolo.temporizador)
# Nuevo objeto timer
                        temporizador.start()

                        ####### MQTT #######
                        property_status = {"status": 'objectdetection', }
                        client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))
                        property_temp = {"temp":
raspberry.temperaturaCPU.temperature, }
                        client.publish(topic=property_topic_temp,
payload=str(json.dumps(property_temp)))
                        # ENVÍO DE DATOS
                        client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))
                        # Actualizamos los valores de los parámetros del sistema
                        raspberry.actualiza_datos(reconocimiento_huellas)
                        # Enviamos los datos
                        client.publish(topic=data_topic,
payload=str(json.dumps(raspberry.datos_dispositivo)))

                    # Mostramos por pantalla la persona detectada
                    video.detecta_cara()
                    print(video.persona_detectada)

                    # ----------RECONOCIMIENTO DE HUELLAS DACTILARES------
                    if (video.cuentacaras > max_caras) and
video.persona_detectada != 'No se detecta a nadie':
                        print('Se ha entrado en el proceso de detección de
huellas dactilares:')
                        # Resetamos la cuenta de las caras que llevamos
                        video.cuentacaras = 0
                        reconocimiento_facial = False

                        ####### MQTT #######
                        property_status = {"status": 'fingerprintrecognition', }
                        client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))
                        property_temp = {"temp":
raspberry.temperaturaCPU.temperature, }
                        client.publish(topic=property_topic_temp,
payload=str(json.dumps(property_temp)))

                        # Cancelamos el temporizador de face_recognition
```

```
                        contador_facerecognition.cancel()

                        # Persona detectada:
                        texto = 'Bienvenido a casa, ' +
str(video.persona_detectada) + '. Introduce la huella en el sensor:'
                        indice = 'bienvenidaacasa'
                        guarda(texto, indice)
                        reproduce(indice)

                        # Creamos el objeto que vamos a usar para el
reconocimiento de huellas:
                        detecta_huellas = HuellaDactilar()
                        detecta_huellas.comprobar_huella()

                        if detecta_huellas.huella_correcta:
                            reconocimiento_huellas = True
                            print('Se ha detectado una huella registrada en el
sistema')
                            # Ponemos en marcha en otro hilo el audio de que se
ha detectado la huella correctamente
                            texto = 'La huella se ha detectado correctamente.'
                            indice = 'huelladetectada'

                            ####### MQTT #######
                            # Actualizamos los valores de los parámetros del
sistema
                            raspberry.actualiza_datos(reconocimiento_huellas)
                            # Enviamos los datos
                            client.publish(topic=data_topic,
payload=str(json.dumps(raspberry.datos_dispositivo)))
                            property_access = {"access": '1', }
                            # ENVÍO DE DATOS
                            client.publish(topic=property_topic_status,
payload=str(json.dumps(property_access)))

                            guarda(texto, indice)
                            reproduce(indice)

                        else:
                            reconocimiento_huellas = False
                            print('No se identifica su huella')

                            ####### MQTT #######
                            # Actualizamos los valores de los parámetros del
sistema
                            raspberry.actualiza_datos(reconocimiento_huellas)
                            # Enviamos los datos
                            client.publish(topic=data_topic,
payload=str(json.dumps(raspberry.datos_dispositivo)))
```

```
                            # Enviamos correo de la persona que ha introducido
una huella que no es
                            print('Se ha enviado una imagen de la persona que
está intentando acceder al domicilio')
                            correo_entrada = Gmail()
                            correo_entrada.enviar_correo(video.persona_detectada,
video.converted_image)

                        # Enviamos un mensaje de texto avisando que alguien ha
accedido al domicilio
                        if (reconocimiento_huellas == True):
                            print('Se ha enviado una imagen de la persona que ha
accedido al domicilio')
                            correo_entrada = Gmail()
                            correo_entrada.enviar_correo(video.persona_detectada,
video.converted_image)


                        # Volvemos a poner en marcha el temporizador del detector
de objetos
                        temporizador = Timer(30, temporizador_yolo.temporizador)
# Nuevo objeto timer
                        temporizador.start()

                        captura.release()

                    # Caso en el que se detecta a alguien pero no es conocido
                    elif (video.cuentacaras > max_caras) and
(video.persona_detectada == 'No se detecta a nadie'):
                        print('No está registrado en el sistema')
                        # Reseteamos la cuenta de las caras que llevamos
                        video.cuentacaras = 0
                        reconocimiento_facial = False
                        # Cancelamos el temporizador de face_recognition
                        contador_facerecognition.cancel()

                        # Volvemos a poner en marcha el temporizador del detector
de objetos
                        temporizador = Timer(30, temporizador_yolo.temporizador)
# Nuevo objeto timer
                        temporizador.start()
                        captura.release()

                # Necesario para que luego pueda abrir la cámara sin error
                captura.release()

        if opcion == 4:
            ####### MQTT #######
            property_status = {"status": 'Configurando huellas dactilares', }
            # ENVÍO DE DATOS
```

```
            client.publish(topic=property_topic_status,
payload=str(json.dumps(property_status)))

            # Audio de configuración:
            texto = 'Elija si desea configurar una nueva huella o quiere borrar
una huella anterior'
            indice = 'configurarhuellas'
            guarda(texto, indice)

            # Creamos un objeto para inicializar el lector de huellas
            print('Elija la opción que desee:')
            print('1.- Configurar nueva huella')
            print('2.- Borrar huella')
            eleccion = int(input('Escriba en el teclado la opción que desee:'))
            if eleccion == 1:
                nueva_huella = HuellaDactilar()
                nueva_huella.configurar_huella()  # Ejecutamos el método de
configuración de una nueva huella

            if eleccion == 2:
                nueva_huella = HuellaDactilar()
                nueva_huella.borrar_huella()

        if opcion == 5:
            print('A continuación se mostrarán las personas configuradas en el
sistema:')

            # Función para apilar todas las imágenes en una foto
            def apilar_imagenes(escala, vectorFotos, nombres):
                tamanoAncho = vectorFotos[0][0].shape[1]
                tamanoAlto = vectorFotos[0][0].shape[0]
                filas = len(vectorFotos)
                columnas = len(vectorFotos[0])
                filasDisponibles = isinstance(vectorFotos[0], list)
                ancho = vectorFotos[0][0].shape[1]
                alto = vectorFotos[0][0].shape[0]
                if filasDisponibles:
                    for x in range(0, filas):
                        for y in range(0, columnas):
                            vectorFotos[x][y] = cv2.resize(vectorFotos[x][y],
(tamanoAncho, tamanoAlto), None, escala)
                            if len(vectorFotos[x][y].shape) == 2:
                                vectorFotos[x][y] =
cv2.cvtColor(vectorFotos[x][y], cv2.COLOR_GRAY2BGR)

                    imagenBlanco = np.zeros((alto, ancho, 3), np.uint8)
                    horizontal = [imagenBlanco] * filas
                    horizontal_concatenada = [imagenBlanco] * filas

                    for x in range(0, filas):
                        horizontal[x] = np.hstack(vectorFotos[x])
```

```
                        horizontal_concatenada[x] =
np.concatenate(vectorFotos[x])
                    vertical = np.vstack(horizontal)
                    ver_con = np.concatenate(horizontal)

                else:
                    for x in range(0, filas):
                        vectorFotos[x] = cv2.resize(vectorFotos[x], (tamanoAncho,
tamanoAlto), None, escala)
                        if len(vectorFotos[x].shape) == 2:
                            vectorFotos[x] = cv2.cvtColor(vectorFotos[x],
cv2.COLOR_GRAY2BGR)
                    horizontal = np.hstack(vectorFotos)
                    horizontal_concatenada = np.concatenate(vectorFotos)
                    vertical = horizontal

                # Para poner letras a las imágenes
                if len(nombres) != 0:
                    anchoImagen = int(vertical.shape[1] / columnas)
                    altoImagen = int(vertical.shape[0] / filas)
                    # print(altoImagen)
                    for d in range(0, filas):
                        for c in range(0, columnas):
                            cv2.rectangle(vertical, (c * anchoImagen, altoImagen
* d),
                                          (c * anchoImagen + len(nombres[d][c]) *
13 + 27, 30 + altoImagen * d),
                                          (255, 255, 255), cv2.FILLED)
                            cv2.putText(vertical, nombres[d][c], (anchoImagen * c
+ 10, altoImagen * d + 20),
                                        cv2.FONT_HERSHEY_COMPLEX, 0.7, (255, 0,
255), 2)

                return vertical

            # Variables necesarias
            nombres = []  # Donde se van a almacenar los nombres de las personas
registradas
            fotos = []  # Donde se van a almacenar las fotos que se van a apilar

            # Cogemos los nombres de las personas registradas
            for file in os.listdir("Resources/FaceRecognition"):
                try:
                    # Extraemos el nombre
                    nombres.append(file.replace(".jpg", ""))
                except Exception as e:
                    pass

            # Almacenamos las imágenes en un vector de fotos
            directorio = 'Resources/FaceRecognition'
```

```
            onlyfiles = [f for f in listdir(directorio) if
isfile(join(directorio, f))]
            fotos = np.empty(len(onlyfiles), dtype=object)
            for n in range(0, len(onlyfiles)):
                fotos[n] = cv2.imread(join(directorio, onlyfiles[n]))

            imagenBlanco = np.zeros((200, 200), np.uint8)

            if len(fotos) == 1:
                imagen_apilada = apilar_imagenes(0.5, ([fotos[0]]), nombres)

            elif len(fotos) == 2:
                imagen_apilada = apilar_imagenes(0.1, ([fotos[0], fotos[1]]),
nombres)

            elif len(fotos) == 3:
                imagen_apilada = apilar_imagenes(0.1, ([fotos[0], fotos[1]],
[fotos[2], imagenBlanco]), nombres)

            elif len(fotos) == 4:
                imagen_apilada = apilar_imagenes(0.1, ([fotos[0], fotos[1]],
[fotos[2], fotos[3]]), nombres)

            elif len(fotos) == 5:
                imagen_apilada = apilar_imagenes(0.1,
                                                 ([fotos[0], fotos[1], fotos[2]],
[fotos[3], fotos[4], imagenBlanco]),
                                                 nombres)

            elif len(fotos) == 6:
                imagen_apilada = apilar_imagenes(0.1, ([fotos[0], fotos[1],
fotos[2]], [fotos[1], fotos[2], fotos[3]]),
                                                 nombres)

            while True:
                cv2.imshow("Perfiles configurados", imagen_apilada)
                if cv2.waitKey(1) & 0xFF == ord(
                        's'):  # M antenemos la ventana abierta hasta pulsar la
tecla s y así tomar una foto
                    print('Elija a quién quiere borrar de los perfiles
configurados')
                    cv2.destroyAllWindows()
                    break  # Para salir del bucle while

            print("Acaba de visualizar a:")
            for i in range(len(nombres)):
                print(nombres[i])  # Mostramos por pantalla el nombre

            eliminar = input('\nSi desea eliminar una de las personas indicadas
anteriormente escriba:\nSI\nNO')
```

```
            if eliminar == 'SI' or eliminar == 'si':
                print('Escriba a quién desea eliminar:')
                for i in range(len(nombres)):
                    print(nombres[i])  # Mostramos por pantalla el nombre
                nombre_eliminar = input()
                os.remove('Resources/FaceRecognition/' + nombre_eliminar +
'.jpg')
                print(nombre_eliminar + 'se ha eliminado correctamente')


@app.route('/')  # NOTA: La función que hay a continuación de esto está atada y
se va a ejecutar
def index():
    return render_template('index.html')


def gen():
    while True:
        if reconocimiento_facial:
            frame = video.converted_image
        else:
            frame = yolo.converted_image
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')


@app.route('/video_feed')
def video_feed():
    return Response(gen(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')


# Ejecución del programa
if __name__ == '__main__':
    # Usamos multihilo para tener streaming y reconocimiento de objetos y facial
a la vez
    program = threading.Thread(target=programa_principal, args=())
    program.daemon = True
    program.start()  # Iniciamos el programa principal
    #app.run()
    app.run(host='192.168.31.35', port=5005)    #Ponemos la dirección IP de la
raspberry para poder acceder desde cualquier sitio
    # app.run(host='0.0.0.0', debug=True, threaded=True)
    print(threading.activeCount())  # Para saber el número de hilos que hay en
marcha
```

YoloDetection.py

```
# -*- coding: utf-8 -*-
import cv2
import numpy as np

#CONSTANTES necesarias
whT = 320 #width, height, Target
umbral_confianza = 0.5  #Constante límite para confianza en que sea un objeto
nmsThreshold = 0.3   #Constante límite para quitar cajas a la hora de recuadrar objetos detectados

#####IMPORTACIÓN DE CLASES#######
archivo_clases = 'Resources/Yolo/coco.names'  #Importamos y almacenamos en la variable classesFile la lista de clases que tenemos de archivo. Hay 80 clases y podemos añadir personlizadas si queremos

#Abrimos el archivo de texto (string) classesFile y extraemos toda la información
with open(archivo_clases, 'rt') as f:   #Lo abrimos con nombre de f
    classNames = f.read().rstrip('\n').split('\n')  # Leemos el archivo
#print(classNames)  #Para mostrar los nombres de todas las clases importadas
#print(len(classNames))   #Para mostrar el número de clases usando el comando len(lenth)

########USO DEL ALGORITMO YOLO########
'''
#Primero importamos los archivos del modelo. En el caso de YOLO viene en dos archivos separados
#El archivo .cfg contiene los detalles de la arquitectura del algoritmo y es un archivo de CONFIGURACIÓN que se puede abrir con cualquier editor de texto
#Si echamos un vistazo veremos un montón de parámetros que se han ajustado para ejecutar la red neuronal de YOLO
#El archivo .weights contiene los datos de entrenamiento
'''

#Uso del YOLO menos exacto
configuracion_red_yolo = 'Resources/Yolo/yolov3-tiny.cfg'  #Almacenamos en el objeto de tipo string la configuración de YOLO
yolo_weights = 'Resources/Yolo/yolov3-tiny.weights'

#CREACIÓN de la red neuronal
#dnn= Deep Neural Network
red_yolo = cv2.dnn.readNetFromDarknet(configuracion_red_yolo, yolo_weights)
#Leemos los archivos de YOLO desde Darknet
#Para configurar que vamos a usar openCV y que use la CPU
red_yolo.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
red_yolo.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

class Yolo():
    # ATRIBUTOS:
```

```
    '''
    thereisperson (variable booleana)
    img (objeto)
    capture (objeto)
    yolo_outputs
    '''

    # MÉTODOS
    # Constructor
    def __init__(self):
        print('Objeto yolo creado:')
        self.capture = object()
        self.img = object()
        self.thereisperson = None
        self.contador_yolo = 0
        self.converted_image = bytearray()

    # Definición de la función de objetos: Nos da el número de objetos detectados
    def encuentraObjetos(self, salidas, imagen):  # Necesitamos los valores de
las salidas de YOLO y la imagen

        self.thereisperson = False

        alto, ancho, canal = imagen.shape  # heights, withs, channel //Cogemos
los valores de la altura, ancho de la imagen
        # Listas para almacenar valores de interés de los objetos DETECTADOS
        bounding_box = []  # bounding box // Almacenará los valores de x,y, withs
y heights
        indices_clases = []  # Lista que tendrá todas indices_clases (todos los
índices)
        confianzas = []  # Valores de la confianza en esa clase

        # Recorremos los valores de las salidas de YOLO
        for salida in salidas:    # Recorremos cada OUTPUT. Hay 3
            for det in salida:    # Recorremos cada fila de cada OUTPUT
                valores_confianza = det[5:]  # Borramos las primeras 5 columnas
porque no son de la confianza ya que de las 85 columnas, las 5 primeras NO son de
la confianza en que sea de una clase
                indice_clase = np.argmax(valores_confianza)  # Busca el índice de
la clase
                confianza = valores_confianza[indice_clase]  # Guardamos el valor
de confianza de la clase

                if confianza > umbral_confianza:  # Si la confianza que nos da
YOLO a la salida es superior al umbral, almacenamos los valores de interés para
luego poder dibujar el recuadro
                    # Almacenamos el ancho y altura de la imagen
                    w, h = int(det[2] * ancho), int(det[2] * alto)
                    # Almacenamos las posiciones en x e y para luego poder
dibujar los recuadros
```

```
                    x, y = int((det[0] * ancho) - w / 2), int((det[1] * alto) - h
/ 2)  # Recordar que las primeras columnas son del centro
                    # append=adjuntar
                    bounding_box.append([x, y, w, h])  # Almacenamos los valores
en la lista bounding_box
                    indices_clases.append(indice_clase)
                    confianzas.append(float(confianza))
        # print(len(bounding_box))   #Nos muestra el número de objetos detectados
        indices = cv2.dnn.NMSBoxes(bounding_box, confianzas, umbral_confianza,
nmsThreshold)  # NonMaximumSupressionBoxes   Nos quita las cajas que se solapan a
la hora de detectar un objeto

        # Casos en los que no se detecta nada:  //Hay que hacer esto para el caso
que no entre en el bucle porque no haya objetos detectados
        if len(indices) == 0:
            print('No se ha detectado a ninguna persona')
            self.thereisperson = False

        # Dibujo del recuadro de detección
        for i in indices:
            i = i[0]  # Hacemos esto porque i viene con un corchete adicional
            caja = bounding_box[i]
            x, y, w, h = caja[0], caja[1], caja[2], caja[3]  # Extraemos los
valores de interés
            # Actualizamos la imagen:
            cv2.rectangle(self.img, (x, y), (x + w, y + h), (0, 0, 255), 2)  #
Recuadro
            cv2.putText(self.img,
f'{classNames[indices_clases[i]].upper()}{int(confianzas[i] * 100)}%', (x, y -
10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)  # Nombre de la clase

            # Para saber que hay una persona:
            if classNames[indices_clases[i]] == 'person':
                print('Se ha detectado a una persona')
                self.thereisperson = True
            else:
                self.thereisperson = False
                print('No se ha detectado a ninguna persona')

            # ANOTACIONES:
            # con indices_clases[i] obtenemos el índice de la clase detectada
            # print(indices_clases[i])
            # Con classNames[indices_clases[i]] obtenemos el NOMBRE de la clase
detectada
            # print(classNames[indices_clases[i]])

    def detecta_objetos(self):
        #while True:
            exito, self.img = self.capture.read()  # Nos va a dejar la imagen //
Almacenamos en la variable img la imagen del objeto cap usando la función read()
```

```
            # En la variable success nos va a indicar si se ha leído
correctamente el frame ya que cap.read() es un bool que devuelve true or false

            # Convertimos a formato blob porque con este formato lo entiende el
algoritmo. Es la única forma para meter la imagen en la red neuronal de YOLO
            # Blob significa Binary Large Object y un blob es un conjunto de
píxeles de una imagen binaria que comparten propiedades similares
(tamaño,forma(circularidad, convexidad,etc),colores)
            imagen_blob = cv2.dnn.blobFromImage(self.img, 1 / 255, (whT, whT),
[0, 0, 0], 1, crop=False)  # Usamos la función blobFromImage
            red_yolo.setInput(imagen_blob)  # Metemos el objeto blob en la red
neuronal

            nombre_capas = red_yolo.getLayerNames()  # Obtención de los nombres
de las capas de la red neuronal
            # print(layerNames)    #Nos da los nombres de TODAS las capas de la
red neuronal (imagen)
            # print(net.getUnconnectedOutLayers())  #Nos muestra por pantalla los
índices de las SALIDAS de la red neuronal
            nombres_capas_salida = [nombre_capas[i[0] - 1] for i in
red_yolo.getUnconnectedOutLayers()]  # Obtención de los nombres de las SALIDAS de
YOLO conociendo los índices de las salidas
            # Como se puede observar se llama a la función
getUnconnectedOutLayers para obtener aquellas capas que no están conectadas que
son las salidas
            #print(outputNames)    #Obtenemos los índices de las capas de SALIDA
Para Yolo_v3 normal son yolo_82, yolo_94 y yolo_106
            # Como estamos usando Yolo TINY obtenemos las salidas yolo_16 y
yolo_23

            salidas = red_yolo.forward(nombres_capas_salida)
            return salidas
```

FaceRecognition.py

```
# -*- coding: utf-8 -*-
import face_recognition   #Librería del reconocimiento facial
import cv2                #Computer Vision
import numpy as np        #Librería necesaria para manejo de vectores y matrices
import os                 #Librería que usaremos para leer los nombres de los
archivos imagen


# Cargamos las imágenes de las caras que deben ser detectadas
# Declaramos las variables donde vamos a cargar las personas y caras conocidas
personas_conocidas = []      # Nombres
imagen_conocida = []       # Objeto imagen
imagen_codificada = []    # Imagen codificada para su análisis
```

```
# Bucle para extraer el nombre y la imagen de las personas registradas
for file in os.listdir("Resources/FaceRecognition"):
    try:
        # Extraemos el nombre de la persona
        personas_conocidas.append(file.replace(".jpg", ""))
        file = os.path.join("Resources/FaceRecognition/", file)
        imagen_conocida = face_recognition.load_image_file(file)

imagen_codificada.append(face_recognition.face_encodings(imagen_conocida)[0])
    except Exception as e:
        pass

class FaceRecognition():
    # ATRIBUTOS:
    capture = object()
    recognition = object()
    ubicacion_cara = []
    cara_codificada = []
    nombres_caras = []
    name_gui = []
    procesa_frame = True
    persona_detectada = []

    # MÉTODOS
    #Constructor
    def __init__(self):
        # Inicialización de variables
        self.capture = None
        self.cuentacaras = 0
        self.converted_image = bytearray()
        self.recognition = None
        self.ubicacion_cara = []
        self.cara_codificada = []
        self.nombres_caras = []
        self.name_gui = []   # Variable cuando queremos reconocer a varias
personas
        self.procesa_frame = True
        self.persona_detectada = 'Persona desconocida'
        print('Objeto video creado:')
    '''
    No vamos a usar el destructor porque sino no se pueden hacer las lecturas
pertinentes desde el main
    #Destructor
    def __del__(self):
        print('Objeto de la clase FaceRecognition eliminado')
    '''
    # Función que nos devuelve la persona detectada en pantalla
    def detecta_cara(self):
        # Cogemos un frame del video para su posterior análisis. Hemos almacenado
el frame en el objeto captura
```

```
        exito, captura_face = self.capture.read()
        if exito == False:
            print('No se ha podido leer correctamente el frame')

        self.procesa_frame = True

        # Reescalamos el frame a una cuarta parte para que se procese mucho más
rápido
        frame_analizado = cv2.resize(captura_face, (0, 0), fx=0.25, fy=0.25)

        # Convertimos la imagen de BGR (que es lo que usa OPENCV) a RGB que es lo
que usa esta librería
        frame_analizado_rgb = frame_analizado[:, :, ::-1]

        # De primeras no se detecta a nadie y si se encuentra a alguien
posteriormente se cambia
        self.persona_detectada = 'No se detecta a nadie'

        # Only process every other frame of video to save time
        if self.procesa_frame:
            # Buscamos todas las caras del frame actual que ese está analizando
            self.ubicacion_cara =
face_recognition.face_locations(frame_analizado_rgb)
            self.cara_codificada =
face_recognition.face_encodings(frame_analizado_rgb, self.ubicacion_cara)  #
Codificamos la cara que se ha detectado para posteriormente analizarla

            self.nombres_caras = []
            for face_encoding in self.cara_codificada:
                # Comprobamos que la cara detectada coincide con una de las caras
registradas
                coincidencias = face_recognition.compare_faces(imagen_codificada,
face_encoding)
                nombre = "Desconocido"

                #if True in coincidencias:
                #    first_match_index = coincidencias.index(True)
                #    nombre = personas_conocidas[first_match_index]

                # Para cuando se detectan varias caras, cogemos aquella que se
encuentra a una menor distancia de la nueva cara
                distancia_caras =
face_recognition.face_distance(imagen_codificada, face_encoding)
                mejor_coincidencia = np.argmin(distancia_caras)
                if coincidencias[mejor_coincidencia]:
                    nombre = personas_conocidas[mejor_coincidencia]
                    self.persona_detectada = nombre  # Guardamos el nombre de la
persona detectada

                self.nombres_caras.append(nombre)
                self.name_gui = nombre
```

```
        self.procesa_frame = not self.procesa_frame

        # Mostramos en el frame el nombre de la persona
        for (arriba, derecha, abajo, izquierda), nombre in
zip(self.ubicacion_cara, self.nombres_caras):
            # Reescalamos el frame para ver el resultado con el mismo tamaño que
la imagen original
            arriba *= 4    # Multiplicamos por 4 ya que antes multiplicamos por
0,25
            derecha *= 4
            abajo *= 4
            izquierda *= 4

            # Dibujamos un rectángulo alrededor de la cara
            cv2.rectangle(captura_face, (izquierda, arriba), (derecha, abajo),
(0, 0, 255), 2)

            # Mostramos debajo de la imagen el nombre de la persona detectada
            cv2.rectangle(captura_face, (izquierda, abajo - 35), (derecha,
abajo), (0, 0, 255), cv2.FILLED)
            # Si queremos que nos ponga varias personas sustituimos nombre por
name_gui
            cv2.putText(captura_face, nombre, (izquierda + 6, abajo - 6),
cv2.FONT_HERSHEY_DUPLEX, 1.0, (255, 255, 255), 1)
            #return frame
            #Actualizamos la cuenta del número de detecciones
            self.cuentacaras += 1
            #Almacenamos el frame actualizado en el objeto video definido en el
main
            self.recognition = captura_face
            # Mostramos por pantalla el resultado obtenido:
            #cv2.imshow('Face Recognition Image', self.recognition)

        # Convertimos la imagen a un formato compatible para Flask
        exito, cara = cv2.imencode('.jpg', captura_face)
        # Convertimos la imagen en bits
        self.converted_image = cara.tobytes()

        # NOTA: Cuando no reconozca a ninguna persona no se va a actualizar
porque no habrá hecho un nuevo análisis
```

Fingerprint.py

```
# -*- coding: utf-8 -*-
# Módulo para el funcionamiento del reconocimiento de huellas dactilares
import time
import hashlib
```

```
from pyfingerprint.pyfingerprint import PyFingerprint   # Importamos la librería
de huellas

class HuellaDactilar():
    # Métodos:

    # INICIALIZACIÓN del sensor de huellas
    def __init__(self):
        self.huella_correcta = False
        try:
            self.huella = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF,
0x00000000)

            if (self.huella.verifyPassword() == False):
                raise ValueError('La contraseña del sensor es errónea')

        except Exception as e:
            print('No se ha podido inicializar el sensor de huellas')
            print('Exception message: ' + str(e))
            #exit(1)
        # Cogemos información del sensor
        print('Currently used templates: ' + str(self.huella.getTemplateCount())
+ '/' + str(self.huella.getStorageCapacity()))

    # Método para configurar una nueva huella dactilar
    def configurar_huella(self):
        try:
            print('Esperando a la huella...')
            # Se espera a la lectura de una huella
            while (self.huella.readImage() == False):
                pass

            # Convierte la imagen en un formato que pueda leer el programa
            self.huella.convertImage(0x01)

            # Comprobación de si la huella ya está registrada
            resultado = self.huella.searchTemplate()
            numero_posicion = resultado[0]

            if (numero_posicion >= 0):
                print('Ya existe esta huella y está en la posición:' +
str(numero_posicion))
                exit(0)

            print('Quite la huella del sensor')
            time.sleep(2)  # Esperamos un tiempo hasta quitar el dedo
            print('Vuelva a colocar el mismo dedo sobre el sensor')

            # Esperamos hasta que se vuelva a colocar la huella sobre el sensor y
hacemos igual que antes
            while (self.huella.readImage() == False):
```

```
                pass

            # Convierte la imagen en un formato que pueda leer el programa
            self.huella.convertImage(0x02)

            # Comparamos con la lectura anterior
            if (self.huella.compareCharacteristics() == 0):
                raise Exception('La huella es distinta')

            # Registramos la huella
            self.huella.createTemplate()

            # Almacenamos la nueva huella registrada y lo guardamos en una
posición del registro
            posicion = self.huella.storeTemplate()
            print('La huella se ha registrado correctamente')
            print('Se ha almacenado en la posición siguiente:' + str(posicion))

        except Exception as e:
            print('Error en el registro')
            print('Mensaje de excepción: ' + str(e))
            exit(1)

    # Método para la eliminación de una huella registrada
    def borrar_huella(self):
        # Recopilación de información del sensor
        print('Posiciones de huellas usadas ' +
str(self.huella.getTemplateCount()) + '/' +
str(self.huella.getStorageCapacity()))

        # Eliminación de la huella registrada
        try:
            numero_posicion = input('Introduzca el número de la huella que desea
eliminar: ')
            numero_posicion = int(numero_posicion)

            if (self.huella.deleteTemplate(numero_posicion) == True):
                print('La huella se ha eliminado correctamente')

        except Exception as e:
            print('Operation failed!')
            print('Mensaje de excepción: ' + str(e))
            exit(1)

    # Método para la comprobación de la huella
    def comprobar_huella(self):
        try:
            print('Esperando huella para su lectura:')

            # Esperamos hasta que se lea una huella
            while(self.huella.readImage() == False):
```

```
                pass

            # Convierte la imagen en un formato que pueda leer el programa
            self.huella.convertImage(0x01)

            # Buscamos si una huella registrada
            result = self.huella.searchTemplate()

            numero_posicion = result[0]
            precision = result[1]

            if (numero_posicion == -1):
                print('No se reconoce ninguna huella registrada')
                pass  # Nos salimos del reconocimiento
                #exit(0)
            else:
                self.huella_correcta = True
                print('Found template at position #' + str(numero_posicion))
                print('La precisión de la detección es: ' + str(precision))

            ## DATOS ADICIONALES

            ## Carga la huella detectada
            self.huella.loadTemplate(numero_posicion, 0x01)

            ## Descarga las características de la huella detectada
            characterics =
str(self.huella.downloadCharacteristics(0x01)).encode('utf-8')

            ## Hashes characteristics of template
            print('SHA-2 hash of template: ' +
hashlib.sha256(characterics).hexdigest())

        except Exception as e:
            print('No funciona el reconocimiento')
            print('Mensaje de excepción: ' + str(e))
            #exit(1)
```

Gmail.py

```
# -*- coding: utf-8 -*-
import smtplib   # SMTP = Simple Mail Transfer Protocol
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.message import EmailMessage  #Importamos la clase EmailMessage
# import imghdr  #Librería necesaria si vamos a convertir una imagen almacenada
```

```
class Gmail():
    # Atributos
    correo = ''
    contrasena = ''
    persona_detectada = ''

    # Métodos
    def __init__(self):
        self.correo = ''
        self.contrasena = ''
        self.persona_detectada = ''

    def __del__(self):
        print('Objeto de la clase Gmail eliminado')

    def config_correo(self):
        # Almacenamiento de los datos necesarios
        self.correo = input('Introduzca el correo que desea usar para enviar los
avisos: ')
        self.contrasena = input('Introduzca la contraseña del correo: ')
        # Almacenamos los datos
        archivo_txt = open('Resources/Gmail/datos_correo.txt', 'w')
        datos = self.correo + '\n' + self.contrasena
        archivo_txt.write(datos)
        #Cerramos el archivo
        archivo_txt.close()

    def enviar_foto(self, imagen):
        # Accedemos a la información del archivo con los datos:
        archivo_txt = open('Resources/Gmail/datos_correo.txt', 'r')  #Abrimos el
archivo con los datos
        #Guardamos los datos en sus respectivos atributos
        lista_datos = archivo_txt.readlines()  # Guarda en una lista los datos en
el archivo
        self.correo = lista_datos[0]
        self.contrasena = lista_datos[1]
        archivo_txt.close()  # Cerramos el archivo abierto para no generar
conflictos

        mensaje = MIMEMultipart('related')
        mensaje['Subject'] = 'Persona detectada'
        mensaje['From'] = self.correo
        mensaje['To'] = self.correo
        mensaje.preamble = 'Actualizacion de la camara de seguridad'

        mensaje_alternativo = MIMEMultipart('alternative')
        mensaje.attach(mensaje_alternativo)
        parte_texto = MIMEText('Se ha detectado un objeto')
        mensaje_alternativo.attach(parte_texto)
        parte_texto = MIMEText('<img src="cid:image1">', 'html')
        mensaje_alternativo.attach(parte_texto)
```

```
        mensaje_imagen = MIMEImage(imagen)
        mensaje_imagen.add_header('Content-ID', '<image1>')
        mensaje.attach(mensaje_imagen)

        # Envío del mensaje
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:  # Usamos conexión
segura SSL
            smtp.login(self.correo, self.contrasena)
            smtp.sendmail(self.correo, self.correo, mensaje.as_string())
            smtp.quit()

    def enviar_correo(self, persona, imagen):
        # Accedemos a la información del archivo con los datos:
        archivo_txt = open('Resources/Gmail/datos_correo.txt', 'r')  # Abrimos el
archivo con los datos
        # Guardamos los datos en sus respectivos atributos
        lista_datos = archivo_txt.readlines()  # Guarda en una lista los datos en
el archivo
        self.correo = lista_datos[0]
        self.contrasena = lista_datos[1]
        archivo_txt.close()  # Cerramos el archivo

        mensaje = MIMEMultipart('related')
        mensaje['Subject'] = str(persona) + ' ha accedido a casa'
        mensaje['From'] = self.correo
        mensaje['To'] = self.correo
        mensaje.preamble = 'Actualización del sistema de seguridad'

        mensaje_alternativo = MIMEMultipart('alternative')
        mensaje.attach(mensaje_alternativo)
        parte_texto = MIMEText('Se ha detectado un objeto')
        mensaje_alternativo.attach(parte_texto)
        parte_texto = MIMEText('<img src="cid:image1">', 'html')
        mensaje_alternativo.attach(parte_texto)

        mensaje_imagen = MIMEImage(imagen)
        mensaje_imagen.add_header('Content-ID', '<image1>')
        mensaje.attach(mensaje_imagen)

        # Envío del mensaje
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:  # Usamos conexión
segura SSL
            smtp.login(self.correo, self.contrasena)
            smtp.sendmail(self.correo, self.correo, mensaje.as_string())
            smtp.quit()
```

Reproductor.py

```
import pygame           # Librería para reproducir archivos de audio
from gtts import gTTS   # Librería de conversión de texto a audio de Google
import threading

# Se guardan los audios que nos interesan
def guarda(texto, indice):
    tts = gTTS(text=texto, lang='es')
    tts.save("Resources/Audio files/" + str(indice) + ".mp3")

def reproduce(indice):
    pygame.mixer.init()
    # Cargamos el archivo de audio
    pygame.mixer.music.load("Resources/Audio files/" + str(indice) + ".mp3")
    # threading.Thread(target=reproduce(indice)).start()  # Forma compacta
    audiothread = threading.Thread(target=pygame.mixer.music.play())
    #audiothread.daemon = True  # Poniendo esto NO funciona. Supuestamente es
para cuando finaliza el main esto también se pare
    audiothread.start()  # Iniciamos la reproducción del audio mientras sigue el
funcionamiento del programa
```

Credentials.py

```
dispositivos = ["Raspberry Pi 4"]

# Credenciales para MQTT
def credenciales_mqtt(nombre_dispositivo):
    #Usuarios
    usuarios = {"Raspberry Pi 4": "raspberry@nicols"}
    #Contraseña
    contrasenas = {"Raspberry Pi 4": "contraseña x"}
    try:
        usuario = usuarios[nombre_dispositivo]
        contrasena = contrasenas[nombre_dispositivo]
    except Exception:
        print("No se ha encontrado el dispositivo")
    return usuario, contrasena

# Ids que cogemos de las generadas desde la web de Smart Works
def thing_ids(nombre_dispositivo):
    #Things
    things = {"Raspberry Pi 4": "id de la Raspberry Pi"}
    thing_id = things[nombre_dispositivo]
    return thing_id

# Credenciales para trabajar con el protocolo HTTP
```

```
def credenciales_http(nombre_dispositivo):
    #Identidades de los clientes
    clients_ids = {"Raspberry Pi 4": "nicols::id de smartworks"}
    #Contraseña del client_id
    # NOTA: Tenemos que coger la contraseña en reset Secret ID
    client_secrets = {"Raspberry Pi 4": "contraseña del client_id"}

    try:
        client_id = clients_ids[nombre_dispositivo]
        client_secret = client_secrets[nombre_dispositivo]
    except Exception:
        print("No se ha encontrado el dispositivo")
    return client_id, client_secret
```

smartworks_sdk.py

```
# Se importan las librerías necesarias
import requests
import time
import json

# Función para enviar datos por HTTP
def http_post(url, payload, headers, print_out = True):
    r_post = requests.post(url, data=json.dumps(payload), headers=headers)
    if(print_out):
        print("\n URL: \n {} \n VERB: POST  \n Headers: \n {} \n Payload \n {} \n
RESPONSE: \n {}".format(url, headers, payload, r_post))
    return r_post.text

# Función para enviar datos por HTTP
def http_put(url, payload, headers, print_out=True):
    r_put = requests.put(url, data=json.dumps(payload), headers=headers)
    if(print_out):
        print("\n URL: \n {} \n VERB: PUT \n Headers: \n {} \n Payload \n {} \n
RESPONSE: \n {}".format(url, headers, payload, r_put))
    return r_put.text

# Función para recibir datos por HTTP
def http_get(url, headers, print_out=True):
    r_get = requests.get(url, headers=headers)
    if(print_out):
        print("\n URL: \n {} \n VERB: GET \n Headers: \n {} \n RESPONSE: \n
{}".format(url, headers, r_get))
    return r_get.text


def timestamp():
    now = time.localtime()
```

```
    hour = now.tm_hour
    minute = now.tm_min
    second = now.tm_sec
    print("\n{}h {}m {}s".format(hour, minute, second))


def get_smartworks_headers(ClientID, Secret, URL, scope):

    urlToken = URL + "/oauth2/token"

    client_id_0 = ClientID.split("::")[0]
    client_id_1 = ClientID.split("::")[1]

    payload =
'client_id={}%3A%3A{}&client_secret={}&grant_type=client_credentials&scope={}'.fo
rmat(client_id_0, client_id_1, Secret, scope)
    headers = {
    'Content-Type': 'application/x-www-form-urlencoded'
    }
    response = requests.request("POST", urlToken, headers=headers, data=payload)

    try:
        token = json.loads(response.text)["access_token"]
        print("Successful token: ", token)
        headers = {
        "Authorization":"Bearer " + token,
        "Content-Type": "application/json"
        }
    except Exception:
        print("Token unsuccesfully retrieved")
        headers = False

    return headers
```

datos_dispositivo.py

```
# -*- coding: utf-8 -*-
# Importación de librerías
import psutil                          # Librería para obtener información de
recursos del sistema
from gpiozero import CPUTemperature  #Librería para obtener temperatura de la cpu
de la Raspberry

datos_dispositivo = "Null"

# Función de actualización de variables
def actualiza_datos(acceso):
    global datos_dispositivo
```

```
    global info_CPU
    global disco_porcentaje
    global memoria_porcentaje
    global temperaturaCPU

    byte_to_GB = 1000000000

    # Información de temperatura de la CPU  de la Raspberry
    temperaturaCPU = CPUTemperature()
    print(temperaturaCPU.temperature)  # Mostramos por pantalla el valor de
temperatura actualizado

    # Información de la CPU
    info_CPU = psutil.cpu_percent(interval=1)

    # Información de la memoria
    memoria = psutil.virtual_memory()
    memoria_total = round(int(memoria[0]) / byte_to_GB, ndigits=2)
    memoria_porcentaje = memoria[2]
    memoria_disponible = round(int(memoria[1]) / byte_to_GB, ndigits=2)
    memoria_usada = round(int(memoria[3]) / byte_to_GB, ndigits=2)
    memoria_libre = round(int(memoria[4]) / byte_to_GB, ndigits=2)

    # Información del disco
    disco = psutil.disk_usage('/')
    disco_total = round(int(disco[0]) / byte_to_GB, ndigits=2)
    disco_usado = round(int(disco[1]) / byte_to_GB, ndigits=2)
    disco_libre = round(int(disco[2]) / byte_to_GB, ndigits=2)
    disco_porcentaje = disco[3]

    # En función de si se ha concedido acceso o no, actualizamos el parámetro
acceso
    if acceso == True:
        datos_dispositivo = {
                "cpu": str(info_CPU),
                "memoria": str(memoria_total),
                "memoria_disponible": str(memoria_disponible),
                "memoria_porcentaje": str(memoria_porcentaje),
                "memoria_usada": str(memoria_usada),
                "memoria_libre": str(memoria_libre),
                "disco": str(disco_total),
                "disco_usado": str(disco_usado),
                "disco_libre": str(disco_libre),
                "disco_porcentaje": str(disco_porcentaje),
         "temperatura_cpu": str(temperaturaCPU.temperature),
         "acceso": str(1),
        }
    else:
        datos_dispositivo = {
            "cpu": str(info_CPU),
            "memoria": str(memoria_total),
```

```
            "memoria_disponible": str(memoria_disponible),
            "memoria_porcentaje": str(memoria_porcentaje),
            "memoria_usada": str(memoria_usada),
            "memoria_libre": str(memoria_libre),
            "disco": str(disco_total),
            "disco_usado": str(disco_usado),
            "disco_libre": str(disco_libre),
            "disco_porcentaje": str(disco_porcentaje),
            "temperatura_cpu": str(temperaturaCPU.temperature),
            "acceso": str(0),
             }
```

SMARTWORKS

En este apartado se encuentran los códigos ejecutados en la plataforma SmartWorks.

Función dataupdate

```
import time
import requests
import json

API_HOST = 'https://api.swx.altairone.com'
PATH_TEMP = "/spaces/nicols/collections/raspberry/things/ thing id
/properties/tempstate"
PATH_CPU = "/spaces/nicols/collections/raspberry/things/ thing id
/properties/statecpu"
PATH="/spaces/nicols/collections/raspberry/things-status/thing id"

CLIENT_ID = "nicols::01F1DGETER8EVRM5MCH186VTCZ"
CLIENT_SECRET = "rN2AQqF3olV7zHmQpauHSHUpFq4u46"

def get_access_token():
    payload = f'grant_type=client_credentials&' \
              f'client_id={CLIENT_ID}&' \
              f'client_secret={CLIENT_SECRET}&' \
              f'scope=thing.read thing.update'

    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    response = requests.request("POST", API_HOST + "/oauth2/token",
                                headers=headers, data=payload)

    return response.json()['access_token']

def handle(req):
    headers = {"Authorization": "Bearer " + get_access_token()}
```

```
    response = requests.request("GET", API_HOST + PATH , headers=headers)
    properties=response.json()['properties']
    cpu = properties['cpu']
    temperature = properties['temp']

    if cpu >= 85:
        state_cpu = "Uso critico"
    elif 50 < cpu < 85:
        state_cpu = "Uso normal"
    else:
        state_cpu = "Bajo uso"

    headers = {"Authorization": "Bearer " + get_access_token()}
    response = requests.request("PUT", API_HOST + PATH_CPU, headers=headers,
                        json={"statecpu": state_cpu})



    if temperature >= 85:
        state = "Temperatura critica"
    elif 50 < temperature < 85:
        state = "Temperatura normal"
    else:
        state = "Temperatura baja"

    headers = {"Authorization": "Bearer " + get_access_token()}
    response = requests.request("PUT", API_HOST + PATH_TEMP, headers=headers,
                        json={"tempstate": state})

    return {
    "body": response.json(),
    "status_code": response.status_code
    }
```

Esquema de propiedades

```
{
        "actions": {
                "Reboot": {
                        "description": "Reboot the system",
                        "input": {
                                "type": "boolean"
                        },
                        "links": [
                                {
                                        "href":
"/spaces/name/collections/raspberry/things/id x/actions/Reboot"
                                }
                        ]
                },
                "TurnOnOff": {
                        "description": "Turn the system on/off",
```

```
                                "input": {
                                        "properties": {
                                                "Turn Off": {
                                                        "type": "string"
                                                }
                                        },
                                        "type": "object"
                                },
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/actions/TurnOnOff"
                                        }
                                ]
                        }
                },
                "base": "https://api.swx.altairone.com",
                "collection": "raspberry",
                "created": "2021-01-01T00:00:00-00:00",
                "description": "Thing Raspberry Pi 4 que se usará para recibir el estado de la
Raspberry y sus propiedades más importantes",
                "events": {
                        "FacialFecognition": {
                                "data": {
                                        "type": "string"
                                },
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/events/FacialFecognition"
                                        }
                                ]
                        },

                        "highTemperature": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/events/highTemperature"
                                        }
                                ]
                        },
                        "lowmemory": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/events/lowmemory"
                                        }
                                ]
                        }
                },
                "href": "/spaces/name/collections/raspberry/things/id x",
                "id":
"https://api.swx.altairone.com/spaces/nicols/collections/raspberry/things/id x",
                "links": [
                        {
                                "href": "/spaces/name/collections/raspberry/things/id
x/properties",
                                "rel": "properties"
                        },
                        {
                                "href": "/spaces/name/collections/raspberry/things/id
x/actions",
                                "rel": "actions"
                        },
                        {
                                "href": "/spaces/name/collections/raspberry/things/id
x/events",
```

```
                                "rel": "events"
                        }
                ],
                "model": {
                        "name": "",
                        "version": 0
                },
                "modified": "2021-06-28T16:15:36+00:00",
                "properties": {
                        "computerType": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/properties/computerType"
                                        }
                                ],
                                "title": "Computer Type",
                                "type": "string"
                        },
                        "cpu": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/properties/cpu"
                                        }
                                ],
                                "maximum": 100,
                                "minimum": 0,
                                "title": "CPU",
                                "type": "number",
                                "unit": "%"
                        },
                        "disk": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/properties/disk"
                                        }
                                ],
                                "maximum": 100,
                                "minimum": 0,
                                "title": "Disk",
                                "type": "number",
                                "unit": "%"
                        },
                        "memory": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/properties/memory"
                                        }
                                ],
                                "maximum": 100,
                                "minimum": 0,
                                "title": "Memory",
                                "type": "number",
                                "unit": "%"
                        },
                        "owner": {
                                "links": [
                                        {
                                                "href":
"/spaces/name/collections/raspberry/things/id x/properties/owner"
                                        }
                                ],
                                "title": "Owner",
                                "type": "string"
                        },
```

```
                    "statecpu": {
                              "links": [
                                        {
                                                  "href":
"/spaces/name/collections/raspberry/things/id x/properties/statecpu"
                                        }
                              ],
                              "title": "Estado CPU",
                              "type": "string"
                    },
                    "status": {
                              "links": [
                                        {
                                                  "href":
"/spaces/name/collections/raspberry/things/id x/properties/status"
                                        }
                              ],
                              "title": "Raspberry Status",
                              "type": "string"
                    },
                    "temp": {
                              "links": [
                                        {
                                                  "href":
"/spaces/name/collections/raspberry/things/id x/properties/temp"
                                        }
                              ],
                              "title": "Temperatura",
                              "type": "number"
                    },
                    "tempstate": {
                              "links": [
                                        {
                                                  "href":
"/spaces/name/collections/raspberry/things/id x/properties/tempstate"
                                        }
                              ],
                              "title": "Estado Temperatura",
                              "type": "string"
                    }
          },
          "space": "nombre del espacio",
          "title": "Raspberry Pi 4",
          "uid": "id"
}
```

Archivo HTML

```
<html>
  <head>
    <title>Sistema de seguridad Raspberry Pi 4</title>
  </head>
  <body>
    <h1>Sistema de seguridad Raspberry Pi 4</h1>
    <img id="bg" src="{{ url_for('video_feed') }}">
  </body>
</html>
```