Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Sistema d'alarma domèstic amb reconeixement facial i control remot

Document: Memòria

Alumne: Robert Ripoll López

Tutor: Anton Bardera Reig
Departament: Informàtica, Matemàtica Aplicada i Estadística
Àrea: Llenguatges i Sistemes Informàtics

Convocatòria (mes/any): Juny 2020

Índex

1. Introducció

Durant el 2019 a Catalunya s'han produït un total d'aproximadament 24.000 robatoris amb força a domicilis, i en aquesta quantitat no estan inclosos els furts (robatoris sense ús de força, violència ni intimidació) ni els robatoris en poblacions inferiors a 50.000 habitants (Ministerio del Interior, 2019). En canvi, a l'any anterior, el 2018, se'n van produir un total de 25.700 (Ministerio del Interior, 2018). Aquest decrement podria fer creure que els robatoris amb força a domicilis estan disminuint, però les xifres de l'any 2011 són de 21.200 robatoris violents (Ministerio del Interior, 2011), i semblaria que el nombre de delictes d'aquest tipus ha anat incrementant al llarg dels anys, a excepció del 2019 on han sigut menors que el 2018.

La majoria dels sistemes d'alarma domèstics que es comercialitzen compten amb sistemes de detecció d'intrusions, i emeten senyals sonors per avisar que s'està produint una intrusió al domicili i per intentar persuadir l'intrús. Malgrat això, aquests sistemes no són capaços d'enregistrar i identificar el rostre de l'intrús. Per tant, la tasca d'identificar l'intrús recau sobre el propietari de l'habitatge, i que tant podria donar-se el cas que es trobés al domicili com que no s'hi trobés en el moment de la intrusió. En el cas que el propietari s'hi trobés, la tasca d'identificar i descriure el rostre de l'intrús recauria sobre ell, i la majoria de cops no són capaços de descriure'l.

1.1. Motivació

Aquesta idea de projecte va sorgir arran de la intenció de l'autor per tenir coneixement teòric i implementar un cas pràctic sobre temes com la intel·ligència artificial, el reconeixement facial, l'Internet de les Coses (Iot), la *Raspberry Pi* i la seguretat domèstica, essent els tres primers temes candents en l'àmbit de la computació i els quatre últims temes no tractats durant el grau. A més a més, aquest propòsit de projecte des del primer moment ha atret l'autor, ja que no existeixen sistemes de seguretat domèstics que utilitzin reconeixement facial en l'enregistrament de vídeo.

Desafortunadament, per a l'autor també és rellevant recuperar el sentiment de seguretat que anteriorment tenia al seu domicili. Això és degut al fet que, fa aproximadament quatre mesos, va acabar-se la construcció d'un edifici adjacent al bloc de pisos on actualment habita, i aquests nous habitatges no van poder-se vendre perquè van ser ocupats il·legalment. Els ocupes que viuen en aquests blocs de pisos han accedit diversos cops a l'edifici de l'autor, i ho han fet amb la intenció d'ocupar habitatges (arribant a forçar portes i manipulant panys) i robar possessions d'alguns dels propietaris del bloc, que es trobaven a l'espai comú.

1.2. Objectius

El propòsit del projecte és desenvolupar un sistema d'alarma amb reconeixement facial domèstic i orientat a pisos, els quals només disposen d'una única via d'accés a l'habitatge: la porta principal del domicili. El sistema està orientat a pisos perquè només actuarà, a través de sensors i actuadors, a la porta principal d'entrada, i per tant no cobriria qualsevol altre possible accés secundari, com bé poden ser el pati, el jardí o la porta posterior del domicili. D'aquesta manera, en cas d'intrusió, es podrà capturar el rostre de l'intrús i, un cop capturat, podrà enviar-se al propietari perquè posteriorment pugui ensenyar el rostre a les autoritats. L'esquemàtic del funcionament del sistema d'alarma és el següent (vegeu figura 1).

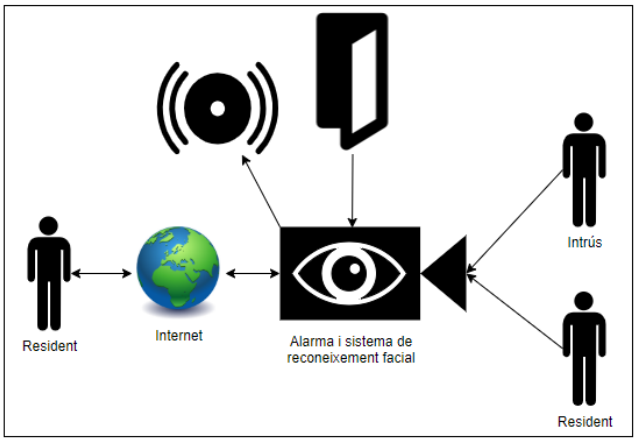


Figura 1. *Representació del sistema d'alarma, que conté els components i actors.*

El projecte té els següents objectius:

- Desenvolupar un sistema d'alarma que, a través de la lectura d'un sensor que detecti l'obertura de la porta del domicili, sigui capaç d'emetre un senyal acústic d'alarma.
- Desenvolupar un sistema de reconeixement facial, amb capacitat per detectar rostres humans i reconèixer els rostres dels ocupants del domicili, a més de ser capaç de distingir un rostre desconegut d'un dels habitants.
- Implementar un sistema de control i gestió en línia que permeti controlar, remotament i en temps real, l'estat de l'alarma i proporcionar informació sobre rostres desconeguts detectats.
- Implementar un sistema de notificacions en temps real que permeti notificar al propietari de l'habitatge de tots els esdeveniments relacionats amb la seguretat domèstica que s'hagin produït mentre no es trobava al domicili, així com rebre fotografies de les cares que es detecten amb el sistema de reconeixement facial.

2. Estudi de viabilitat

2.1. Viabilitat econòmica

Tenint en compte els següents costos: cost d'un ordinador monoplaca oscil·la entre els 30-40 €[1], cost d'un sensor magnètic es troba dins del marge de 8-12 €[2], càmera sense filtre infraroig de 8MP té un preu d'entre 20-30€[2], un brunzidor passiu té un cost dins del rang dels 2-7€[2] i un pack de cables pont pot costar entre 2€ i 4€[2], el cost estimat d'aquest projecte és de 80€. La quantitat econòmica de 80€ pot ser assumida, per tant aquest projecte sí és viable econòmicament.

En cas que el projecte es comercialitzés, caldria desenvolupar un pla de negoci que permetés cobrir els costos i també generés ingressos. El pas previ és determinar, de forma precisa, quins costos associats comporten el desenvolupament d'aquest projecte.

Tenint en compte que el total d'hores és de 375 h i que es treballaria amb una jornada laboral de 8 h dels quals 5 h són productives i el mes té 20 dies laborables, es calculen els dies i mesos necessaris per a desenvolupar aquest projecte:

$$t_{dies} = \frac{375h}{5h/dia} = 75\ dies$$

$$t_{mesos} = \frac{75\ dies}{20\ dies/mes} = 3{,}75\ mesos$$

Per aquest pla de negoci, es considera que només hi haurà un treballador, que tindrà el perfil d'enginyer de software, i serà el que desenvolupi la totalitat del projecte. Per determinar el salari que tindrà aquest treballador, és necessari establir el preu hora que percebrà. La mitjana del cost hora d'un projecte de software a mida el 2019 a Espanya se situa entre els 28,6 € - 41 € per hora (Arkbauer, 2019). El preu/hora que es prendrà serà un valor intermig: 35€/hora.

Amb l'objectiu de reduir costos de lloguer, electricitat, aigua potable, gas i connexió a Internet, com a lloc de treball s'ha escollit un centre de *coworking*. La mitjana de preus mensuals dels espais de *coworking* a Girona trobats estan dins del rang de 211 € a 218 € mensuals. S'utilitzarà el valor mitjà de 214,5 €/mes per concretar el cost de l'espai de treball.

Havent determinat i explicat els costos de personal i de l'espai de treball, a continuació hi ha una taula on s'exposa cadascun dels costos (vegeu taula 1).

[1] Consulteu l'annex 1 per veure comparativa d'ordinadors monoplaca del fabricant *Raspberry Pi*.
[2] Consulteu la taula 1 per veure els preus dels components.

Taula 1. Desglossi dels costos directes i indirectes d'aquest projecte.

Concepte	Cost
Costos directes	**13.230 €**
Cost de personal	375 h x 35 €/hora = 13.125 €
Ordinador monoplaca (*Raspberry Pi 3 Model A+*)	35 €
Sensor magnètic (*Fdit*)	15 €
Càmera No-IR 8MP (*Raspberry Pi NoIR Camera V2*)	32 €
Brunzidor passiu (*FTVOGUE 3-24V*)	6 €
Cables pont (*Neuftech 40x 20cm Jumper Dupont Cables*)	4 €
Font d'alimentació (*Gameware Fuente de Alimentación 2,5A*)	12 €
Costos indirectes	**1.570 €**
Lloc de treball	4 mesos x 214,5 €/mes = 858 €
Estació de treball (*Toshiba Tecra*)	700 €
Consumibles (papereria)	10 €
Total	**14.800 €**

Per tant, el cost total d'aquest projecte és d'uns 15.000 €. Com que la intenció és tenir un marge de beneficis d'un 20%, el que hauran d'assumir els clients seran 18.000 €. Si s'utilitza un model de negoci basat en quotes mensuals, el prototip quedaria amortitzat en 1,5 anys si 30 clients paguessin la quota de 35 €/mes.

2.2. Viabilitat tècnica

La situació excepcional de l'epidèmia ha reduït el ventall disponible de components *hardware* que s'havien d'adquirir per desenvolupar el projecte. Per aquest motiu, la placa i la càmera s'han comprat a través del distribuïdor oficial de *Raspberry Pi* anomenat *Tiendatec*, mentre que el brunzidor, el sensor magnètic i els cables pont s'han adquirit a través d'*Amazon*, mitjançant enviaments urgents d'*Amazon Prime*.

Pel que fa a l'adquisició de vídeo a través del forat de l'espiell de la porta, hi ha uns espiells digitals que poden instal·lar-se a l'orifici corresponent en substitució del tradicional. La varietat en aquest tipus d'espiells és escassa, ja que la majoria no solen disposar de connectivitat Wi-Fi, i conseqüentment no seria possible obtenir el vídeo capturat en temps real mitjançant l'ordinador monoplaca. A més a més, els espiells que sí que disposen de Wi-Fi no tenen bones prestacions quant a qualitat d'imatge (no superen 1 megapíxel), i poden arribar a costar més de 200 €. Tanmateix, existeix una alternativa viable: adquirir una

càmera endoscòpica o de petit format i col·locar-la a l'orifici de l'espiell un cop retirat aquest. Una altra opció seria la d'adquirir una càmera sense filtre infraroig que tingui unes dimensions adequades d'acord al diàmetre del forat de l'espiell de la porta.

En matèria de reconeixement facial, existeixen tècniques i algorismes de detecció de rostres i reconeixement facial amb llicència lliure, i permeten assolir els requeriments de qualitat definits, per tant en aquest aspecte és tècnicament viable.

Per a la detecció de l'obertura de porta també existeixen sensors magnètics i emissors de senyals acústics a baix cost.

Així doncs, el projecte sí que és viable tècnicament.

2.3. Viabilitat temporal (de dates)

La situació excepcional causada per l'epidèmia de la COVID-19 ha facilitat la disponibilitat personal de temps per desenvolupar aquest projecte. D'acord amb la planificació realitzada, sí que és viable «temporalment» realitzar el projecte amb 375 hores.

2.4. Viabilitat legal

L'Agència Espanyola de Protecció de Dades exclou els espiells digitals d'ús domèstic, sempre que s'utilitzin exclusivament per verificar la identitat de la persona que ha trucat el timbre. En el cas d'aquest projecte no quedaria exclòs de l'aplicació del Reglament General de Protecció de Dades, ja que s'estaria enregistrant vídeo contínuament amb la finalitat de detectar i reconèixer cares (Agencia Española de Protección de Datos, 2019). Per tant, hauríem de desenvolupar el programari de forma que contínuament detectés la presència d'un rostre humà a la imatge i intentés reconèixer rostres d'habitants. En cas de que no fos un rostre desconegut (i per tant no fos el d'un habitant), es guardaria una captura del rostre temporalment, i només s'enviaria al propietari si la porta principal d'accés a l'habitatge s'obrís.

D'acord amb el que estipula aquest reglament, aquesta càmera estaria rebent vídeo constantment de l'exterior del domicili i per tant d'una zona comuna de la comunitat de propietaris. En conseqüència, seria necessari que es realitzés el següent:

- Obtenir l'autorització dels veïns que utilitzin la zona comuna enregistrada i
- adherir un cartell indicant qui és el responsable al qual poden exercir-se els drets del reglament.

Per tant, si es realitzessin aquestes accions, el projecte seria viable legalment.

3. Metodologia

Per a la planificació i gestió del projecte s'ha utilitzat part de la guia PMBOK (Project Management Institute, Inc., 2017), que conté un conjunt de terminologies i pautes estàndard per a gestionar projectes, generalment reconegudes com a bones pràctiques. L'àmbit d'aplicació d'aquesta guia és universal, de manera que és aplicable amb qualsevol tipus de projecte, ja que s'hi defineixen fonaments universals en relació a la direcció de projectes. Aquesta guia té l'objectiu de guiar i orientar a aquells que tenen la tasca de dirigir projectes, de forma que els permeti avançar en els projectes que dirigeixen, mitjançant una sèrie de passos per assolir els resultats i objectius proposats.

Aquesta guia identifica 5 cicles de vida amb els quals es troba qualsevol projecte:

1. **Inici:** en aquest cicle de vida és quan es planteja el problema que s'ha de resoldre amb el desenvolupament del projecte. Es defineix un nou projecte o una nova fase d'execució del projecte, mitjançant l'acta de constitució del projecte. A l'acta s'exposarà la solució que es planteja resoldre amb el desenvolupament del projecte. En aquest projecte l'acta de constitució del projecte és el full del TFG, document en el qual es plantegen els antecedents, l'objecte i l'abast.
2. **Planificació:** s'estableixen objectius que volen assolir-se en desenvolupar el projecte, juntament amb la definició d'estratègies per garantir el compliment d'aquests objectius. La guia defineix que en aquesta fase s'ha de desenvolupar el pla de direcció del projecte, gestionar i definir l'abast, recopilar requisits, definir les activitats i planificar l'execució d'aquestes (incloent-ne la seqüència), estimar la duració de les activitats i desenvolupar un cronograma que defineixi l'execució d'aquestes activitats. Aquests criteris han sigut utilitzats en aquest projecte a l'hora d'establir els requisits i definir els paquets de treball, que són agrupacions d'activitats del mateix tipus, i que en la majoria de casos han arribat a ser components del projecte (com per exemple, el paquet de treball de reconeixement facial). De les activitats agrupades per aquests paquets, s'ha estimat la duració de cadascuna d'aquestes, se n'ha planificat l'execució i s'ha dissenyat un cronograma que estableix la seqüència d'execució de les activitats.
3. **Execució:** realització de les activitats proposades seguint l'estratègia definida a la fase de planificació. En aquest cicle és quan s'hauran de manifestar els problemes sorgits, a més d'exposar les solucions trobades que resolen aquests problemes.
4. **Control i monitoratge:** se supervisa i avalua el compliment dels objectius proposats al cicle de planificació, per tal d'evitar problemes no previstos i d'obtenir un projecte que no compleixi amb els objectius proposats.

5. **Tancament:** es tanca el projecte i es revisen els objectius que inicialment s'havien proposat, amb la finalitat d'avaluar que aquests s'hagin assolit. Addicionalment, també es defineixen futures millores, per tal d'implementar-les en execucions posteriors del projecte.

Per al desenvolupament del projecte s'ha aplicat la metodologia Agile, una metodologia de treball que consisteix a fragmentar el projecte en components que s'han de completar en un termini de temps prefixat (per exemple, dues setmanes), i està basada en un desenvolupament incremental i iteratiu (Tena, 2018). Per components s'entén com a aquell conjunt de fragments amb els quals un projecte pot dividir-se, com podrien ser els paquets de treball.

El desenvolupament iteratiu es basa en l'entrega d'aquests fragments del projecte al client, de forma que aquest pugui revisar-los i l'equip de desenvolupament del projecte pugui anar-los modificant d'acord amb el feedback que el client aporta vers aquests components. A continuació es representa gràficament el procés iteratiu d'aquesta metodologia (vegeu figura 2).

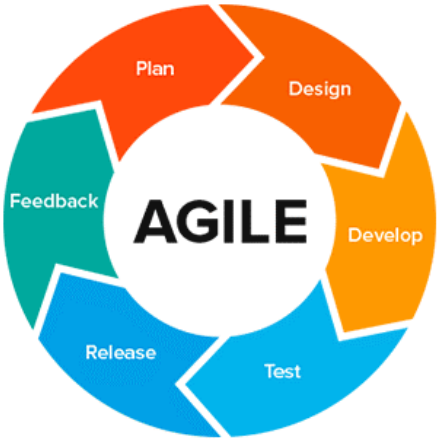


Figura 2. *Representació gràfica del procés iteratiu de la metodologia Agile.* Autor: *SPISoftware*. Extret de *SPI and Agile Software Development*. Font: https://static.wixstatic.com/media/d107d2_6c8477f3def342da814ad81690306a0e~mv2.png/v1/fit/w_300,h_300,al_c,q_5/file.png

Durant el desenvolupament del projecte s'han produït desviacions, concretament a l'hora d'escollir, adquirir i provar la càmera que s'havia d'utilitzar per al reconeixement facial. Per aquest motiu, el termini de temps del paquet de treball de l'adquisició dels components s'ha vist repercutit, i conseqüentment no s'ha pogut completar aquest paquet amb el termini de temps prefixat inicialment. Aquest fet també ha provocat que hi hagin endarreriments en les entregues dels demés paquets de treball. En apartats posteriors s'expliquen els problemes apareguts durant l'adquisició de la càmera, i les solucions que s'han plantejat per tal d'avançar amb el desenvolupament del projecte.

4. Marc de treball i conceptes previs

Aquest apartat explica conceptes teòrics i eines utilitzades a la fase de desenvolupament del projecte, amb l'objectiu d'explicar-les per facilitar la comprensió del projecte.

4.1. Ordinador monoplaca

Els ordinadors monoplaca són ordinadors continguts en una única placa, en la que es troben tots els components que els conformen (vegeu figura 3). Aquesta placa conté el microprocessador, l'entrada/sortida, memòria RAM i altres components (Single-board computer, 2020).

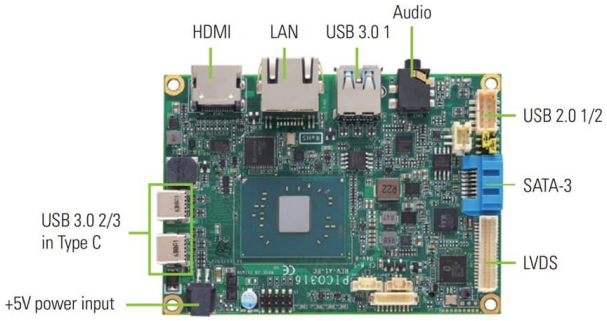


Figura 3. *Fotografia frontal de l'ordinador monoplaca Axiomtek PICO316.* Extret de *Mouser*. Font: https://www.mouser.es/images/marketingid/2018/microsites/130721827/PICO316_Board_Front.png

Alguns models monoplaca disposen de ports d'entrada i sortida que permeten a l'ordinador comunicar-se amb altres components electrònics, de forma programàtica. Els components que s'hi solen connectar són sensors (per exemple: termòmetre, sensor de moviment, sensor de vibració, etc.) i actuadors (per exemple: LED, brunzidor, motor, etc.).

La diferència d'aquesta tipologia d'ordinadors amb els ordinadors tradicionals d'escriptori és que normalment els monoplaca no disposen de ranures per perifèrics o expansió. Una altra diferència rellevant és l'eficiència energètica i el nivell de soroll emès pels ordinadors monoplaca, ja que els ordinadors tradicionals funcionen amb fonts d'alimentació més potents (conseqüentment, més consum energètic) i emeten més soroll (a causa dels discs durs HDD i dels ventiladors) (Techopedia, 2017).

4.1.1. *Raspberry Pi*

D'acord amb la informació que proporciona el fabricant (Raspberry Pi Foundation), la *Raspberry P*i és un ordinador de baix cost, i de la mida d'una targeta de crèdit que es connecta a un monitor, utilitza un teclat estàndard i un ratolí (vegeu figura 4). L'objectiu d'aquest ordinador és el de promoure l'educació en l'àmbit de la ciència de la computació (i altres àmbits associats) tant a adults com a infants.

Tot i que l'objectiu del fabricant és el de promoure l'educació i facilitar ordinadors de baix cost a països del tercer món, és un model d'ordinador monoplaca que disposa de ports d'entrada i sortida genèrics,

i ens permetran rebre informació de sensors (en el cas d'aquest projecte, del sensor d'obertura de porta i de la càmera), i enviar senyals a actuadors (s'ha utilitzat un brunzidor per emetre senyals acústics). A més a més, també disposa de connexió a Internet, que permetrà enviar informació sobre els esdeveniments produïts als propietaris, com també permetre als residents gestionar l'estat de l'alarma.

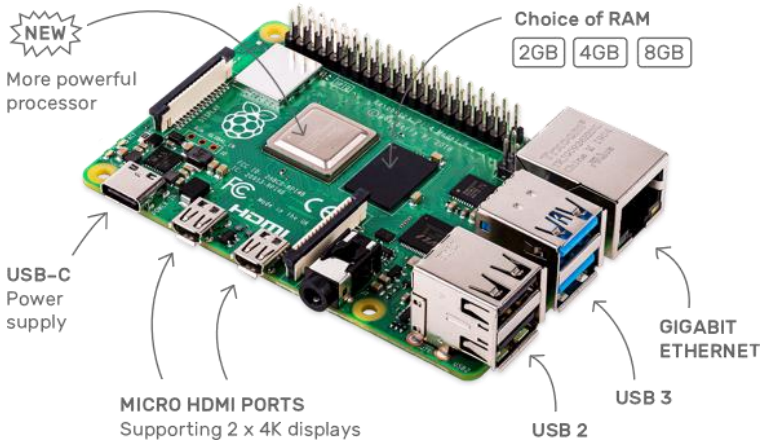


Figura 4. *Imatge de la Raspberry Pi 4 Model B.* Extret de *Raspberry Pi*. Font: https://www.raspberrypi.org/homepage-9df4b/static/raspberry-pi-4-labelled-2857741801afdf1cabeaa58325e07b58.png

4.2. Càmera

A continuació s'explicaran les tipologies de càmeres que s'han valorat pel desenvolupament del sistema de reconeixement facial.

4.2.1. Càmera borescòpica

Les càmeres borescòpiques (vegeu figura 5) són una tipologia de càmera orientada a la inspecció no destructiva de llocs de difícil accés (gairebé inaccessibles). La característica principal que les distingeix de les càmeres principals són les dimensions, ja que disposen de lents de diàmetre petit (normalment menors a 20 mm) i d'una llargada del cablejat considerable, per tal de facilitar tasques d'inspecció (De Máquinas y Herramientas, 2013).

Figura 5. *Fotografia d'una càmera borescòpica.* Extret de *Superelectrónica.* Font: https://superelectronica.es/2490-large_default/endoscopio-usb-20m-camara-inspeccion-fontaneria-9mm.jpg

Inicialment van desenvolupar-se per a diagnòstic mèdic (PCE Instruments), i les seves aplicacions s'han acabat extrapolant a l'àmbit industrial (revisió de soldadures), mecatrònic, armamentístic (inspecció de l'interior d'armes) i de seguretat (comprovació de l'existència d'explosius o estupefaents en vehicles).

Ja que aquestes càmeres estan fabricades amb la finalitat d'inspeccionar llocs estrets i de difícil accés, això pot perfectament aplicar-se a l'orifici de l'espiell d'una porta (és circular, estret i es troba dins de la porta). És per això que en aquest projecte s'ha plantejat la idea d'utilitzar una càmera d'aquest tipus (vídeoboroscopi), per així poder captar vídeo en temps real de què està succeint a l'exterior de l'habitatge (en aquest cas, al passadís comunitari) i capturar i identificar rostres humans (és a dir, potencials intrusos). D'aquesta manera no seria necessari desmuntar l'espiell ja instal·lat a la porta del domicili, ja que el vídeoboroscopi es trobaria dins del mateix espiell.

4.2.2. Càmera sense filtre infraroig

Les càmeres tradicionals disposen d'un filtre de raigs infraroigs per tal de limitar l'espectre de color capturat per l'objectiu, i d'aquesta manera estan acotades a la llum visible com la visió humana (vegeu figura 6). Oposadament, les càmeres sense filtre infraroig són aquelles càmeres que no només capturen la llum visible, sinó que també capturen l'espectre que inclou la llum ultraviolada i el rang d'espectre infraroig (al voltant dels 880 nm), que no és visible pels humans (ArduCam, 2020).

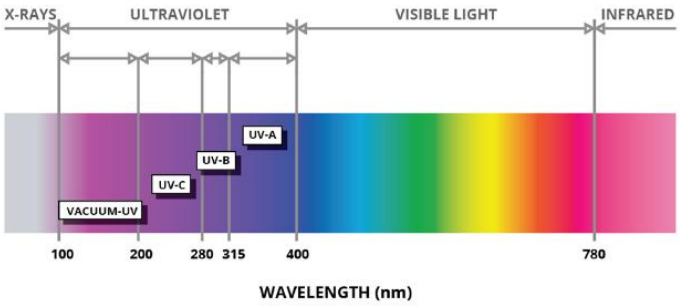


Figura 6. *Espectre electromagnètic, en el que s'indica quin rang és visible per a l'ull humà.* Extret de *Azooptics*. Font: https://www.azooptics.com/image.axd?src=%2fimages%2fArticle_Images%2fImageForArticle_1087(1).png&ts=20160208021842&ri=640

Ja que el llindar de color cobert per aquest tipus de càmeres és més extens, són ideals per a condicions de baixa lluminositat (en el cas d'aquest projecte s'hi ajusta perfectament, ja que l'objectiu és poder tenir visió tant en baixa lluminositat com en condicions de llum normals), a més de seguir sent usables en condicions de lluminositat òptima o normal (malgrat que les imatges capturades tinguin un to vermellós). A continuació, dues fotografies comparatives de l'espectre de color capturat per una càmera tradicional respecte una càmera sense filtre d'infraroig (vegeu figura 7).

Figura 7. *A l'esquerra, fotografia capturada amb una càmera estàndard; a la dreta, fotografia capturada amb una càmera sense filtre IR.* Autor: *Alex Eames.* Extret de *Comparison of RasPiCam and Pi NoIR output in daylight.* Font: https://raspi.tv/2013/pinoir-whats-it-for-comparison-of-raspicam-and-pi-noir-output-in-daylight

4.3. Interruptor magnètic

Aquest tipus d'interruptors requereixen l'ús d'un imant mòbil, que és detectat per l'interruptor. Quan aquest interruptor detecta la presència del camp magnètic que emet l'imant, obre o tanca un circuit elèctric. En el cas dels interruptors *reed*, si la peça que conté l'imant es troba en una posició propera a l'interruptor, el circuit contingut dins d'aquest interruptor es tanca.

L'interruptor està format per dues làmines ferromagnètiques, hermèticament segellades en una càpsula de vidre (vegeu figura 8), i quan un imant s'aproxima a l'interruptor, provoca que ambdues làmines que es troben dins d'aquest entrin en contacte. Mentre es toquen, les làmines tanquen els contactes normalment oberts, fet que permet que circuli l'electricitat. Un imant que s'apropi a l'interruptor provocarà que es desconnecti el circuit (Arrow, 2018).

Figura 8. *Interruptor "reed" normalment obert.* Extret de *Arrow*. Font: https://static4.arrow.com/-/media/arrow/images/miscellaneous/0/0818_reed_2.jpg?la=en&hash=62EE58EAFE9BE93FEA3BF0CADBB1943CB9BB89F7

4.4. Brunzidor

Un brunzidor és un component elèctric que transforma electricitat en un so o brunzit intermitent o continu d'un mateix to (vegeu figura 9).

Figura 9. *Imatge il·lustrativa del brunzidor Multicomp MCKPR3-G4210-4136.* Extret de *Farnell*. Font: https://es.farnell.com/productimages/standard/en_US/4682931.jpg

Hi ha dos tipus de brunzidors: els actius i els passius. Els brunzidors actius disposen d'una font d'oscil·lació interna, i per tant sempre emeten so de la mateixa intensitat i to. En canvi, els passius no disposen d'una font d'oscil·lació interna, i en requereixen una d'externa. Com que requereixen una oscil·lació externa, és possible modificar el to i la intensitat del so que emet. (Manorshi, 2019)

No obstant això, per aquest projecte no és necessari aquesta «personalització» del so emès pel brunzidor, ja que només s'utilitzarà per emetre senyals acústics d'avís que s'està produint una intrusió al domicili. Per aquest motiu, s'ha escollit un brunzidor actiu.

4.5. Aprenentatge automàtic (*machine learning*)

És una branca de la intel·ligència artificial que té l'objectiu de dissenyar i utilitzar sistemes que identifiquen patrons ocults i complexos en conjunts molt grans de dades (González). A partir d'aquests patrons, els sistemes d'aprenentatge automàtic poden elaborar prediccions per a casos totalment nous (és a dir, pels quals no han sigut entrenats).

4.5.1. Aprenentatge supervisat

És una subcategoria d'aprenentatge automàtic que consisteix a etiquetar les dades, s'entrenen aquests algorismes de *machine learning* (vegeu figura 10). La finalitat és que l'algorisme sigui capaç de trobar una funció que, a partir de les dades d'entrada se'ls assignin etiquetes (Santos, 2017).

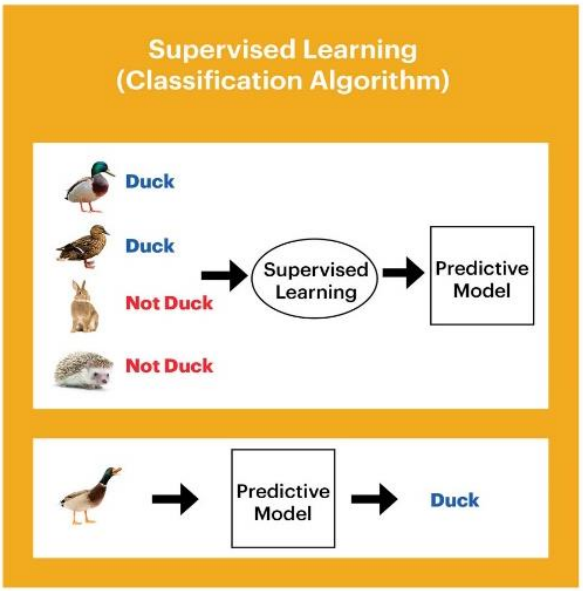


Figura 10. *Exemple de funcionament d'un algorisme de classificació, que forma part de la categoria d'aprenentatge supervisat.* Extret de *Western Digital*. Font: https://blog.westerndigital.com/wp-content/uploads/2018/05/supervised-learning-diagram.jpg

4.5.2. Classificadors

La classificació és una tasca de la categoria d'aprenentatge supervisat que consisteix a assignar una classe o una altra a un objecte (vegeu figura 11), en funció de les característiques o atributs d'aquest (Acebo, 2019).

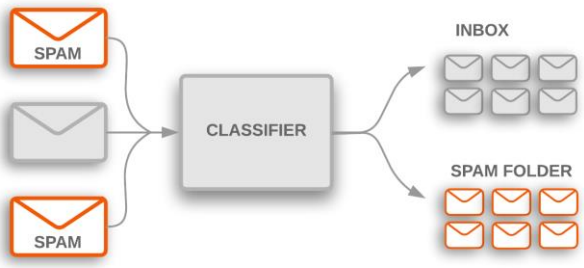


Figura 11. *Classificació de correus en desitjats i no desitjats (correu brossa).* Extret de *Google Developers*. Font: https://developers.google.com/machine-learning/guides/text-classification/images/TextClassificationExample.png

4.5.2.1. KNN (K-Nearest Neighbours)

És un algorisme de classificació simple que s'encarrega d'assignar una classe a aquells objectes no classificats, basant-se amb la classe dels objectes classificats més propers (Acebo, 2019). A continuació, l'algorisme KNN (vegeu figura 12) i un exemple d'execució de l'algorisme KNN sobre un conjunt de dades de dues categories A i B (vegeu figura 13).

```
Entrenament o aprenentatge :
    Simplement memoritzar tots els exemples del
    conjunt d'entrenament.

Execució :
    Donat un nou vector d'entrada X que es vol
    classificar:
        Trobar els K exemples que es troben a menor
        distància de X
        Assignar a X la classe a la que pertanyen
        la majoria d'aquests K exemples*
```

Figura 12. *Algorisme KNN.* Font: https://moodle2.udg.edu/mod/resource/view.php?id=730226

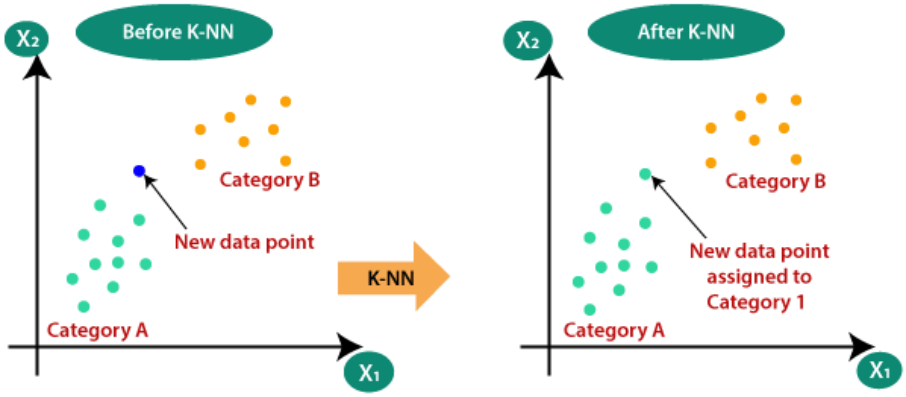


Figura 13. *Exemple d'execució de l'algorisme KNN sobre un conjunt de dades de dos categories A i B, en el que toca assignar una classe a un punt nou no classificat.* Extret de *JavaTpoint.* Font: https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning2.png

4.7. Reconeixement facial

El reconeixement facial és un tipus de sistema amb capacitat per identificar, de forma autònoma, rostres humans en imatges capturades digitalment. Aquests sistemes analitzen els trets facials que presenten els individus que apareixen a les imatges analitzades, amb la finalitat de comparar aquests trets amb els que es troben a la base de dades de rostres reconeguts (Wikipedia, 2020).

Els passos que s'han de dur a terme per tal de reconèixer el rostre d'un individu són els següents (Geitgey, Modern Face Recognition with Deep Learning, 2016):

1. **Analitzar la imatge per trobar rostres humans.**

En aquest pas s'utilitza una tècnica anomenada Histograma de Gradients Orientats (en anglès, *HOG*). Es comença transformant la fotografia a blanc i negre. Seguidament, es fragmenta la imatge en blocs de 16x16 píxels. Es processa cadascun d'aquests blocs i es detecta com de fosc és el bloc processat respecte a

els altres blocs que l'envolten. Després de realitzar aquesta comparació de foscor, es reemplaça el bloc per una fletxa que indica la direcció en què la imatge es va enfosquint. El que s'haurà obtingut amb aquest procediment és l'estructura bàsica d'una cara (vegeu figura 14):

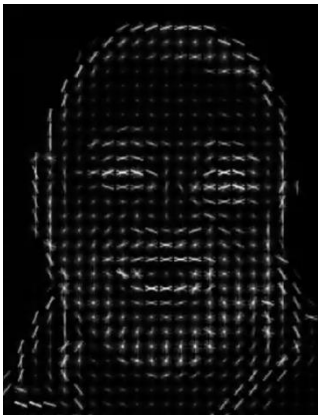


Figura 14. *Estructura bàsica d'un rostre utilitzant la tècnica HOG.* Autor: *Rehan Ahmad*. Extret de *Analytics Vidhya.* Font: https://miro.medium.com/max/1400/1*gWeh90aKfv7A-bpc0vC0oQ.png

2. Per cada rostre trobat, analitzar els trets característics que el fan únic.

En aquest pas és possible que calgui aplicar transformacions al rostre, ja que pot ser que aparegui orientada cap a una direcció (és a dir, que no miri directament a la càmera) o bé hi hagi mala il·luminació.

Si és necessari, la transformació que es realitzarà serà la de deformar la imatge perquè els ulls i els llavis sempre es trobin en la mateixa posició. Per fer això, s'utilitzarà l'algorisme d'estimació de punts de referència de la cara, que consisteix trobar els 68 punts específics que existeixen en tot rostre (la part superior de la barbeta, la vora de cada ull, la vora de cada cella, etc.). A partir d'aquests 68 punts (vegeu figura 15), s'entrenarà un algorisme de *machine learning* perquè trobi aquests 68 punts específics en qualsevol rostre:

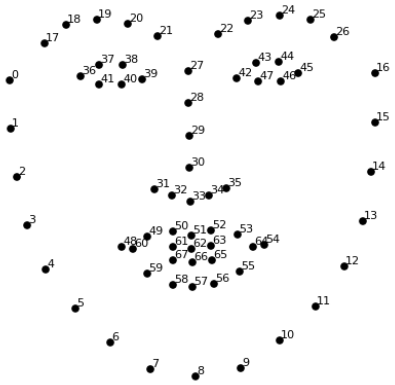


Figura 15. *Els 68 punts específics que existeixen en qualsevol rostre.* Autor: *Brandon Amos*. Extret de *Adam Geitgey*. Font: https://miro.medium.com/max/828/1*AbEg31EgkbXSQehuNJBlWg.png

Després d'haver esbrinat la posició dels ulls i la boca, es rotarà, escalarà i retallarà el rostre, de manera que el resultat final sigui la imatge del rostre amb els ulls i boca el més centrats possibles (vegeu figura 16).

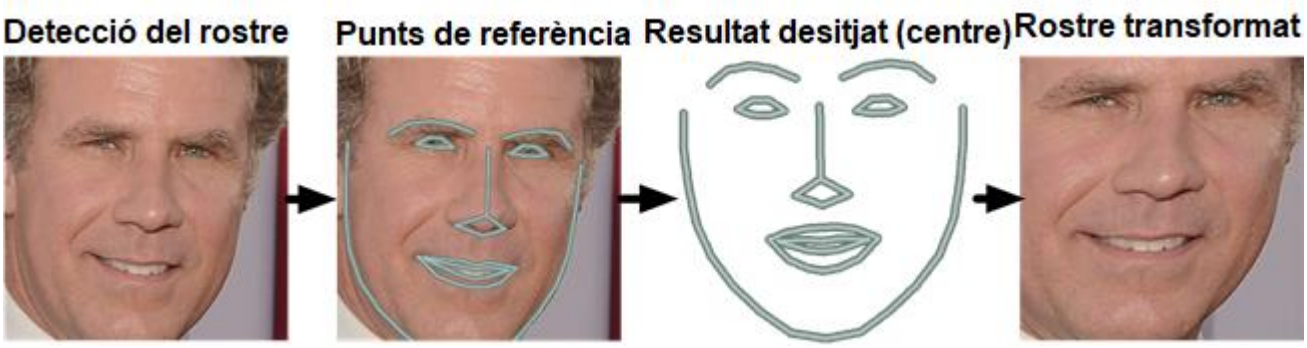


Figura 16. *Procés de transformació del rostre per centrar boca i ulls.* Autor: *Adam Geitgey*. Extret de *Modern Face Recognition with Deep Learning.* Font: https://miro.medium.com/max/1400/1*igEzGcFn-tjZb94j15tCNA.png

3. Distingir els rostres. És a dir, identificar de forma única una cara.

La solució d'aquest pas és la d'entrenar una *Convolutional Neural Network*[3], amb l'objectiu que trobi 128 mesures per a cada rostre identificat. El procés d'entrenament és el següent:

1. Carregar una imatge d'entrenament del rostre d'una persona coneguda.
2. Carregar una altra imatge de la mateixa persona.
3. Carregar una imatge d'una persona totalment diferent.

L'algorisme comprovarà quins mesuraments està generant per cadascuna de les tres imatges, i ajustarà lleugerament la xarxa neuronal perquè les mesures generades de la imatge 1 i 2 siguin similars i les mesures obtingudes a partir de la imatge 2 i 3 siguin llunyanes.

Després de repetir aquest procediment per milions de cops amb milions d'imatges de milers de persones, la xarxa neuronal serà capaç de generar, de forma fiable, 128 mesures per a cadascuna de les persones.

Afortunadament, a Internet existeixen xarxes neuronals preentrenades (Amos, Ludwiczuk, & Satyanarayanan, 2016) que poden descarregar-se gratuïtament. Només fa falta executar les imatges dels rostres que es volen reconèixer a través d'aquestes xarxes i s'obtindran 128 mesuraments.

4. Predir el nom de la persona.

En aquest pas es pot utilitzar qualsevol algorisme de classificació de *machine learning*. L'entrada d'aquest classificador seran els mesuraments d'una imatge amb el rostre d'una persona, i en qüestió de

[3] Consulteu l'annex 2 per llegir què és una *Convolutional Neural Network*.

mil·lisegons, la sortida serà el nom de la persona que més s'ajusta als mesuraments d'entrada, a més de retornar un percentatge de confiança sobre aquesta predicció (que representa la incertesa de la persona predita).

4.8. Paral·lelisme

La computació paral·lela és un paradigma que permet l'execució de diverses tasques de forma simultània i paral·lela, és a dir executar diverses tasques a la vegada. Aquest tipus de paradigma s'utilitza en computadors que disposen d'un processador multicore, un circuit integrat d'un processador que disposa de diverses unitats de processament (cores), les quals cadascuna d'elles executen i llegeixen instruccions (Wikipedia, 2020).

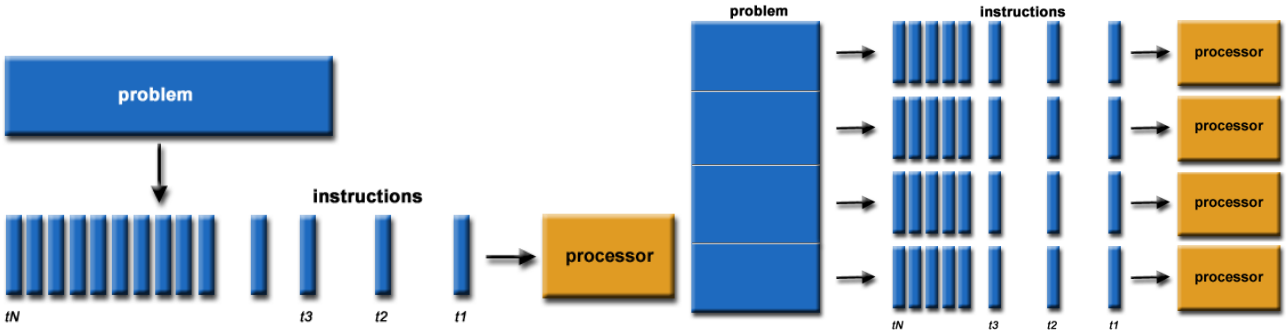


Figura 17. *Comparativa entre paradigmes de computació (a l'esquerra, computació seqüencial; a la dreta computació paral·lela).* Font: https://computing.llnl.gov/tutorials/parallel_comp/images/serialProblem.gif i https://computing.llnl.gov/tutorials/parallel_comp/images/parallelProblem.gif

La finalitat d'aquest paradigma és la d'aprofitar cada unitat de processament, de forma que simultàniament s'executa una tasca a cada core. Amb altres paraules, l'execució d'una tasca es realitza a una unitat, una altra tasca a una altra unitat de processament i així successivament.

Aquest paradigma s'ha utilitzat en el desenvolupament del sistema d'alarma i del sistema de reconeixement facial, de forma que es pogués donar ús als 4 cores dels que disposa l'ordinador monoplaca escollit. Gràcies al paradigma de computació paral·lela, ambdós sistemes podran rebre, processar i enviar missatges mentre executen la tasca principal (ja sigui l'alarma en si o l'algorisme de reconeixement facial) de forma ininterrompuda.

El primer sistema de classificació d'ordinadors paral·lels i seqüencials va ser creat per *Michael J. Flynn*, que realitzava aquesta classificació en funció de dues dimensions independents: flux d'instruccions i flux de dades.

La classificació d'ordinadors paral·lels era la següent (Barney, 2020):

- **SISD (únic flux d'instruccions i únic flux de dades)**
  - Ordinadors no paral·lels.
  - Només s'executa un flux d'instrucció a la CPU durant un cicle de rellotge.
  - Només s'utilitza un flux de dades com a entrada durant un cicle de rellotge.
  - Execució determinista.

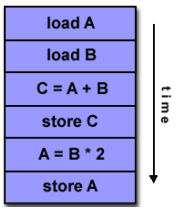


Figura 18. *Execució d'un programa en una arquitectura SISD.*

- **SIMD (únic flux d'instruccions i múltiples fluxos de dades)**
  - Totes les unitats de processament executen la mateixa instrucció durant un cicle de rellotge concret.
  - Cadascuna de les unitats de processament pot operar amb dades diferents.
  - Arquitectura més adequada per problemes d'alt grau de regularitat (processat d'imatge).

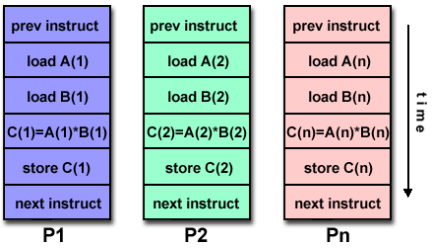


Figura 19. *Execució d'un programa en una arquitectura SIMD.*

- **MISD (múltiples fluxos d'instruccions i únic flux de dades)**
  - Cadascuna de les unitats de processament opera amb les dades de forma independent.
  - Només hi ha un flux de dades per a cada unitat de processament.
  - Un exemple d'un cas d'ús seria l'execució d'algorismes criptogràfics amb la finalitat de trencar l'encriptat d'una cadena de text per obtenir-ne el contingut original.

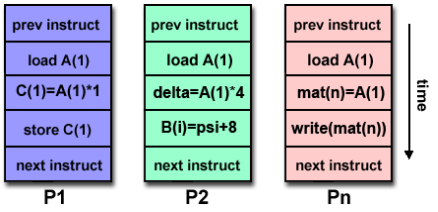


Figura 20. *Execució d'un programa en una arquitectura MISD.*

- **MIMD (múltiples fluxos d'instruccions i múltiples fluxos de dades)**
  - Cadascun dels processadors pot executar un flux diferent d'instruccions.
  - Cada processador pot treballar amb un flux diferent de dades.
  - L'execució pot ser síncrona o asíncrona, determinista o no determinista.

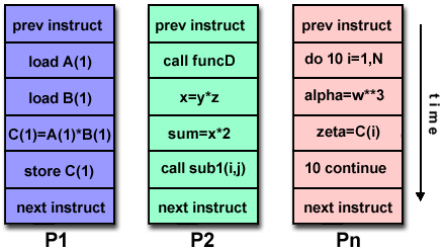


Figura 21. *Execució de diverses instruccions en una arquitectura MIMD.*

4.9. Asincronisme

L'asincronisme és un paradigma de programació que consisteix a executar una instrucció, i en comptes d'esperar el resultat de l'execució d'aquesta instrucció, s'executen les següents instruccions del programa. Un cop el resultat de l'execució de la qual no s'ha esperat el resultat finalitza, el programa rep l'esdeveniment i obté les dades que ha retornat l'execució de la instrucció (Haverbeke, 2018). Gràcies a aquest paradigma (Malik, 2019), es millora la latència (mesura de temps transcorregut per realitzar quelcom) total del programa a conseqüència de l'augment del rendiment (la quantitat de tasques executades per unitat de temps).

Com pot apreciar-se a la figura 22, s'obté la cadena de text de l'URL "http://msdn.microsoft.com" i es continua executant les instruccions posteriors, fins que arriba un punt de la funció on cal obtenir el resultat de la cadena de text per executar operacions sobre aquesta.

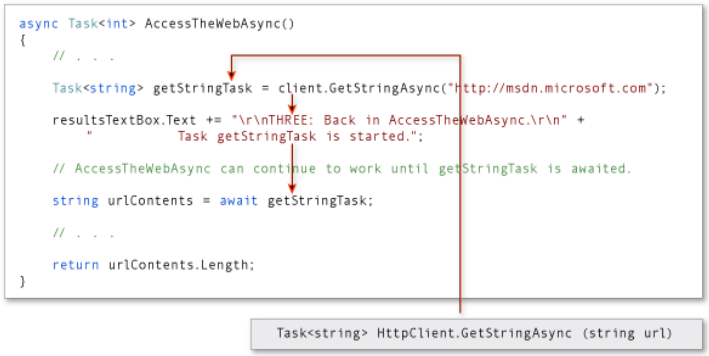


Figura 22. *Exemple d'una funció asíncrona.* Extret de *MDN*. Font: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/media/asynctrace-onetwo.png

Aquest paradigma de programació s'ha utilitzat a tots els sistemes desenvolupats (alarma, reconeixement facial, esdeveniments i portal web). Als sistemes d'alarma i reconeixement facial s'ha hagut d'usar degut a la naturalesa asíncrona de la llibreria *websockets* de *Python* emprada, que ha servit per establir comunicació amb el sistema d'esdeveniments i el portal web. Aquests dos darrers sistemes s'han implementat utilitzant el llenguatge de programació *NodeJS*, un motor d'execució que permet executar *JavaScript* fora dels navegadors, tal i com s'explica a l'apartat 7.10.

4.10. *WebSocket*

El protocol *WebSocket* va sorgir com una alternativa al funcionament tradicional del protocol HTTP, en el que es requereix la interacció del client per carregar dades noves d'un servidor. La idea darrere aquest protocol és la d'establir comunicacions persistents bidireccionals de baixa latència, utilitzant la capa de transport TCP, de forma que qualsevol de les dues estacions implicades pot enviar dades en qualsevol moment, sense necessitat que l'altre li sol·liciti explícitament (Ubl & Kitamura, 2010).

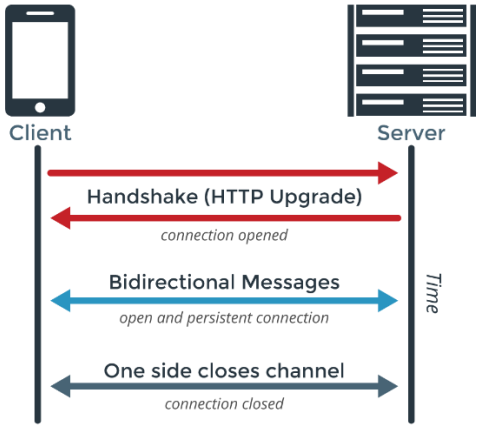


Figura 23. *Esquemàtic dels missatges intercanviats entre client i servidor en una connexió WebSocket*. Extret de *Hackplayers*. Font: https://3.bp.blogspot.com/-6uxEcnF6Vr4/WNmlbsUbN1I/AAAAAAAA5ww/ns9z4r6vOZgRt9MYrbJ9d8bFY0Y8yb-DgCLcB/s1600/websocket.png

Per establir una connexió a través d'aquest protocol, el client envia una sol·licitud al servidor per indicar-li que vol canviar del protocol HTTP a WebSocket (vegeu figura 24), i ho expressa utilitzant la capçalera "Upgrade" (WebSocket):

```
GET ws://echo.websocket.org/?encoding=text HTTP/1.1
Origin: http://websocket.org
Cookie: __utma=99as
Connection: Upgrade
Host: echo.websocket.org
Sec-WebSocket-Key: uRovscZjNol/umbTt5uKmw==
Upgrade: websocket
Sec-WebSocket-Version: 13
```

Figura 24. *Exemple d'una petició enviada per un client per sol·licitar el canvi de protocol.* Extret de *WebSocket*. Font: https://www.websocket.org/aboutwebsocket.html

Si el servidor decideix acceptar la sol·licitud de canvi de protocol, també utilitza la capçalera "Upgrade", i respon el següent (vegeu figura 25):

```
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Fri, 10 Feb 2012 17:38:18 GMT
Connection: Upgrade
Server: Kaazing Gateway
Upgrade: WebSocket
Access-Control-Allow-Origin: http://websocket.org
Access-Control-Allow-Credentials: true
Sec-WebSocket-Accept: rLHCkw/SKsO9GAH/ZSFhBATDKrU=
Access-Control-Allow-Headers: content-type
```

Figura 25. *Exemple de la resposta d'un servidor acceptant el canvi de protocol d'HTTP a WebSocket.* Extret de *WebSocket*. Font: https://www.websocket.org/aboutwebsocket.html

4.11. API REST

REST és un estil d'arquitectura utilitzat per descriure interfícies que treballen sobre el protocol HTTP. Les sigles REST signifiquen REpresentional State Transfer (transferència d'estat representacional), que consisteix en el fet que el servidor no emmagatzema dades entre peticions, és a dir, el servidor no guarda l'estat o el resultat de peticions realitzades amb anterioritat (Geeky Theory).

Aquesta arquitectura està definida per les següents característiques:

- Els dos components implicats (client i servidor) no necessiten saber el funcionament de l'altre: el client no ha de conèixer els detalls de funcionament intern del servidor i el servidor no ha de conèixer quin ús i com usa les dades que envia al client.
- No es guarda l'estat entre peticions.
- S'utilitzen els mètodes HTTP GET, POST, PUT i DELETE per realitzar operacions sobre entitats: accés, creació, actualització i esborrat.

El concepte d'API REST prové de la fusió entre el concepte explicat anteriorment i el concepte d'API. Per API s'entén qualsevol interfície utilitzada per dos sistemes amb la finalitat de comunicar-se.

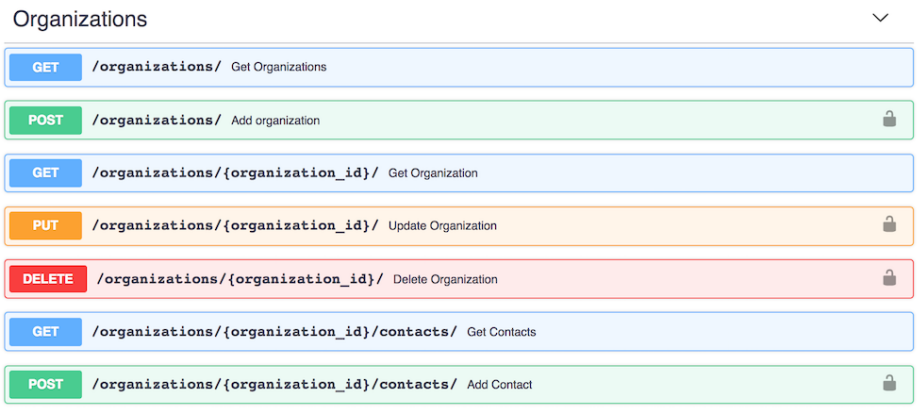


Figura 26. *Exemple d'un conjunt de rutes API REST per gestionar entitats de tipus "Organització".* Extret de *API Evangelist*. Font: https://s3.amazonaws.com/kinlane-productions/open-referral/hsda-organizations-documentation.png

4.12. Webhook

Un webhook és l'execució d'una acció desencadenada per conseqüència del succés d'un esdeveniment, i l'acció que s'executa consisteix a realitzar una petició HTTP a una API REST que emmagatzemi les dades d'aquest succés i actuï en conseqüència. És una tècnica que permet a un sistema rebre notificacions de forma automàtica a l'instant es produeixin esdeveniments, en substitució a la tècnica *long polling*, que consisteix a anar preguntant de forma periòdica (en funció d'un interval de temps prefixat) al sistema que informa dels esdeveniments (Gavrailova, 2019).

A continuació, una imatge descriptiva de la comparació entre la tècnica *long polling* i la tècnica dels *webhook* (vegeu figura 27):

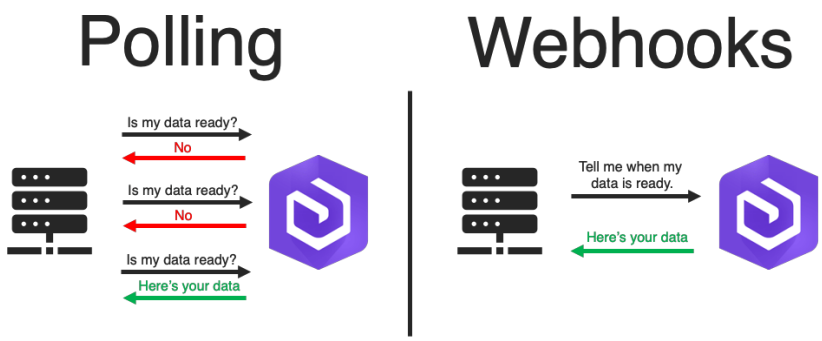


Figura 27. *Comparació gràfica entre la tècnica de long polling (a l'esquerra) i la tècnica dels webhooks (a la dreta).* Extret de *Aprendiendo Arduino*. Font: https://www.process.st/wp-content/uploads/2019/10/webhooks-vs-apis.png

4.13. Platform as a Service (PaaS)

Les plataformes com a servei (PaaS) són entorns de desenvolupament i desplegament al núvol, en les que la infraestructura i manteniment d'aquesta és proporcionada i gestionada pel proveïdor (vegeu figura 28), de manera que el desenvolupador pot centrar-se amb desenvolupar i desplegar l'aplicació que construeixi (Axarnet, 2018).

El principal avantatge d'utilitzar aquest tipus de plataformes és que l'escalabilitat és gestionada de forma automàtica per part del proveïdor, de manera que s'utilitzen més recursos de forma transparent si així fos convenient. L'única tasca del desenvolupador és la d'assegurar-se que l'aplicació que s'executa a la plataforma sigui el més òptim possible per utilitzar el mínim possible de recursos (Rodríguez, 2012).

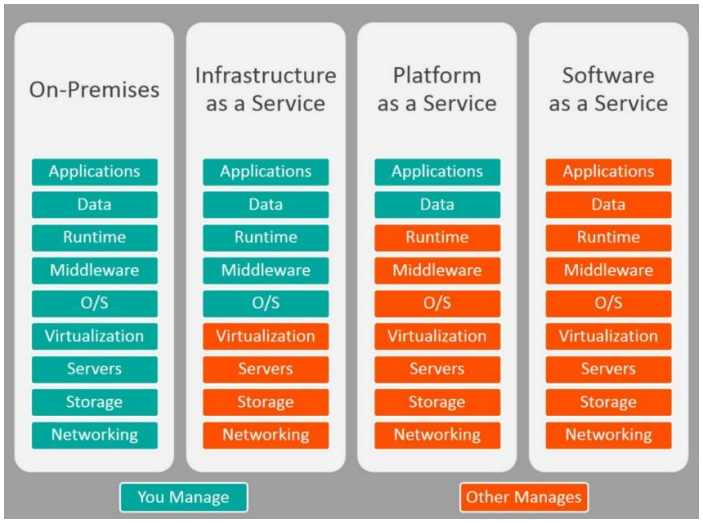


Figura 28. *Representació dels components que són administrats pel proveïdor (de color vermell) i els components administrats manualment (el consumidor, de color blau) d'acord al tipus de server utilitzat.* Extret de *BMC.* Font: https://blogs.bmc.com/wp-content/uploads/2017/09/iaas-paas-saas-comparison-1024x759.jpg

5. Requisits del sistema

Els requeriments exposats a continuació estan agrupats segons l'actor que executa les accions exposades pels requeriments, i estan classificats segons un tipus de prioritat, essent el grup de prioritat 1 el més urgent (prioritat més alta), el 2 de prioritat mitjana i el 3 de prioritat baixa.

5.1. Requisits funcionals

1.1. Sistema

1.1.1. Detectar rostres humans. *[PRIORITAT 1]*

1.1.2. Identificar els rostres dels residents de l'habitatge i distingir-los dels no reconeguts. *[PRIORITAT 1]*

1.1.3. Detectar que la porta d'accés a l'habitatge s'ha obert. *[PRIORITAT 1]*

1.1.4. Emetre senyals acústics en cas que la porta d'accés a l'habitatge s'obri i l'alarma estigui activada. *[PRIORITAT 1]*

1.1.5. Enviar una captura als residents en detectar el rostre d'un possible intrús (persona no resident), fotografiada a través de la càmera. *[PRIORITAT 2]*

1.1.6. Notificar, en temps real, als residents sobre esdeveniments que es produeixin en relació amb el sistema d'alarma i al sistema de reconeixement facial (canvis d'estat, intrusions, deteccions i reconeixements). *[PRIORITAT 2]*

1.2. Resident

1.2.1. Consultar i gestionar l'estat del sistema, de forma local i remota, a través d'un portal web. *[PRIORITAT 1]*

1.2.2. Visualitzar i filtrar els esdeveniments produïts. *[PRIORITAT 3]*

5.2. Requisits no funcionals

2.1. Fiabilitat

2.1.1. El sistema ha de detectar correcte i ràpidament els rostres dels residents, en condicions de baixa i alta lluminositat. *[PRIORITAT 1]*

2.1.2. El sistema de reconeixement i detecció facial no ha d'afectar el sistema d'alarma (el que detecta l'obertura de la porta d'accés a l'habitatge i que posteriorment emet el senyal acústic) quant a rendiment. *[PRIORITAT 1]*

2.1.3. En cas de reconnexió a Internet, l'alarma ha d'enviar totes les notificacions d'esdeveniments que s'hagin produït durant l'interval de desconnexió al resident (i que no hagin pogut enviar-se). *[PRIORITAT 2]*

2.2. Eficiència

2.2.1. El temps de resposta per detectar una intrusió i emetre el posterior senyal acústic ha de ser igual o menor a 100 ms. *[PRIORITAT 1]*

2.2.2. El temps de detecció de rostres ha de ser inferior a 2 segons. *[PRIORITAT 1]*

2.2.3. Els canvis d'estat del sistema realitzats a través del portal web s'han de propagar en un temps inferior o igual a 1 segon. *[PRIORITAT 3]*

2.2.4. Les notificacions enviades han de tenir un retard màxim d'arribada de 2 segons. *[PRIORITAT 3]*

2.3. Usabilitat

2.3.1. El sistema de reconeixement facial ha de reconèixer, com a mínim, 3 rostres de residents. *[PRIORITAT 2]*

2.3.2. Minimitzar el risc de desactivació i d'intrusió informàtica per manipular el sistema. *[PRIORITAT 2]*

5.3. Matriu de dependències

A continuació apareix la matriu que mostra les dependències entre requisits funcionals i no funcionals (vegeu figura 29).

1.1.1.	1.1.2.	1.1.3.	1.1.4.	1.1.5.	1.1.6.	1.2.1.	1.2.2.	2.1.1.
	1.1.1.		1.1.3.	1.1.1.	1.1.1.		1.2.1.	1.1.1.
				1.1.2.	1.1.2.			1.1.2.
					1.1.3.			
					1.1.5.			

2.1.2.	2.1.3.	2.2.1.	2.2.2.	2.2.3.	2.2.4.	2.3.1.	2.3.2.
1.1.1.	1.1.6.	1.1.3.	1.1.1.	1.2.1.	1.1.6.	1.1.1.	1.1.1.
1.1.2.		1.1.4				1.1.2.	1.1.2.
1.1.3.							1.1.3.
1.1.4.							1.1.4.
							1.1.5.
							1.1.6.
							1.2.1.
							1.2.2.

Figura 29. *Matriu de dependències entre requisits funcionals i no funcionals.*

6. Planificació

En aquest apartat s'explica la planificació que s'ha realitzat respecte al projecte.

6.1. Paquets de treball

A continuació s'exposa la descomposició feta del projecte (vegeu figura 30) per tal de determinar-ne les unitats de treball i els conjunts de tasques que s'han de realitzar per tal d'assolir els objectius i requeriments proposats en els apartats anteriors.

PT1. Anàlisi previ	PT2. Planificació i preparació	Components		Desenvolupament					PT10. Avaluació
Anàlisi d'entorn i situació del problema	Definició de requeriments funcionals i no funcionals	PT3. Adquisició dels components	PT4. Preparació i prova dels components	PT5. Sistema de reconeixement facial	PT6. Sistema d'alarma	PT7. Sistema d'esdeveniments	PT8. Sistema de control i gestió	PT9. Documentació	Prova de funcionament components desenvolupats
Anàlisi de context	Estudi de viabilitat	Estudi de mercat	Anàlisi de documentació	Recerca d'algorismes i biblioteques	Desenvolupament	Recerca plataformes de notificacions existents	Implementació d'interfícies	Producció de documentació	Revisió compliment requisits
Anàlisi de necessitats	Planificació del desenvolupament	Selecció dels components	Preparació ordinador	Selecció biblioteca	Prova i validació	Selecció plataforma	Desenvolupament portal web	Validació documentació	Definició futures millores
	Implantació de l'entorn de treball	Adquisició dels components	Connexió dels components	Desenvolupament sistema detecció rostres		Anàlisi de documentació	Integració portal amb sist. recon. facial, sist. alarma i sist. esdev.		
			Prova dels components	Desenvolupament sistema reconeixement facial		Desenvolupament sistema de notificació d'esdeveniments	Prova i validació		
				Entrenament sistema de reconeixement facial		Prova i validació sistema notificació			
				Prova i validació		Integració sistema notificació amb sist. recon. facial i sist. d'alarma			
						Prova i validació integració			

Figura 30. *Descomposició en paquets de treball d'aquest projecte, juntament amb les activitats que s'han d'executar per a cada paquet.*

PT1. Anàlisi previ

- **Temporalització:** 15 hores.
- **Tasques:**

  T1. Anàlisi de l'entorn i situació del problema.

  T2. Anàlisi del context (solucions disponibles al mercat).

  T3. Anàlisi de necessitats.

- **Resultat obtingut:**

  Definició del tema del projecte i l'acta de constitució del projecte (el full del treball).

PT2. Planificació i preparació

- **Temporalització:** 45 hores.
- **Tasques:**

  T1. Definició de requisits funcionals i no funcionals del projecte.

  T2. Estudi de viabilitat.

T3. Planificació del desenvolupament.

T4. Implantació de l'entorn de treball.

- **Resultat obtingut:**

  Abast, viabilitat, planificació i entorn de treball del projecte.

PT3. Adquisició dels components

- **Temporalització:** 10 hores.
- **Tasques:**

  T1. Estudi de mercat dels ordinadors monoplaca disponibles, que compleixin amb els requisits concretats.

  T2. Selecció d'un dels ordinadors estudiats que s'ajusti millor al projecte.

  T3. Adquisició de l'ordinador seleccionat.

  T4. Selecció d'una font d'alimentació compatible amb l'ordinador monoplaca adquirit.

  T5. Adquisició de la font d'alimentació seleccionada.

  T6. Estudi de mercat de càmeres que siguin compatibles amb l'ordinador adquirit i permetin obtenir vídeo en condicions de lluminositat baixa i lluminositat normal.

  T7. Selecció d'una càmera de qualitat i òptima per situacions de lluminositat normal i lluminositat baixa, que s'ajusti millor a l'objectiu principal: construir un sistema de reconeixement facial.

  T8. Adquisició de la càmera escollida.

  T9. Estudi de mercat d'altaveus de mida reduïda que siguin compatibles amb l'ordinador adquirit i disposin d'un nivell acústic adequat.

  T10. Selecció d'un dels altaveus estudiats.

  T11. Adquisició de l'altaveu seleccionat.

  T12. Estudi de mercat de sensors d'obertura de porta que siguin compatibles amb l'ordinador adquirit i s'ajusti als requisits del projecte.

  T13. Selecció un dels sensors d'obertura trobats.

  T14. Adquisició del sensor escollit.

- **Resultat obtingut:**

Els components físics (hardware) necessaris per començar a desenvolupar el sistema d'alarma i el sistema de reconeixement facial.

PT4. Preparació i prova dels components

- **Temporalització:** 10 hores.
- **Tasques:**

  T1. Anàlisi de la documentació existent de l'ordinador monoplaca adquirit.

  T2. Descàrrega del sistema operatiu recomanat pel fabricant de l'ordinador.

T3. Instal·lació del sistema operatiu a l'ordinador.

T4. Configuració de l'ordinador monoplaca.

T5. Instal·lació del programari necessari per al desenvolupament del nostre projecte.

T6. Prova i validació del funcionament de l'ordinador monoplaca.

T7. Anàlisi de la documentació del fabricant de la càmera.

T8. Connexió de la càmera adquirida a la placa.

T9. Desenvolupament d'un programa que permeti realitzar una prova de vídeo de la càmera.

T10. Prova i validació del funcionament de la càmera en condicions de lluminositat normal i lluminositat baixa, utilitzant el programari desenvolupat.

T11. Anàlisi de la documentació del fabricant de l'altaveu.

T12. Connexió de l'altaveu adquirit a la placa.

T13. Desenvolupament d'un programa que permeti realitzar una prova de so de l'altaveu.

T14. Prova i validació del funcionament de l'altaveu d'acord amb les nostres necessitats, utilitzant el programari desenvolupat.

T15. Anàlisi de la documentació del fabricant del sensor.

T16. Connexió del sensor d'obertura de porta a la placa.

T17. Desenvolupament d'un programa que permeti provar i validar el funcionament del sensor.

T18. Prova i validació del funcionament del sensor utilitzant el programa desenvolupat.

- **Resultat obtingut:**

  Prova i validació de funcionament de tots els components adquirits.

PT5. Sistema de reconeixement facial

- **Temporalització:** 55 hores.

- **Tasques:**

T1. Recerca dels algorismes i biblioteques existents en matèria de reconeixement facial.

T2. Selecció una biblioteca que s'ajusti a les necessitats del projecte.

T3. Desenvolupament d'un sistema de detecció de rostres.

T4. Desenvolupament d'un sistema de reconeixement facial.

T5. Entrenament del sistema de reconeixement facial amb els rostres que ha de reconèixer.

T6. Unificació de tots els sistemes desenvolupats en un únic sistema.

T7. Prova i validació del funcionament del sistema unificat, per determinar si: detecta rostres i reconeix correctament els rostres amb els quals ha sigut entrenat.

- **Resultat obtingut:**

Amb aquest paquet de treball s'haurà obtingut un sistema de reconeixement facial que, en situacions de lluminositat normal i lluminositat baixa, serà capaç de: reconèixer rostres humans i identificar si el rostre humà és un dels residents del domicili.

El rostre capturat quedarà registrat al registre d'esdeveniment, a més d'enviar-se en temps real als residents.

Aquest sistema estarà connectat: al sistema d'esdeveniments, per deixar constància de les deteccions de rostres i reconeixements dels rostres dels habitants del domicili; i al portal web de gestió de l'alarma.

PT6. Sistema d'alarma

- **Temporalització:** 45 hores.
- **Tasques:**

  T1. Desenvolupament un sistema d'alarma que, en cas d'obertura de la porta (senyal del sensor), emeti un senyal acústic d'alarma (altaveu).

  T2. Prova i validació del funcionament del sistema d'alarma desenvolupat, per determinar si en cas d'obertura de la porta emet el senyal acústica configurada.

- **Resultat obtingut:**

S'haurà obtingut i avaluat un sistema d'alarma que emeti un senyal acústic en cas de detectar l'obertura de la porta del domicili.

Aquest sistema d'alarma estarà connectat: al sistema d'esdeveniments, per deixar constància de les activacions, desactivacions i registrar quan s'ha detectat obertura de la porta del domicili; i al portal web, per permetre gestionar l'estat actual de l'alarma (activació i desactivació).

PT7. Sistema d'esdeveniments

- **Temporalització:** 55 hores.
- **Tasques:**

  T1. Recerca de plataformes existents de notificació d'esdeveniments.

  T2. Selecció d'una plataforma de notificació.

  T3. Anàlisi de la documentació de la plataforma seleccionada.

  T4. Desenvolupament un sistema de notificació d'esdeveniments que utilitzi la plataforma escollida.

  T5. Prova i validació del funcionament del sistema desenvolupat.

  T6. Integració del sistema de notificació amb el sistema de reconeixement facial i el sistema d'alarma.

  T7. Prova i validació del funcionament del sistema resultant de la integració.

- **Resultat obtingut:**

El resultat d'aquest paquet de treball serà un sistema capacitat per rebre i emmagatzemar esdeveniments que li arribin del sistema d'alarma i el sistema de reconeixement facial. A més a més, aquest sistema també permetrà consultar aquests esdeveniments al portal web (sistema de control i gestió), i notificarà d'aquests esdeveniments als residents.

PT8. Sistema de control i gestió (portal web)

- **Temporalització:** 50 hores.
- **Tasques:**

  T1. Estudi de tecnologies backend i frontend.

  T2. Selecció de les tecnologies estudiades.

  T3. Disseny d'interfícies del portal web.

  T4. Implementació de les interfícies dissenyades.

  T5. Desenvolupament del portal web.

  T6. Integració dels sistemes de reconeixement facial, sistema d'alarma i sistema d'esdeveniments amb el portal web.

  T7. Prova i validació del funcionament del portal web amb la integració realitzada.

- **Resultat obtingut:**

Amb aquest paquet de treball s'haurà obtingut un sistema de control i gestió (un portal web, en essència) que permetrà visualitzar deteccions i reconeixements dels rostres i l'històric de deteccions d'obertura de la porta del domicili, i gestionar l'estat del sistema de reconeixement facial i el sistema d'alarma.

PT9. Documentació

- **Temporalització:** 70 hores.
- **Tasques:**

  T1. Producció de la documentació del projecte, simultàniament a la fase de desenvolupament.

  T2. Validació de la documentació generada amb l'objectiu de verificar que s'ajusti al que s'ha desenvolupat.

- **Resultat obtingut:**

  Documentació contínua en el temps que es produirà sempre que es realitzin modificacions.

PT10. Avaluació

- **Temporalització:** 20 hores.
- **Tasques:**

T1. Prova completa de funcionament dels components desenvolupats.

T2. Revisió del compliment dels requisits funcionals i no funcionals.

T3. Definició de futures millores.

- **Resultat obtingut:**

Documentació respecte a l'avaluació del funcionament dels components desenvolupats, d'acord amb les necessitats establertes a l'inici del projecte i els requisits definits a l'anàlisi del projecte.

6.2. Matriu de traçabilitat

La figura a continuació mostra la relació entre requisits i paquets de treball (vegeu figura 31).

PT1. Anàlisi previ	PT2. Planificació i preparació	Components		Desenvolupament					PT10. Avaluació
		PT3. Adquisició dels components	PT4. Preparació i prova dels components	PT5. Sistema de reconeixement facial	PT6. Sistema d'alarma	PT7. Sistema d'esdeveniments	PT8. Sistema de control i gestió	PT9. Documentació	2.1.1.
				1.1.1.	1.1.3.	1.1.5.	1.2.1.		2.1.2.
				1.1.2.	1.1.4.	1.1.6.	1.2.2.		2.1.3.
				1.1.5	1.1.6.	1.2.2.	2.2.3.		2.2.1.
				1.1.6.	1.2.1.	2.1.3.	2.3.2.		2.2.2.
				1.2.1.	1.2.2.	2.2.4.			2.2.3.
				1.2.2.	2.1.2.	2.3.2.			2.2.4.
				2.1.1.	2.1.3.				2.3.1.
				2.1.2.	2.2.1.				2.3.2.
				2.1.3.	2.2.3.				
				2.2.2.	2.2.4.				
				2.2.3.	2.3.2.				
				2.2.4.					
				2.3.1.					
				2.3.2.					

Figura 31. *Relació entre requisits i paquets de treball.*

6.3. Cronograma

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**Anàlisi previ**	15 horas	mié 01/04/20	jue 02/04/20
2		Anàlisi previ	0 horas	jue 02/04/20	jue 02/04/20
3		**Planificació i preparació**	45 horas	jue 02/04/20	vie 10/04/20
4		Planificació i preparació	0 horas	vie 10/04/20	vie 10/04/20
5		**Adquisició dels components**	10 horas	vie 10/04/20	lun 13/04/20
6		Adquisició dels components	0 horas	lun 13/04/20	lun 13/04/20
7		**Preparació i prova dels components**	10 horas	lun 13/04/20	mar 14/04/20
8		Preparació i prova dels components	0 horas	mar 14/04/20	mar 14/04/20
9		**Sistema de reconeixement facial**	55 horas	mié 15/04/20	jue 23/04/20
10		Sistema de reconeixement facial	0 horas	jue 23/04/20	jue 23/04/20
11		**Sistema d'alarma**	45 horas	jue 23/04/20	vie 01/05/20
12		Sistema d'alarma	0 horas	vie 01/05/20	vie 01/05/20
13		**Sistema d'esdeveniments**	55 horas	vie 01/05/20	mar 12/05/20
14		Sistema d'esdeveniments	0 horas	mar 12/05/20	mar 12/05/20
15		**Sistema de control i gestió**	50 horas	mar 12/05/20	mié 20/05/20
16		Sistema de control i gestió	0 horas	mié 20/05/20	mié 20/05/20
17		**Documentació**	70 horas	mié 20/05/20	mar 02/06/20
18		Documentació	0 horas	mar 02/06/20	mar 02/06/20
19		**Avaluació**	20 horas	mar 02/06/20	jue 04/06/20
20		Avaluació	0 horas	jue 04/06/20	jue 04/06/20

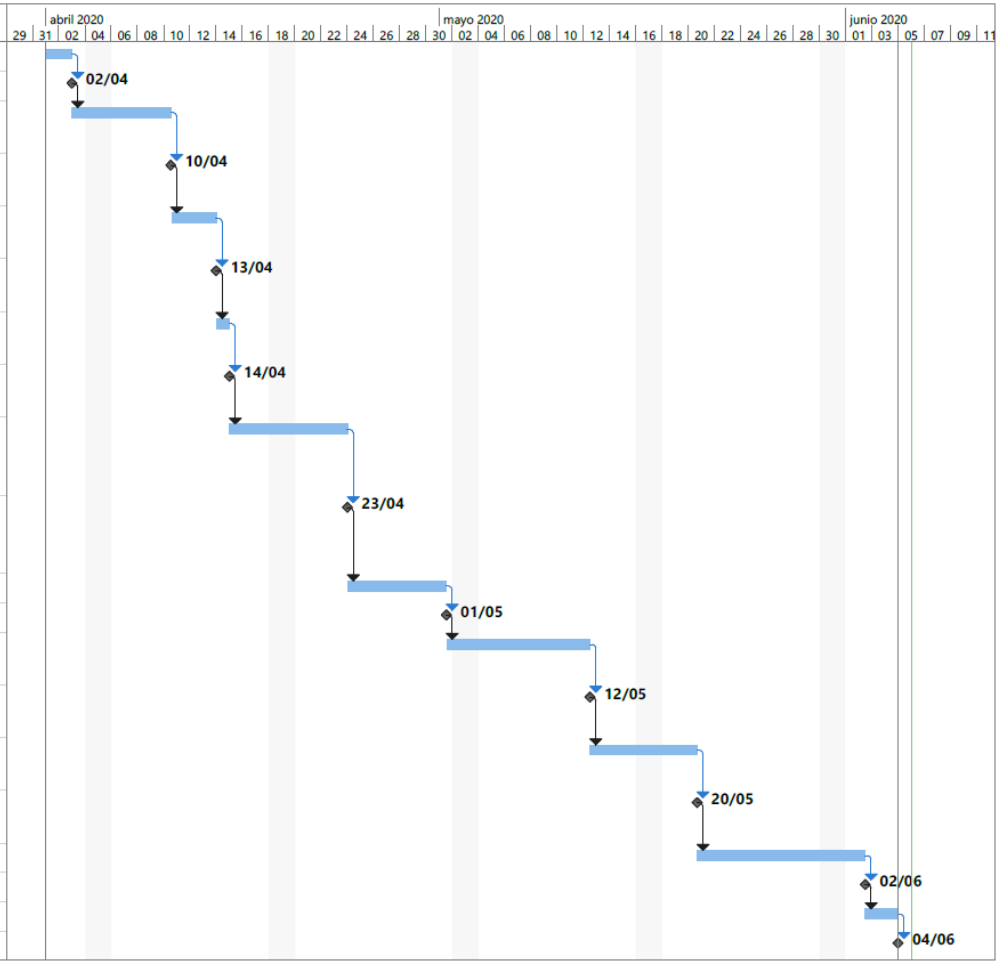


Figura 32. *Diagrama Gantt dels paquets de treball.*

7. Estudis i decisions

Els estudis realitzats i les decisions preses mitjançant aquests estudis estan explicats en els subapartats a continuació, en els que s'han agrupat les decisions en funció de quin component s'aplicaven.

7.1. Ordinador monoplaca

A l'hora d'escollir un fabricant l'autor es va decantar per *Raspberry Pi* pels següents motius: el cost i mida de la placa són reduïts, disposa d'una àmplia comunitat que proporciona suport (els mateixos fòrums oficials i articles fets per *makers*[4]), hi ha publicada una documentació comprensible i les plaques d'aquest fabricant són senzilles d'utilitzar.

D'entre els models de la *Raspberry Pi*, es va realitzar una comparativa[5] per esbrinar quina era la placa que oferia les prestacions justes i necessàries, segons els factors: freqüència de CPU, nombre de cores, memòria RAM, connectivitat Wi-Fi i presència de port GPIO. El model que es va acabar triant va ser la ***Raspberry Pi 3 Model A+*** (vegeu taula 3 per consultar les especificacions tècniques de la placa, i vegeu la figura 33 per visualitzar la part frontal de la placa). El cost d'aquesta placa va ser de 28 €.

Taula 2. Especificacions tècniques de la Raspberry Pi 3 Model A+. Extret de *Raspberry Pi Foundation*. https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/

Component	Característiques
Processador	Quad-core de 64 bits a 1,4GHz (*Broadcom BCM2837B0*)
Memòria RAM	512MB LPDDR2 SDRAM
Wi-Fi	Connectivitat 2,4GHz i 5GHz IEEE 802.11.b/g/n/ac
Ports de propòsit general (component externs)	Capçal GPIO de 40 pins

[4] El concepte *maker* és un moviment que es basa en la cultura de "fes-t'ho tu mateix".
[5] Consulteu l'annex 1 per veure una comparativa de prestacions entre models de *Raspberry Pi*.

Figura 33. *Fotografia frontal de la placa Raspberry Pi 3 Model A+.* Extret de *Raspberry Pi Foundation*. Font: https://www.raspberrypi.org/homepage-9df4b/static/d8c289d96032055ebaf1521ef2dc6577/052d8/f94faf0b49dad4c43c28509b80074f2d474be863_3a-3.jpg

Es va decidir que el software desenvolupat a la *Raspberry Pi* seria escrit en *Python*, un llenguatge interpretat d'alt nivell i de propòsit general, que compta amb un ampli suport per part del fabricant de la placa, a més d'haver-hi un ampli ventall de llibreries *Python* relacionades amb la *Raspberry Pi*. A més a més, un factor clau en la decisió d'aquest llenguatge de programació va ser la facilitat d'aprenentatge i la quantitat de línies requerides per desenvolupar un programa.

7.2. Càmera

Al principi, l'aproximació plantejada va ser la d'adquirir una càmera borescòpica, per evitar extreure l'espiell ja instal·lat a la porta del domicili, i que per tant la càmera es trobés dins del forat de l'espiell i hi capturés imatge estant situada allà.

El primer pas va ser mesurar el diàmetre de l'espiell i les dimensions de la porta (vegeu figura 34): el gruix i el diàmetre del forat de l'espiell.

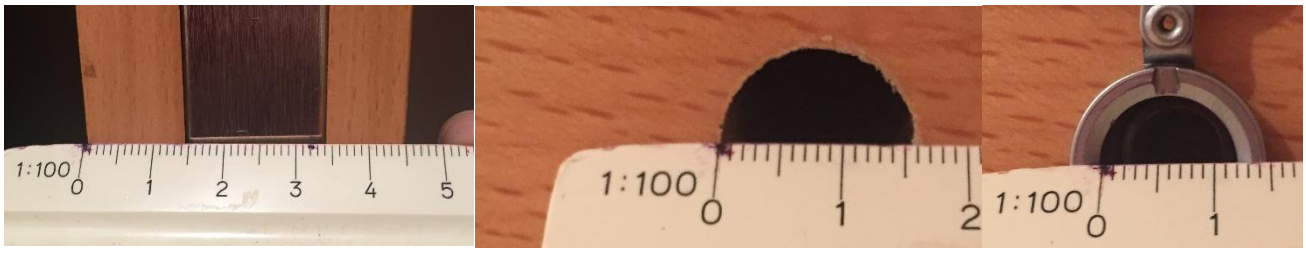


Figura 34. *Mesura del gruix de la porta (45 mm), mesura del diàmetre de l'orifici de l'espiell (16 mm) i mesura del diàmetre de l'espiell (12mm).*

Coneixent aquestes mesures, va procedir-se a buscar una càmera borescòpica a *Amazon* que no superés els 12 mm de diàmetre. La majoria de càmeres trobades disposaven d'un cable suficientment llarg

(més d'1 metre), per tant el gruix de la porta no era un impediment. Dos altres aspectes tinguts en compte van ser la qualitat de la càmera i la connectivitat (que anés connectat per USB, ja que la placa escollida disposava d'un port USB).

Es va trobar i adquirir la càmera borescòpica de la marca *Depstech* model *NTC86T-5M*, amb un cost de 26€. Aquest borescopi tenia un objectiu d'1 MP, podia gravar a 720p, utilitzava connexió per USB i el diàmetre de l'objectiu era de 5,5 mm.

No obstant, després de rebre el borescopi i d'instal·lar-lo dins l'espiell, la imatge que es capturava era la següent (vegeu figura 35):

Figura 35. *Imatge capturada pel borescopi, instal·lat a dins de l'espiell.*

Com pot apreciar-se a la figura 35, la qualitat d'imatge era pèssima i insuficient per detectar un rostre. A causa d'això, l'autor va decidir extreure l'espiell de la porta per fer-hi passar únicament la càmera borescòpica. La imatge que capturava després d'aplicar aquest canvi era la següent (vegeu figura 36):

Figura 36. *Imatge capturada del borescopi un cop es va extreure l'espiell.*

Tot i ser una notable millora respecte a l'aproximació anterior, la qualitat d'aquesta càmera endoscòpica seguia sent insuficient, ja que no hi ha bon equilibri de colors: un costat de la cara apareix de color molt clar i l'altre extrem apareix fosc i és difícil apreciar els trets facials.

Després d'aquesta aproximació es va arribar a la solució definitiva: adquirir una càmera que funcionés òptimament en condicions de lluminositat baixa i normal. L'autor va acabar-se decantant per una càmera oficial del mateix fabricant de la placa, la *Raspberry Pi NoIR Camera V2* (vegeu la taula 4 per consultar les especificacions tècniques i vegeu la figura 37 per visualitzar una fotografia del model de la càmera). Aquesta càmera oferia compatibilitat nativa amb la *Raspberry Pi*, i disposava de llibreries en *Python* fàcils d'utilitzar per capturar vídeo. La càmera escollida no tenia un filtre d'infraroig, i per tant la feia més sensible a la llum (per conseqüència, podia enregistrar de forma òptima en situacions de baixa lluminositat). El cost d'aquesta darrera càmera va ser de 27€.

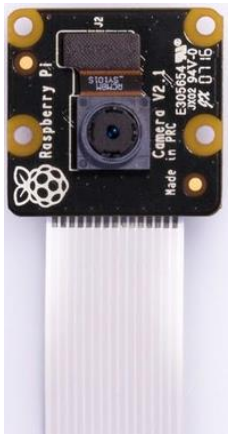


Figura 37. *Fotografia de la càmera Raspberry Pi No IR v2*. Extret de *Raspberry Pi Foundation*. Font: https://www.raspberrypi.org/products/pi-noir-camera-v2/

Malauradament, el format del cable que acompanya la càmera no és circular, sinó que és pla. Això és un inconvenient, ja que l'orifici de l'espiell de la porta és circular (vegeu figura 34 per visualitzar les mesures de la porta del domicili). Això va comportar que la *Raspberry Pi* s'hagués de trobar a la part exterior de la porta, per evitar danyar el cable de la càmera, que és pla, al passar-lo a través del forat de l'espiell.

Taula 3. Especificacions tècniques de la càmera *Raspberry Pi NoIR v2*. Extret de *TiendaTec*. Font: https://www.tiendatec.es/raspberry-pi/camaras/237-camara-para-raspberry-pi-noir-8mpx-1080p-v2-640522710898.html

Sensor d'imatge	Sony IMX 219 PQ CMOS
Resolució	8 megapíxels
Resolució d'imatge	3290 x 2464
Frames per segon (fps)	30 fps a 1080p

	60 fps a 720p
Angle de visió	62,2 º
Dimensions de la càmera	23,86 x 25 x 9 mm
Dimensions del cable	140 x 16 mm

Per altra banda, el fet que no tingués aquest filtre feia que les imatges que capturaven quedessin amb tonalitats de color rosades (vegeu figura 38). Això, però, no va presentar cap dificultat de cara al reconeixement facial, perquè com ja s'ha explicat, abans de tractar la imatge es transforma a blanc i negre.

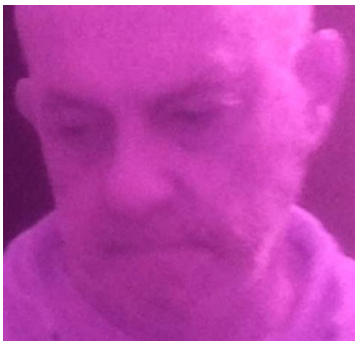


Figura 38. *Imatge capturada per la càmera NoIR.*

7.3. Interruptor magnètic

A causa de la situació excepcional de l'epidèmia, l'autor va cercar aquest producte a *Amazon*, que amb el servei *Prime* oferia enviaments ràpids en menys de 48 h. Dels productes que hi havia a la plataforma, n'hi havia molts que la data d'enviament era per passat el 15 de juny o més tard, cosa que va limitar molt el ventall d'opcions d'interruptors magnètics. Finalment, es va acabar escollint un interruptor de la marca *Fedit,* d'un cost de 15 € i que tenia bones valoracions.

7.4. Brunzidor

Va passar quelcom similar amb el brunzidor: es van acabar cercant brunzidors a *Amazon* i el ventall també era molt limitat a causa de l'epidèmia i a l'estat d'alarma. Finalment es va acabar adquirint un brunzidor, marca *FTVogue,* que tenia un cost de 6 €, era ben valorat pels clients i arribava en un temps raonable.

7.5. Reconeixement facial

En fer recerca sobre llibreries *Python* existents en matèria de reconeixement facial, l'autor va decantar-se per *FaceRecognition*, creada per *Adam Geitgey*. És una llibreria molt simple d'utilitzar, que internament utilitza un reconeixement facial d'última generació construït amb un t*oolkit* d'algorismes de

machine learning, anomenat *dlib* (Geitgey, The world's simplest facial recognition api for Python and the command line, 2020). El model de rostres utilitzat per *FaceRecognition* té una precisió d'un 99,38%, mesurada amb el *benchmark Labeled Faces in the Wild* (Huang, Ramesh, Berg, & Learned-Miller, 2018).

Gràcies a la facilitat d'ús d'aquesta llibreria (tenia un model preentrenat), es va escollir *FaceRecognition* com a la llibreria utilitzada per reconèixer rostres.

Per construir el conjunt de dades d'entrenament dels rostres dels residents a reconèixer va decidir-se enregistrar fragments de vídeo simulant l'arribada dels residents, portant ulleres i sense (els residents del domicili de l'autor utilitzen ulleres), i arribant des de les escales (des de la llunyania del passadís) i des de l'ascensor (sortint de l'ascensor, que es troba a l'esquerra de la porta del domicili).

Respecte a la captura de vídeo, va utilitzar-se la llibreria *Python* anomenada *PiCamera*, que proporcionava una interfície d'accés a la càmera intuïtiva i senzilla d'utilitzar.

7.6. Alarma

En dissenyar el sistema d'alarma, s'havia d'utilitzar una llibreria per poder rebre dades del sensor magnètic (d'obertura de porta) i per poder enviar dades al brunzidor, de forma que s'emetés un senyal acústic en funció de la lectura del sensor magnètic.

La llibreria que va utilitzar-se era *RPi.GPIO*, que oferia una interfície simple per interactuar amb els ports de propòsit general GPIO.

7.7. Paral·lelisme

Un dels objectius d'aquest projecte és assegurar la fiabilitat i minimitzar el temps de resposta quan es produeix una intrusió i quan es detecta un rostre no reconegut, de forma que paral·lelament s'actuï d'acord amb aquests esdeveniments. Per aquest motiu, a la fase de desenvolupament s'ha hagut d'implementar un mecanisme que permetés assolir aquest objectiu, de manera que s'aprofitessin els quatre nuclis que disposa el processador de la placa.

Desafortunadament, l'intèrpret del llenguatge de programació *Python* està limitat a l'ús simultani d'un únic *thread*, ja que utilitza un mecanisme anomenat *Python Global Interpreter Lock* (GIL). Com que *Python* disposa d'un control intern de referències a objectes que es declaren i s'utilitzen, quan arriba el punt que un objecte ja no és referit enlloc aquest objecte s'allibera de la memòria (similar al *Garbage Collector* de *Java*). La tasca principal d'aquest mecanisme, el GIL, és el d'impedir que diversos fils alterin les referències dels objectes que *Python* està comptabilitzant, evitant que un fil decrementi les referències d'un objecte i s'alliberi perquè aquest recompte arriba a 0, mentre que en un altre fil es torna a referir a l'objecte (Código Facilito).

A causa d'aquesta necessitat de poder executar tasques de forma paral·lela que s'ha de cobrir, s'ha utilitzat la llibreria estàndard *multiprocessing*, que permetrà «esquivar» el GIL de *Python* i crear subprocessos en comptes de fils (Python Software Corporation, 2020).

7.8. Infraestructura

Durant el desenvolupament del projecte, es va decidir alliberar la càrrega d'execució d'un servidor web de la placa. Aquest servidor web havia de permetre als residents accedir al portal de control i gestió, enviar notificacions i rebre missatges de Telegram. La solució que es va seguir va ser la d'externalitzar l'execució d'aquest servidor, i d'aquesta manera, la placa disposava de més capacitat per detectar i reconèixer rostres, i s'evitava que els temps de resposta del sistema d'alarma i del sistema de reconeixement facial es veiessin repercutits.

Per simplificar la creació, configuració i manteniment del servidor web i de la infraestructura sobre la qual s'executa aquest servidor es va escollir la plataforma *Heroku*, que ofereix serveis *Platform as a Service* (PaaS). A *Heroku* s'encarreguen d'administrar el *hardware* i els servidors mentre el desenvolupador se centra amb el desenvolupament i el desplegament d'aquesta aplicació, de forma molt senzilla (Rusev & Gadzhev, 2018).

Els contenidors virtuals on s'allotgen i s'executen les aplicacions a *Heroku* s'anomenen *dynos*. Els llenguatges de programació oficialment suportats per *Heroku* i executables en aquests contenidors són: *NodeJS, Ruby, Java, PHP, Python, Go, Scala i* Clojure. La plataforma possibilita la instal·lació de funcionalitats addicionals en aquests contenidors, mitjançant els *addons*. Exemples de funcionalitats instal·lables són: bases de dades, emmagatzematge al núvol, missatgeria, monitoratge, testing, caching, entre d'altres. Els tipus de *dynos* són els següents (Frías):

- Web: proporciona un servei d'aplicatiu web.
- Worker: executa la base de dades.
- Cron: s'executa per a processos de curt termini.

Quan s'utilitzen *dynos* de tipus «web», la plataforma *Heroku* s'encarrega de redirigir les peticions HTTP als *dynos* de l'aplicació, a través d'un balancejador de càrrega de la mateixa plataforma (Heroku, 2020). La comunicació de dades entre el client i l'enrutador d'*Heroku* està encriptada per defecte amb SSL.

Va ser necessari l'ús d'un sistema de gestió de bases de dades relacionals per emmagatzemar i persistir les dades generades amb la interacció entre el resident i els sistemes, com també les interaccions produïdes entre el servidor web i els sistemes. Malauradament, el fet d'haver escollit *Heroku* i utilitzar un compte gratuït va limitar el ventall de tecnologies de bases de dades disponibles, ja que *PostreSQL* era l'única que permetia utilitzar-se en comptes gratuïts sense necessitat d'haver d'afegir una targeta de crèdit.

7.9. Comunicacions

El fet d'haver decidit «externalitzar» l'allotjament del servidor web va comportar que s'hagués d'estudiar quin protocol s'utilitzaria per a les comunicacions entre l'alarma i el servidor extern.

Com que es volia limitar l'exposició de l'alarma a Internet, es va determinar que no s'haguessin d'obrir ports ni que es poguessin establir connexions directament amb la placa, sinó que la placa fos la que establís la connexió amb el servidor (en cas de comercialitzar-se, això facilitaria als clients la configuració i instal·lació de l'alarma). Per tant, això volia dir que només la placa podria sol·licitar establir connexions amb estacions que es trobessin connectades a Internet, però no rebre connexions establertes per estacions externes. Ja que un dels requisits d'aquest projecte és el de notificar en temps real sobre els esdeveniments que es produeixen, era convenient trobar un protocol que garantís connexions persistents bidireccionals (del servidor a l'alarma i viceversa).

L'autor es va acabar decantant per a l'ús dels *WebSocket*, ja que d'aquesta forma tant el servidor web com l'alarma o el reconeixement facial poden enviar dades sense la sol·licitud explícita de l'altra estació.

Per la creació d'un servidor *WebSocket* a l'aplicació allotjada a *Heroku*, es va fer servir la llibreria de *NodeJS* anomenada *ws*, de funcionalitat completa i facilitat per adaptar-la amb l'ús conjunt i paral·lel d'*ExpressJS*.

Per a l'ús i establiment de connexions *WebSocket* per part dels sistemes amb l'aplicació d'*Heroku,* va utilitzar-se la llibreria de *Python* de nom *websockets*.

El format de transport de missatges que va escollir-se va ser *JSON*, ja que aporta flexibilitat i és un format suportat de sèrie en molts llenguatges de programació.

Quant a l'enviament d'imatges, l'autor va decidir codificar-les en base64 per tal de poder incloure la imatge en un missatge que utilitzi el format JSON.

7.10. API

Per construir l'API es va decidir utilitzar NodeJS, un entorn d'execució que permet executar codi JavaScript fora d'un navegador web que utilitza el motor JavaScript V8 de Google. Aquest entorn està orientat al desenvolupament d'aplicacions d'Internet fàcilment escalables i asíncrones (Wikipedia, 2020). A continuació un fragment d'exemple d'un servidor web que retorna la cadena de text «Hello, World!» (Node.js, 2019):

```
1. const http = require('http');
2.
3. const hostname = '127.0.0.1';
```

```
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

El motiu per escollir aquest entorn és la facilitat d'implementació d'un servidor WebSocket, a més del ventall de llibreries orientades a crear servidors web disponibles.

Seguidament es va haver de triar una web framework, i l'autor va decantar-se per ExpressJS. Aquesta framework proporciona components com: encaminament, gestió de sessions i galetes, motors de plantilles i middlewares. El motiu principal darrere aquesta elecció va ser la facilitat que tenia ExpressJS per treballar conjuntament amb WebSocket, de forma que amb un únic servidor web podia servir una API i alhora un servidor de WebSocket. A continuació un exemple d'un servidor web que atén peticions al port 3000 i retorna el missatge «Hello World!» quan es visita la pàgina principal (ExpressJS, 2020):

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening at
http://localhost:${port}`))
```

7.11. Portal web

Per desenvolupar el backend del portal web, també s'ha emprat *NodeJS* i la *framework ExpressJS.* A causa de la facilitat d'integració amb l'esmentada *framework*, va utilitzar-se *Pug* com a motor de plantilles[6].

Com a llenguatges de programació, s'han escollit HTML5, per especificar l'estructura de les pàgines, CSS3, per especificar les regles de disseny dels elements que conformen les pàgines, JavaScript i jQuery, per les crides AJAX[7] en segon pla, i Bootstrap, una biblioteca que permet crear fàcilment interfícies d'usuari adaptables a qualsevol resolució i tipus de dispositiu, a més de disposar de plantilles de disseny i components prefabricats (Wikipedia, 2020).

7.12. Notificacions

[6] Un motor de plantilla és un processador que combina plantilles HTML amb dades per produir pàgines web.
[7] *AJAX* és una tecnologia que permet actualitzar continguts web sense recarregar la pàgina.

Pel que fa a plataforma d'entrega de notificacions, calia escollir-ne una que permetés l'enviament tant de missatges de text com d'imatges.

L'autor va escollir *Telegram* com a plataforma d'entrega de notificacions, perquè disposa de comptes *bot* (un tipus especial de compte que no requereix un número de telèfon per crear-la), i d'una API per implementar el comportament d'aquest bot.

Els motius per haver escollit *Telegram* són: no requereix invertir temps a dissenyar, implementar i publicar una aplicació específica perquè funcioni per a l'alarma, i l'usuari final pot evitar descarregar més aplicacions al dispositiu mòbil, ja que amb tenir instal·lat *Telegram* és suficient.

Com que la interacció amb els servidors de *Telegram* només es realitzava des de l'API, es va decidir l'ús de la llibreria *NodeJS* anomenada *Node.js Telegram Bot API*.

8. Anàlisi i disseny del sistema

A continuació s'exposa l'anàlisi detallat i els diagrames que s'han realitzat per desenvolupar el projecte.

8.1. Casos d'ús

L'únic actor identificar és el resident, que és l'únic que actuarà i realitzarà accions directament i explícitament sobre l'alarma.

El resident podrà realitzar els casos d'ús representats a continuació des del sistema de gestió i control (vegeu figura 39).

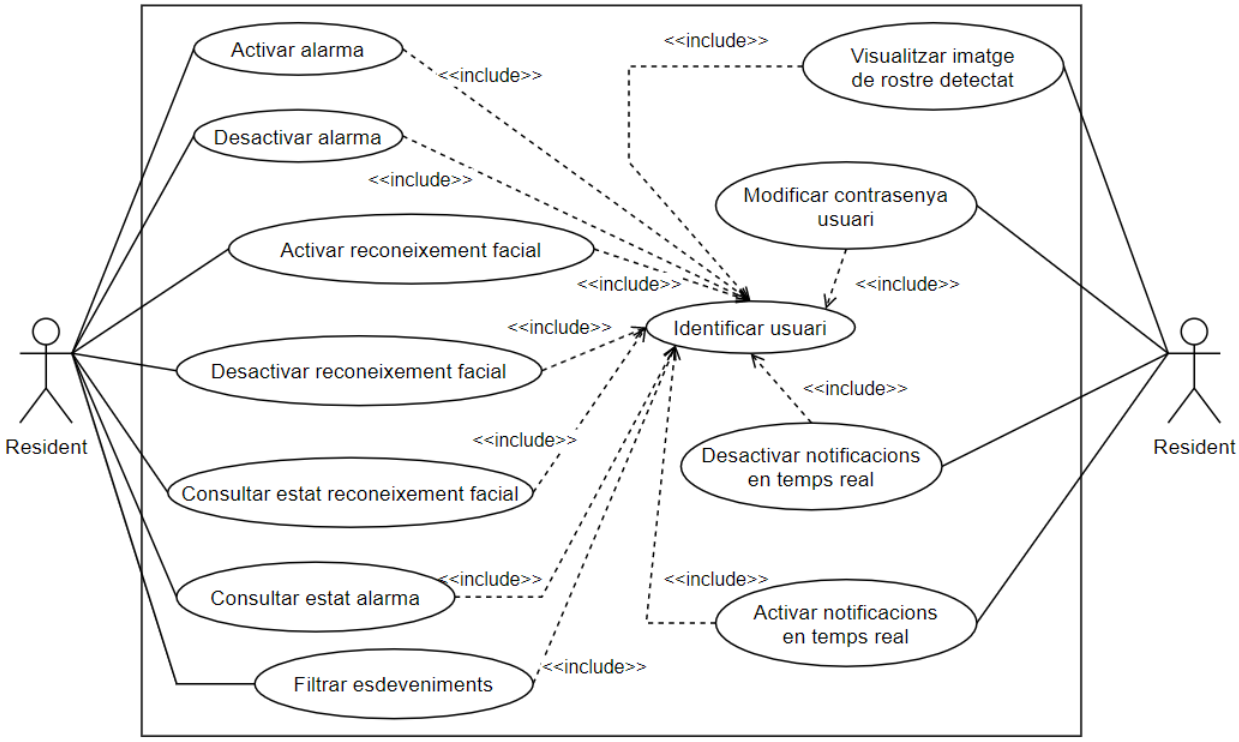


Figura 39. *Diagrama de casos d'ús d'aquest projecte.*

8.2. Model de processos

8.2.1. Alarma

El procés d'alarma consisteix en definir el comportament que tindrà el sistema d'alarma respecte a l'estat de la porta, és a dir, si la porta del domicili s'obre o es tanca (vegeu figura 40).

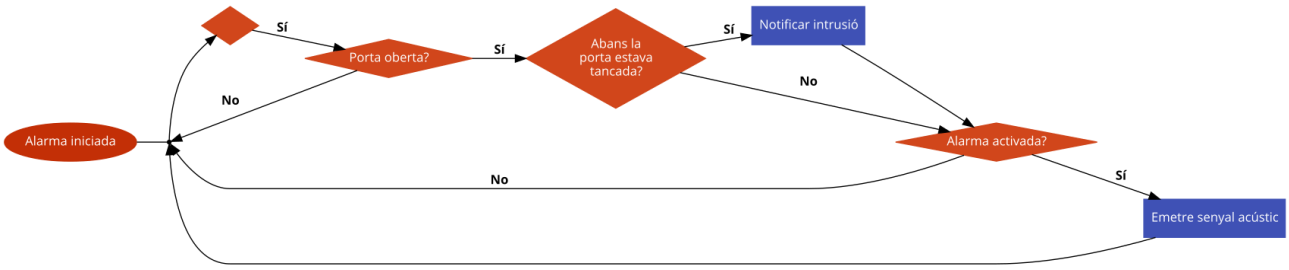


Figura 40. *Procés d'alarma en cas d'obertura de la porta del domicili.*

8.2.2. Reconeixement facial

El procés de reconeixement facial consisteix a definir el comportament que tindrà el sistema del reconeixement facial respecte a la presència de rostres capturats, per al posterior reconeixement (vegeu figura 41).

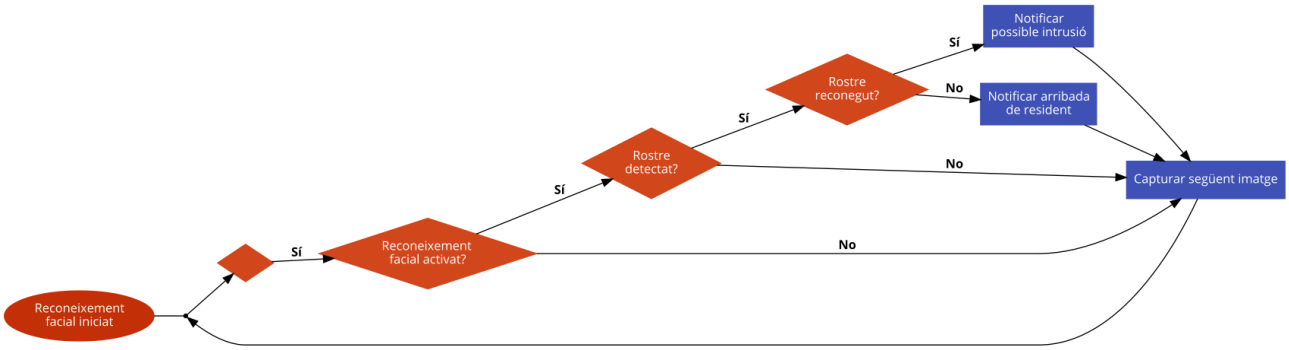


Figura 41. *Procés de reconeixement facial.*

8.3. Model de dades

Les taules que van determinar-se en fer l'anàlisi del model de dades són les següents (vegeu figura 42):

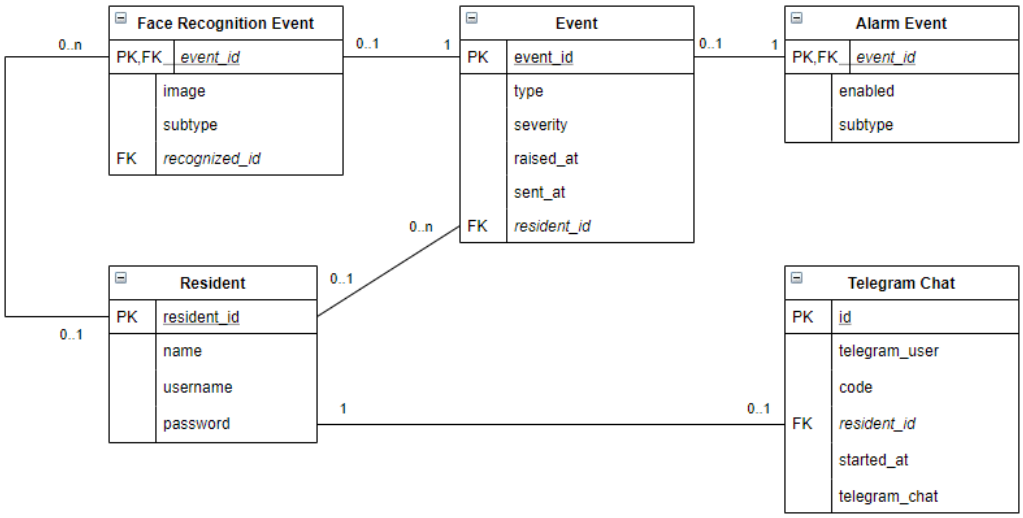


Figura 42. *Esquema del model de dades del projecte.*

8.4. Interfícies

La interfície de la pàgina inicial del sistema de gestió i control que va dissenyar-se per a la posterior implementació és la següent (vegeu figura 43):

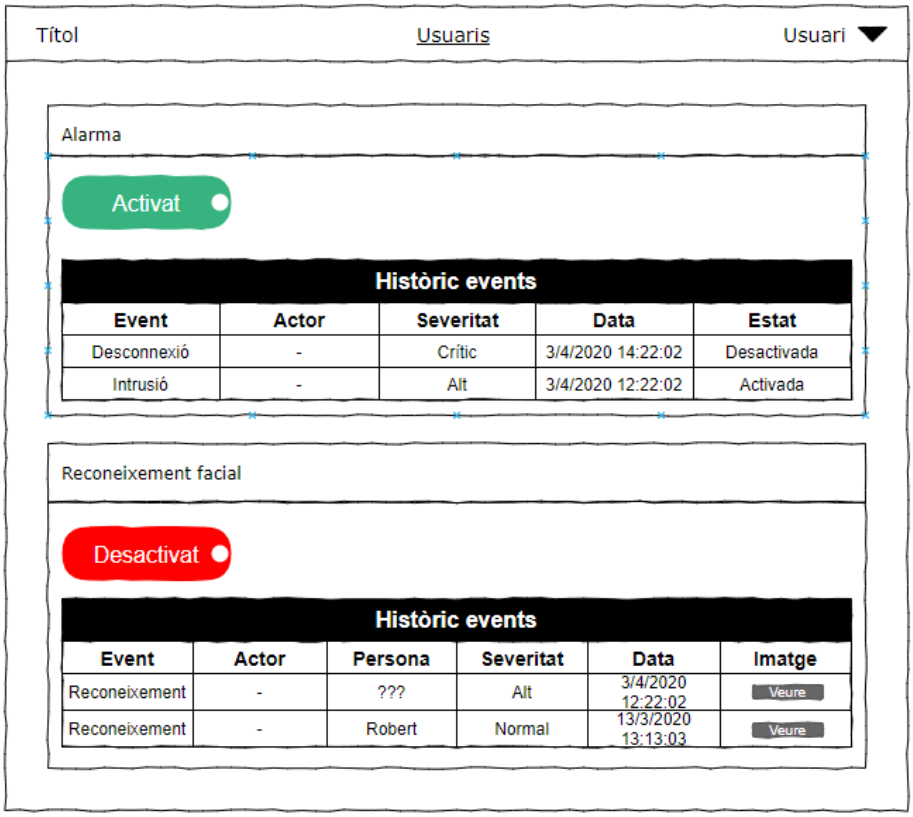


Figura 43. *Interfície de la pàgina inicial.*

Quant a la pàgina de gestió d'usuaris, la interfície que va dissenyar-se és la següent (vegeu figura 44):

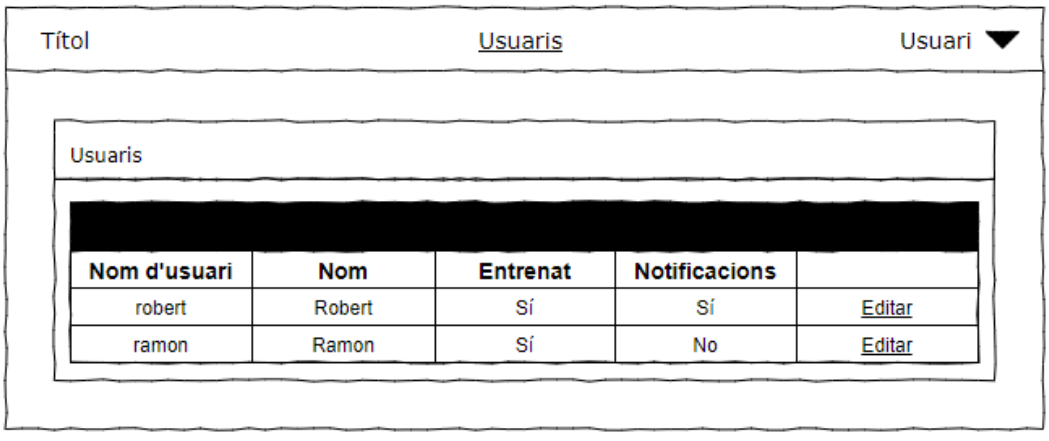


Figura 44. *Interfície de gestió d'usuaris de l'alarma.*

9. Implementació i proves

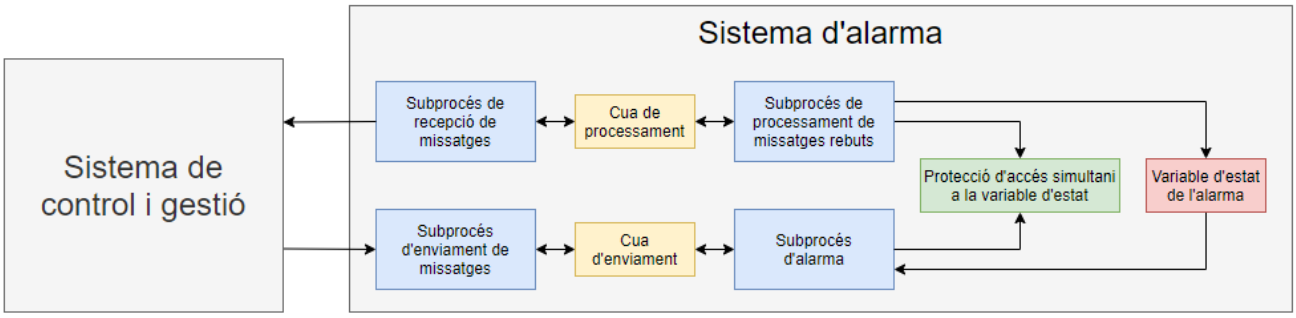
En aquest apartat es detallen els problemes i solucions aplicades durant la fase d'implementació del projecte.

9.1. Implementació

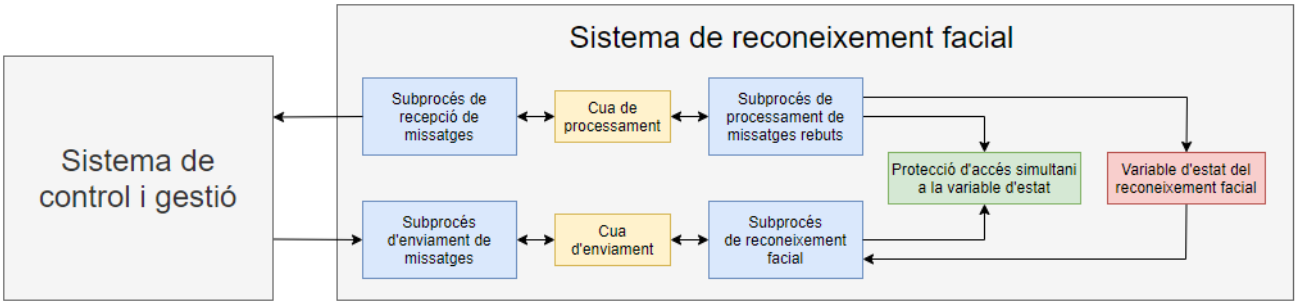
Per tal de facilitar la comprensió de les solucions aplicades amb els problemes apareguts, s'explicarà quina és l'estructuració del sistema d'alarma i el sistema de reconeixement facial a l'apartat d'estructuració.

9.1.1. Estructuració

- **Sistema d'alarma**



- **Sistema de reconeixement facial**



Com pot apreciar-se a les imatges anteriors, els dos sistemes segueixen el mateix patró i estructuració de divisió per subprocessos (components) i utilitzen variables similars (variable d'estat, mecanisme de protecció d'accés simultani i les cues d'enviament i processament), a més dels subprocessos que es comuniquen amb el sistema de control.

Va decidir-se la separació de la recepció dels missatges per part del sistema de control i el processament d'aquests en dos subprocessos diferents per garantir que no es perdessin missatges enviats pel sistema de control i gestió mentre es processaven els que ja havien sigut enviats.

Havent explicat l'estructuració dels sistemes, els problemes que han sorgit durant la implementació han sigut s'expliquen a l'apartat que ve a continuació.

9.1.2. Problemes i solucions

- **Control d'autenticació en connexions *websocket***

Per garantir que les connexions *websocket* establertes amb el sistema de control fossin legítimes (i que per tant procedissin dels sistemes desenvolupats i no d'orígens maliciosos), es va concretar la necessitat d'un sistema de control d'accés. Com que de per si aquest protocol de comunicació no disposa d'un mecanisme d'autenticació de fàbrica, l'autor va decidir-se per incloure la contrasenya d'autenticació a l'URL del servidor *websocket*, com es pot veure a continuació (vegeu taula 4):

Taula 4. Comparativa del format de la URL de connexió amb el servidor *websocket*.

Abans	Després
wss://ripoll-alarm-api.herokuapp.com	wss://ripoll-alarm-api.herokuapp.com/**CONTRASENYA**

Per tant, si quan s'establia una connexió amb el servidor *websocket* no existia aquest paràmetre a l'URL o bé la contrasenya especificada era incorrecta, el servidor no acceptava el canvi de protocol d'HTTP a *websocket* i tancava la connexió.

A més a més, com que el sistema d'alarma i el sistema de reconeixement facial establirien connexions separades de manera que cada sistema establís una connexió amb el servidor de control, va decidir-se concretar una contrasenya diferent per a cada sistema.

- **Variable compartida entre subprocessos**

D'acord amb els requisits establerts en les fases inicials d'anàlisi i disseny, s'ha de permetre l'activació i desactivació remota tant del sistema d'alarma com del sistema del reconeixement facial.

Com que ambdós sistemes han sigut implementats utilitzant paral·lelisme (l'ús de subprocessos per garantir l'execució paral·lela d'aquests), s'han desenvolupat subprocessos en aquests sistemes que garantissin, de forma paral·lela, que es poguessin rebre missatges per part del sistema de control i gestió i es poguessin enviar missatges a esmentat sistema. Això va comportar que es creés un subprocés per la recepció de missatges i un altre subprocés per l'enviament de missatges. El fet de compartir una variable entre dos o més subprocessos no és una pràctica recomanada, ja que poden haver-hi problemes de consistència i de corrupció del valor d'aquesta variable en cas que els subprocessos que accedeixen i modifiquen la variable produeixin canvis a la vegada.

Per aquest motiu, va decidir-se la creació d'una connexió *websocket* per a cada direcció (és a dir, una connexió per a l'enviament de missatges i una altra connexió per a la recepció). D'aquesta manera, se

suprimia la necessitat d'utilitzar una variable compartida (la connexió *websocket* amb el sistema de control) entre dos subprocessos diferents. Perquè el sistema de control tingués constància de quina era la direcció de la connexió que s'establia, va tornar-se a modificar l'URL de connexió amb el servidor *websocket*. D'aquesta forma, s'annexava la direcció al final de l'URL: «tx» per indicar que el sistema que establia la connexió utilitzaria aquesta connexió per transmetre informació i «rx» per indicar el contrari. El resultat de l'aplicació d'aquesta solució pot observar-se a continuació (vegeu taula 5):

Taula 5. Comparativa del nou format de la URL de connexió amb el servidor *websocket*, en la que ara s'indica la direcció dels missatges.

Abans	Després
wss://ripoll-alarm-api.herokuapp.com/CONTRASENYA	wss://ripoll-alarm-api.herokuapp.com/CONTRASENYA/**tx|rx**

Per altra banda, d'entrada va suposar un inconvenient el fet de permetre l'activació i desactivació remota dels sistemes d'alarma i reconeixement facial, ja que si no s'hagués aplicat el paradigma de programació paral·lela no s'haurien pogut atendre les peticions remotes de canvi d'estat i els esdeveniments produïts (ja sigui una intrusió en el cas del sistema d'alarma, o bé la captura d'imatges en el cas del sistema del reconeixement facial). És a dir, si ambdós sistemes haguessin estat programats per treballar amb un únic procés, el fet d'estar atenent peticions remotes hauria comportat la pèrdua de la «captació» d'esdeveniments relatius a la seguretat del domicili, com per exemple perdre un *frame* on apareixia un rostre o bé perdre la lectura del senyal del sensor magnètic que indicava que la porta s'havia obert.

Una solució que es va plantejar per pal·liar aquest problema va ser la d'utilitzar una variable d'estat del sistema (activat/desactivat) compartida entre el subprocés del mateix sistema (sigui l'alarma o el del reconeixement facial) i el subprocés del processament de missatges rebuts en la connexió establerta amb el sistema de control, conjuntament amb l'ús d'un objecte anomenat *lock* (*multiprocessing.Lock*) que garantís protecció contra l'accés simultani a aquesta variable d'estat compartida. També va suposar-se el cas que el sistema d'alarma detectés una intrusió i hagués de consultar la variable de l'estat, mentre aquesta variable estava protegida perquè el subprocés de processament de missatges estava modificant-ne el valor. La solució que es va implementar per aquest problema des del procés d'alarma va ser que, si no era possible obtenir l'accés a la variable (perquè l'altre subprocés l'estava modificant), se suposava que el valor de la variable d'estat era activat. El mateix va aplicar-se pel sistema de reconeixement facial.

- **Comunicació entre subprocessos**

Com ja s'ha explicat en el punt anterior, el fet de compartir variables entre subprocessos no és una bona pràctica en el paradigma de programació paral·lela. Tanmateix, era necessari que el subprocés d'alarma generés l'esdeveniment d'intrusió en obrir-se la porta del domicili i que el transmetés al sistema de control, però el subprocés d'alarma no era el que disposava de la via de comunicació amb el sistema de control, sinó que era el subprocés d'enviament de missatges. Per aquest motiu, la solució que es va plantejar i utilitzar va ser la d'usar una cua de missatges a enviar, que era llegida i processada pel subprocés d'enviament de missatges, que posteriorment enviava els missatges de la cua. Perquè els altres subprocessos poguessin indicar-li al subprocés d'enviament els missatges que havia d'enviar, va crear-se una variable compartida de tipus *Queue* (*multiprocessing.Queue*), que està implementada amb un mecanisme intern de protecció d'ús compartit. Aquesta variable es passava per paràmetre als demés subprocessos, fet que permetia afegir missatges per enviar al sistema de control. Aquesta solució també va ser aplicada al sistema de reconeixement facial.

Aquesta solució també es va aplicar amb la comunicació entre el subprocés de recepció dels missatges i el subprocés de processament de missatges rebuts.

- **Pèrdua de connectivitat amb el sistema de control**

D'acord amb el requisit no funcional 2.1.3. els sistemes de l'alarma han de ser capaços de transmetre al sistema de control i gestió els esdeveniments produïts durant períodes de pèrdua de connectivitat amb l'esmentat sistema.

Per garantir el compliment d'aquest requisit, l'autor va decidir que seria convenient implementar un tractament d'excepcions en les accions relatives als *websocket* i adaptar el mètode d'establiment de connexió amb el sistema de control. A més a més, també va utilitzar-se una cua prioritària de missatges pendents per enviar però que no ha sigut possible la transmissió d'aquests degut a la pèrdua de connectivitat (anomenada cua d'enviament interna), que tenia més prioritat que la cua d'enviament de missatges compartida. Aplicant aquestes modificacions, el flux del subprocés d'enviament de missatges va passar a ser el següent (vegeu figura 45):

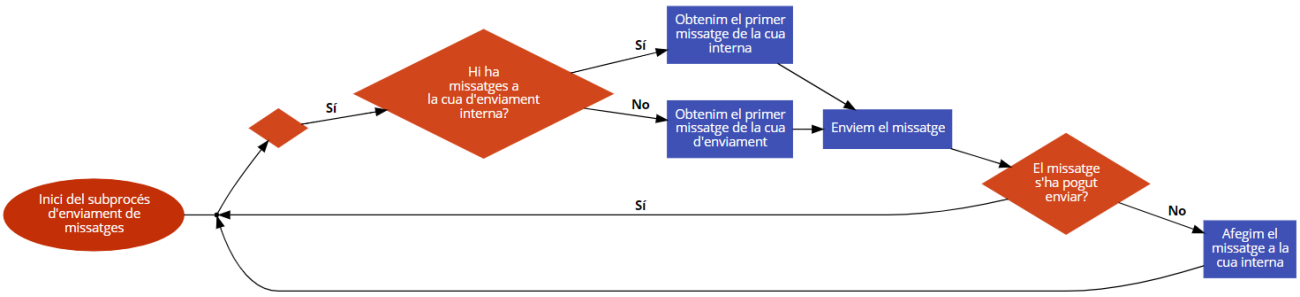


Figura 45. *Nou flux del subprocés d'enviament de missatges.*

El motiu darrere la decisió d'utilitzar una de cua de prioritat per a la cua d'enviament interna va ser que, utilitzant aquest tipus de cua, es prioritzaven els missatges d'esdeveniments que s'havien produït primer, i per tant s'enviaven primer que els esdeveniments produïts posteriorment.

- **Ús de la llibreria asíncrona *Python* anomenada *websockets***

Com que la llibreria *Python* utilitzada per treballar amb *websocket* utilitza el paradigma de programació asíncrona, en els subprocessos de recepció i enviament de missatges (és a dir, els que establien connexions de tipus *websocket*), va haver-se d'utilitzar la llibreria *asyncio* oficial de *Python.*

asyncio és una llibreria de programació asíncrona que permet l'ús de codi concurrent utilitzant els mecanismes i la sintaxi *async/await* (Python, 2020).

A continuació es mostra un fragment de codi utilitzat en el subprocés d'enviament de missatges:

```
1.  async def websocket_sender(send_queue):
2.      print("ALARM [WebSocket_Sender]: Iniciant websocket de sortida...")
3.
4.      websocket = await connect(WEBSOCKET_URL)
5.      pending_queue = PriorityQueue()
6.
7.      while True:
8.          ...
9.
10. def start_websocket_sender(send_queue):
11.     try:
12.         asyncio.get_event_loop().run_until_complete(websocket_sender(send_queue))
13.
14.     except KeyboardInterrupt:
15.         print("ALARM [WebSocket_Sender]: Tancant websocket de sortida...")
```

- **Hibernació aplicació *Heroku***

Un inconvenient que tenen els comptes gratuïts d'*Heroku* és que les aplicacions que s'executen es posen en un mode d'hibernació, i es reactiven quan reben peticions externes (Heroku, 2019). Aquest sistema de per sí no suposa un inconvenient per servidors webs o APIs, però sí suposa obstacles a l'hora de desenvolupar un servidor que treballa amb *WebSockets*, ja que són connexions persistents (Heroku, 2019).

Per aquest motiu, es va haver de crear una funció perquè enviés missatges de comprovació de l'estat de les connexions dels websockets:

```
1. // Funció que s'executa cada 30 segons, per enviar un paquet `ping` a cada
   connexió websocket oberta
2. setInterval(() => {
3.   wss.clients.forEach((client) => {
4.           client.send(client.ping());
5.   });
6. }, 30000);
```

- **Baix rendiment del sistema de reconeixement facial**

Per evitar perdre *frames* capturats per la càmera a causa del processament de les imatges en la cerca de rostres humans, es va reduir la resolució de la captura de la càmera de 1280x720 píxels a 480x368 píxels. Aplicant aquesta disminució de qualitat, es va aconseguir millorar el rendiment, de forma que el temps de processament de les imatges no era tan elevat.

Per tal de mesurar l'increment de rendiment al reduir la resolució es va calcular la diferència en nano segons entre les marques de temps UNIX abans i després d'aplicar l'algoritme de reconeixement facial, i va fer-se quatre cops per a cada resolució (vegeu taula 6).

Taula 6. Comparativa de temps d'execució de l'algorisme de reconeixement facial en diferents resolucions.

1280x720		480x368	
9,700883953 x 10^{-9} segons		2317073062 x 10^{-9} segons	
9807095140 x 10^{-9} segons		2359445913 x 10^{-9} segons	
9617274513 x 10^{-9} segons		2257963609 x 10^{-9} segons	
9700073751 x 10^{-9} segons		2293758230 x 10^{-9} segons	
Mitjana:	9706331839,25 x 10^{-9} segons	**Mitjana:**	2307060203,5 x 10^{-9} segons
Millora de rendiment: 23,77%			

- **Fus horari incorrecte en mostrar les marques de temps del sistema de gestió i control**

Les aplicacions que s'executen a la plataforma *Heroku* per defecte tenen configurat el fus horari UTC (temps universal coordinat), i per tant a l'hora de mostrar les marques de temps al portal web, aquestes es mostraven d'acord amb el fus horari mencionat (Wikipedia, 2020).

Per tal de configurar el fus horari per defecte del sistema de control i gestió (és a dir, la pàgina web amb la qual es controla l'alarma), es va haver de definir la variable d'entorn «TZ» amb el valor "Europe/Madrid" (Heroku).

- **Consulta de l'estat dels sistemes**

Quan un resident sol·licita l'estat d'algun dels sistemes (alarma o reconeixement facial) a través de *Telegram* o bé consultant-ho mitjançant el portal web, era necessari enviar un missatge al sistema del qual se'n volia conèixer l'estat. Per evitar aquest tràfic de missatges innecessari, es van crear dues variables al sistema de gestió i control que emmagatzemaven l'estat d'ambdós components. En cas que algun dels dos components no estigués operatiu (no tingués connectivitat), el sistema de gestió en tindria coneixement, ja que com s'ha explicat anteriorment s'envien missatges de comprovació d'estat cada 30 segons.

- **Duració del període de pèrdua de connectivitat de l'alarma**

Per evitar perdre la marca de temps de pèrdua de connexió de l'alarma amb el sistema de gestió i control, es va decidir crear una taula en la que emmagatzemar de forma persistent aquest valor. D'aquesta manera, si l'aplicació executada a *Heroku* era reiniciada o la seva execució era afectada per factors externs, la variable sempre estaria conservada a la base de dades, en comptes de guardar-se a memòria volàtil.

9.1.3. Algorismes rellevants

L'algorisme més rellevant del projecte ha sigut el que s'ha utilitzat i implementat per desenvolupar el sistema de reconeixement facial.

Per desenvolupar aquest sistema, s'han seguit els passos explicats a l'apartat 4.7., com també s'ha utilitzat la llibreria *FaceRecognition* que disposa d'un model preentrenat.

A continuació es mostra un fragment de codi de l'algorisme de reconeixement facial:

```
1.  def facerec(send_queue, is_active, is_active_lock):
2.      # Carreguem a memòria el model de classificació KNN
3.      with open('model.clf', 'rb') as f:
4.          knn_clf = pickle.load(f)
5.
6.      # Inicialitzem la captura i especifiquem la resolució i frames per segon
    destijats
7.      camera = PiCamera()
8.      camera.resolution = (WIDTH, HEIGHT)
9.      camera.framerate = FPS
10.     capture = PiRGBArray(camera, size=(WIDTH, HEIGHT))
11.
12.     sleep(0.1)
13.
14.     # Processarem només 1 de cada 2 frames capturats
15.     process_frame = True
16.
17.     # Iterem els frames capturats per la càmera
18.     for frame in camera.capture_continuous(capture, format='rgb',
    use_video_port=True):
19.         image = frame.array
20.
21.         # Processarem 1 de cada 2 frames
22.         if process_frame:
23.             # Cerquem ubicacions de rostres al frame
```

```
            face_locations = face_recognition.face_locations(image)

            # No s'han trobat rostres
            if len(face_locations) == 0:
                capture.truncate(0)
                continue

            # Cerquem les mesures dels rostres trobats
            faces_encodings = face_recognition.face_encodings(image,
known_face_locations=face_locations)

            # Busquem el veí més proper del model de classificació KNN prèviament
entrenat
            closest_distances = knn_clf.kneighbors(faces_encodings, n_neighbors=1)

            matches = []

            # Iterem cada rostre detectat
            for i in range(len(face_locations)):
                # Guardem si la distància amb el veí trobat és inferior o igual al
llindar (proximitat amb el veí)
                matches.append(closest_distances[0][i][0] <= DISTANCE_THRESHOLD)

            predictions = []

            # Iterem les prediccions realitzades
            for (person, recognized) in zip(knn_clf.predict(faces_encodings),
matches):
                # És un rostre reconegut
                if recognized:
                    predictions.append(person)
                # Rostre desconegut
                else:
                    predictions.append('desconegut')

        process_frame = not process_frame
        capture.truncate(0)
```

9.2. Proves

Les proves realitzades només han sigut proves de sistema, ja que es prova el funcionament i la integració de tots els components en conjunt, i que demostraran que s'han complert els requisits definits o bé no s'han pogut satisfer.

- **Amb l'alarma activada, separar les dues peces magnètiques del sensor de la porta.**
  - Resultat: senyal acústic de l'alarma, missatge via *Telegram* d'una intrusió, i notificació present al portal web.
  - Estat de la prova: superada.
- **Amb l'alarma desactivada, separar les dues peces magnètiques del sensor de la porta.**
  - Resultat: missatge via *Telegram* d'una intrusió, notificació present al portal web i sense senyal acústic d'alarma.

    - Estat de la prova: superada.
- **Tancar els processos del sistema d'alarma i del sistema de reconeixement facial.**
    - Resultat: missatge via *Telegram* informant de la pèrdua de connectivitat de l'alarma i del reconeixement facial.
    - Estat de la prova: superada.
- **Consultar l'estat de l'alarma.**
    - Resultat: missatge via *Telegram* amb l'estat actual de l'alarma.
    - Estat de la prova: superada.
- **Activar reconeixement facial i situar a la càmera un rostre no reconegut.**
    - Resultat: imatge del rostre enviada via *Telegram* i esdeveniment present al lloc web.
    - Estat de la prova: superada.
- **Desactivar reconeixement facial i situar a la càmera un rostre no reconegut.**
    - Resultat: cap.
    - Estat de la prova: superada.
- **Activar reconeixement facial i situar a la càmera el rostre d'un resident.**
    - Resultat: imatge del rostre reconegut enviada via *Telegram* i esdeveniment present al lloc web.
    - Estat de la prova: superada.
- **Filtrar esdeveniments des del portal web.**
    - Resultat: esdeveniments mostrats d'acord amb els criteris de filtratge fixats.
    - Estat de la prova: superada.
- **Desactivar l'alarma mentre està sonant.**
    - Resultat: silenci.
    - Estat de la prova: superada.

10. Implantació i resultats

10.1. Procés de desenvolupament

10.1.1. Ordinador monoplaca

El primer component comprat va ser l'ordinador monoplaca, el model *Raspberry Pi Model 3 A+*. Era necessària una targeta de memòria microSD, i l'autor va utilitzar una targeta que ja tenia de 8GB de capacitat. També va ser necessària una font d'alimentació amb un amperatge de 2,5 A (Raspberry Pi, 2020), que va haver-se d'adquirir.

Per poder engegar la placa feia falta que a la targeta de memòria s'hi trobés el sistema operatiu oficial, *Raspberry Pi OS* (cal remarcar que qualsevol altra distribució de Linux funciona). Per facilitar aquesta tasca, el fabricant disposa d'una eina oficial gratuïta anomenada *NOOBS* (*New Out Of the Box Software*) que descarrega i copia el sistema operatiu a la microSD de forma automàtica. Va ser necessari descarregar aquesta eina i copiar-la a la targeta de memòria.

Després d'inserir la targeta de memòria a la ranura corresponent, va ser necessari instal·lar el sistema operatiu (utilitzant l'eina NOOBS, vegeu figura 46), per després configurar el sistema operatiu: crear un compte d'usuari i connectar la placa a la xarxa Wi-Fi domèstica.

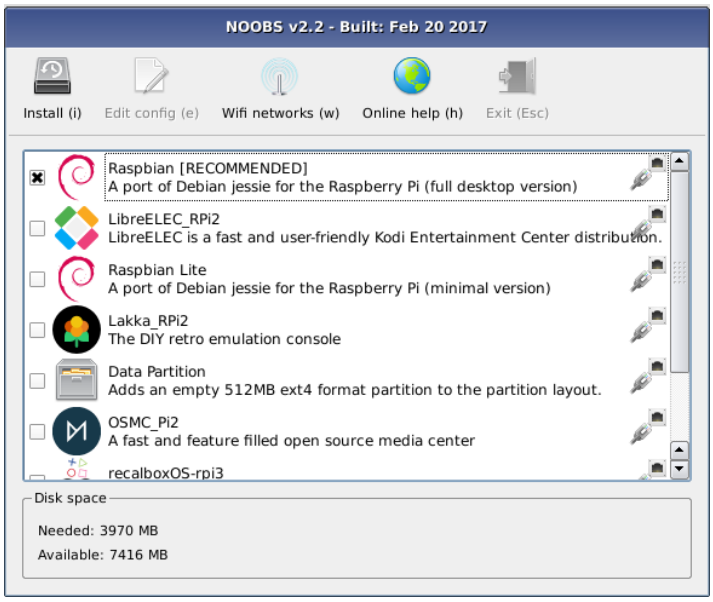


Figura 46. *Captura de pantalla de l'eina NOOBS.* Extret de *Raspberry Pi Foundation*. Font: https://www.raspberrypi.org/documentation/installation/images/noobs.png

10.1.2. Sistema de reconeixement facial

Com ja s'ha mencionat anteriorment, el següent component comprat va ser una càmera borescòpica, marca *Depstech* i model *NTC86T-5M*. Va connectar-se per USB i es va dur a terme una prova de

qualitat d'imatge, però després de veure que no era adequada pel reconeixement facial va adquirir-se la *Raspberry Pi NoIR Camera V2* (vegeu figura 47 per visualitzar el port de connexió utilitzat i per visualitzar la càmera connectada al port).

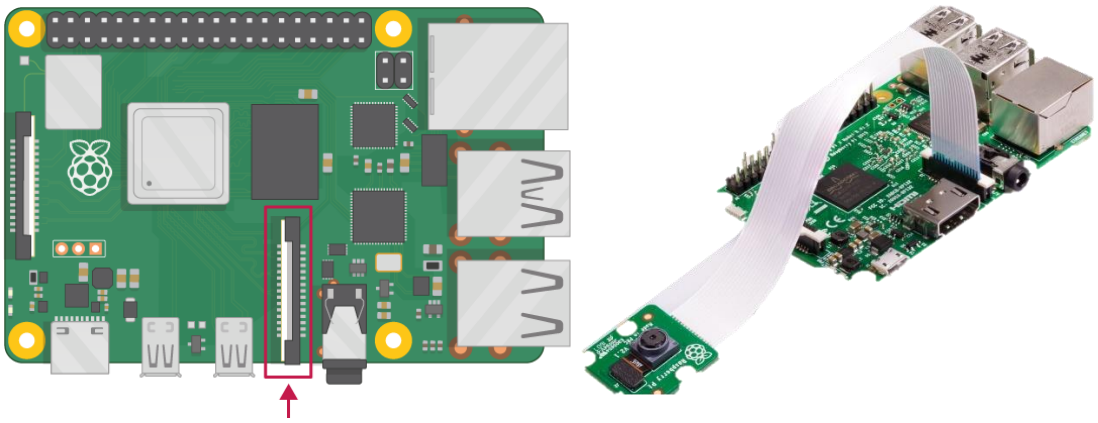


Figura 47. *A l'esquerra, ubicació del port de connexió de la càmera; a la dreta, la càmera connectada a l'esmentat port.* Extret de *Raspberry Pi Foundation*. Font: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

Un cop provat el correcte funcionament de la darrera càmera comprada, va procedir-se a desenvolupar el sistema de reconeixement facial.

Per instal·lar la llibreria *FaceRecognition* es va haver de descarregar i instal·lar la llibreria *dlib*, com també la llibreria *openCV* (Geitgey, Install dlib and face_recognition on a Raspberry Pi, 2017). Va decidir-se compilar manualment la llibreria *openCV*, ja que d'aquesta manera es podria instal·lar una versió optimitzada (Rosebrock, Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster, 2019). També va fer falta instal·lar la llibreria *scikit-learn* per poder entrenar el classificador KNN.

Seguidament, ja es disposaven de totes les llibreries *Python* de l'àmbit de reconeixement facial instal·lades. La següent tasca va ser la de crear un conjunt de dades d'entrenament, que com ja s'ha explicat va fer-se enregistrant fragments de vídeo dels residents simulant l'arribada. Per fer això, va haver-se de desenvolupar: un programa que rebés un nom de fitxer i guardés la captura de vídeo en aquest fitxer de sortida especificat, un altre programa que procés la captura de vídeo i en guardés els *frames* i finalment un altre programa que entrenés un classificador KNN i guardés aquest classificador en un fitxer, per a la seva posterior càrrega.

Quan ja van tenir-se desenvolupats, van utilitzar-se per enregistrar fragments de vídeo en els quals els residents a reconèixer simulaven l'arribada al domicili, amb ulleres i sense i des de diferents llocs, arribant des de l'ascensor i arribant des del passadís comunitari. En aquest moment va detectar-se la

limitació de la posició de la càmera, la qual influïa a l'hora de detectar rostres de persones d'altures diferents. En disposar dels fragments, va executar-se el programa que entrenava el classificador KNN.

Figura 48. *Imatge d'entrenament utilitzada per reconèixer a un dels residents.*

El següent pas va ser desenvolupar el sistema de reconeixement facial, que carregués a memòria el fitxer del model de classificació KNN i iterés els *frames* capturats per la càmera per analitzar si al *frame* hi havia un rostre i si era un rostre reconegut.

Un cop es va tenir el sistema de reconeixement facial va validar-se el seu funcionament a partir del model prèviament guardat, i de nou simulant l'arribada dels residents per tal de comprovar si els rostres eren degudament reconeguts, a més de fer proves amb rostres no reconeguts.

10.1.3. Sistema d'alarma

Després d'això, van implementar-se dos programes per verificar el funcionament de l'interruptor magnètic i el funcionament de l'altaveu. Abans, però, va ser necessària la connexió d'ambdós components a la placa. Va analitzar-se la documentació del fabricant relativa al port de propòsit general GPIO (vegeu figura 49):

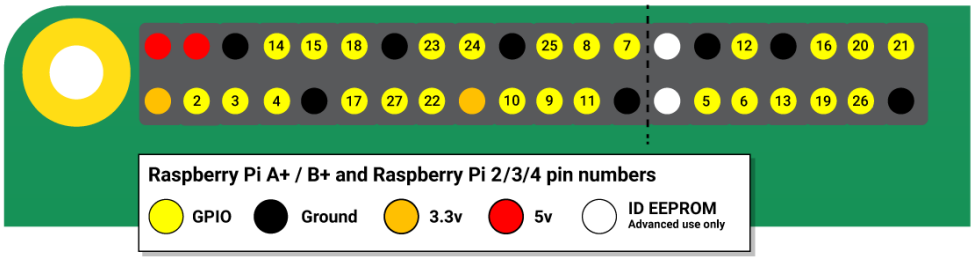


Figura 49. *Disposició i esquemàtic dels pins del port GPIO*. Extret de *Raspberry Pi*. Font: https://www.raspberrypi.org/documentation/usage/gpio/images/GPIO.png

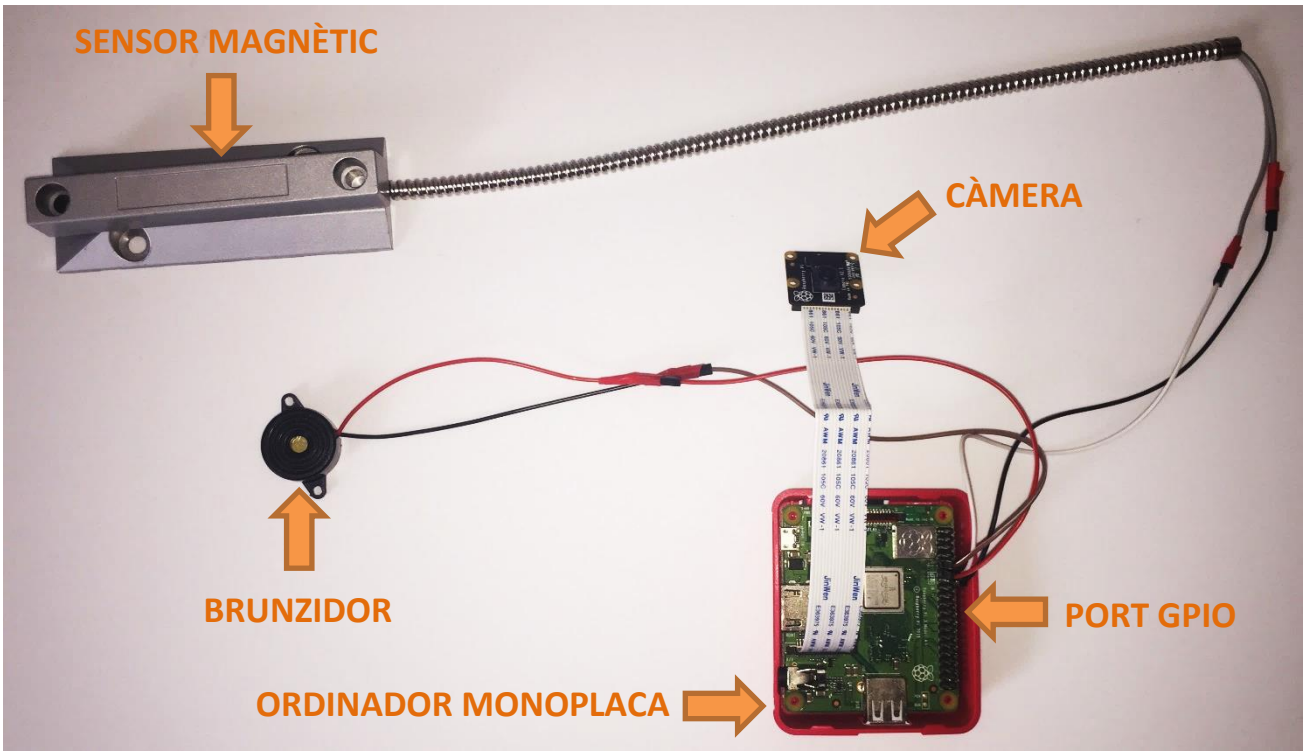
En el cas específic d'aquest projecte, tant pel sensor com pel brunzidor no era necessari que se'ls hi proporcionés una font d'alimentació. Es va connectar el cable de polaritat positiva del brunzidor al pin número 17 (vegeu figura 49), i el cable de polaritat negativa al pin situat a l'esquerra del pin 17. Quant al sensor magnètic, era indiferent quin dels dos cables connectar a la presa de terra (pin GROUND). El que es va fer va ser connectar un dels dos cables al port 18, i l'altre cable del sensor va connectar-se al port dret del 18, que és una presa de terra (GROUND).

El pas següent va ser la instal·lació de la llibreria *RPi* de *Python*, per seguidament desenvolupar els programes de prova de funcionament dels dos components.

El programa desenvolupat per provar el funcionament del brunzidor simplement assignava el valor cert al respectiu pin (en aquest cas, al pin número 17), configurat en mode sortida. En assignar el valor cert al pin on estava connectat el brunzidor feia sonar un senyal acústic constant.

Com que el sensor magnètic no estava connectat a cap font d'alimentació, el valor de lectura del pin número 18 seria indefinit, i en conseqüència el valor de lectura canviaria molt degut a les interferències que es produirien (cal tenir en compte que no està connectat a cap voltatge). Per tant, a l'hora de desenvolupar el programa de prova del component del sensor, va fer falta configurar el pin número 18 amb un «pull-up» (Desconegut, 2016). Després d'això, es va implementar un bucle continu que comprovés la lectura d'aquest pin, i quan el valor era cert es mostrava un missatge per pantalla informant de que les plaques magnètiques s'havien separat.

A continuació una imatge en la que s'aprecia les connexions dels components amb l'ordinador monoplaca:



Finalment, es van integrar els dos components perquè en separar-se les plaques magnètiques s'emetés un senyal acústic, com es pot veure a continuació:

```
1. is_active = True # Estat de l'alarma
2. is_open = None # Estat d'obertura de la porta
```

```
3.  old_is_open = None # Estat anterior d'obertura de la porta
4.
5.  # Event de canvi d'estat de la porta
6.  def door_change(channel):
7.      old_is_open = is_open # Guardem l'estat anterior
8.      is_open = GPIO.input(channel) # Lectura de l'estat actual
9.
10. GPIO.setmode(GPIO.BCM)
11.
12. # Configurem els pins i els modes
13. GPIO.setup(DOOR_SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
14. GPIO.setup(BUZZER_PIN, GPIO.OUT)
15.
16. # Inicialitzem la sortida del pin del brunzidor
17. GPIO.output(BUZZER_PIN, False)
18.
19. # Afegim l'event de canvi d'estat de la porta
20. GPIO.add_event_detect(DOOR_SENSOR_PIN, GPIO.RISING, callback=door_change,
    bouncetime=200)
21.
22. while True:
23.     # Emetem un so intermitent en cas de que la porta s'hagi obert i l'alarma
    estigui activada
24.     if is_active and is_open:
25.         GPIO.output(BUZZER_PIN, True)
26.         sleep(3)
27.         GPIO.output(BUZZER_PIN, False)
28.         sleep(1)
```

10.1.4. Sistema d'esdeveniments

El pas següent va ser començar a desenvolupar la infraestructura remota, que estaria allotjada a *Heroku* i seria l'element amb el qual l'alarma i el reconeixement facial interactuarien directament, amb la finalitat de notificar esdeveniments o bé enviar comandes d'activació, desactivació o consulta de l'estat d'algun dels dos sistemes.

En primer lloc, va fer falta crear una aplicació, que va anomenar-se «ripoll-alarm-api». Aquesta aplicació seria la que interactuaria amb els sistemes d'alarma i de reconeixement facial, rebria els missatges dels servidors de *Telegram* i seria l'aplicació amb la qual el portal web interactuaria per consultar esdeveniments i gestionar l'estat dels sistemes. La regió que es va seleccionar a l'hora de crear l'aplicació va ser la regió d'Europa, ja que d'aquesta manera podríem minimitzar la latència per proximitat geogràfica.

Posteriorment, per desenvolupar aquesta aplicació va consultar-se la documentació existent d'*Heroku* en matèria de WebSockets programats utilitzant *NodeJS* (Heroku, 2020).

El primer pas va ser el de descarregar les dependències necessàries per muntar un servidor de *WebSockets*. Fet això, es va procedir a implementar un servidor *WebSocket* de prova, que per consola escrivís quan un client s'ha connectat, ha enviat un missatge i quan s'ha desconnectat:

```
1. // Creem la aplicació Express
2. const api = express();
3.
4. // Registrem una ruta
5. api.get('/', (req, res) => res.send('Hola món!'));
6.
7. // Fem que l'aplicació escolti al port
8. api.listen(PORT, () => console.log('API: Servidor API escoltant peticions al port
   ' + PORT));
9.
10. // Creem el servidor WebSocket
11. const wss = new Server({ server: api });
12.
13. // Event de connexió d'un client WebSocket
14. wss.on('connection', (ws) => {
15.   // Event de missatge rebut d'un client WebSocket
16.   ws.on('message', function(message) {
17.           console.log('WSS: Rebut: %s', message);
18.           ws.send(message);
19.           });
20.
21.   console.log('WSS: Client connectat');
22.
23.   // Event de desconnexió d'un client WebSocket
24.           ws.on('close', function() {
25.                   console.log('WSS: Client desconnectat');
26.           });
27. });
```

Per tal de provar aquest servidor, va utilitzar-se la consola de desenvolupadors de *Google Chrome*, amb les següents comandes:

```
1. var ws = new WebSocket('wss://ripoll-alarm-api.herokuapp.com');
2. ws.send('Bona tarda!');
3. ws.close();
```

En executar aquest codi, s'establia una connexió amb el servidor *WebSocket* muntat, s'enviava la cadena de text «Bona tarda» i es tancava la connexió.

Després d'haver assegurat el funcionament correcte d'aquest servidor *WebSocket*, es va implementar un servei web (per muntar l'API, que seria utilitzada per *Telegram* en arribar missatges i pel portal web) que fos paral·lel amb el servidor *WebSocket*. En aquest punt, va sorgir l'inconvenient que les aplicacions executades a *Heroku* reben el tràfic a través d'un encaminador o proxy gestionat per *Heroku.* Això va significar que tant servei web com servidor *WebSocket* havien d'utilitzar el mateix port (assignat de forma automàtica per la plataforma). Com els dos serveis no podien estar utilitzant el mateix port per rebre tràfic, es va crear el servei web i es va «escoltar» l'esdeveniment d'Upgrade produït quan s'intenta establir una connexió *WebSocket*. Anteriorment ja s'ha explicat que quan s'estableix una connexió *WebSocket*, es fa

una petició GET a l'adreça del *WebSocket,* i com a capçalera s'inclou un paràmetre que indica que el client vol establir una connexió *WebSocket*. Per tant, va ser necessari fer el següent:

```
1. // Creació del servidor WebSocket sense cap port associat
2. const wss = new Server({ noServer: true });
3.
4. // Creació de l'aplicació Express
5. const app = express();
6. // Obtenim el servidor HTTP
7. const server = http.createServer(app);
8.
9. // Recepció de l'event `Upgrade` causat per un client que vol connectar-se al
   servidor WebSocket
10. server.on('upgrade', function (req, socket, head) {
11.   // Fem que el servidor de WebSocket rebi l'event `Upgrade`
12.   wss.handleUpgrade(request, socket, head, function done(ws) {
13.           // Emetem manualment l'event de connexió d'un client
14.           wss.emit('connection', ws, request);
15.   });
16. });
```

D'aquesta forma va ser possible el paral·lelisme d'ambdós serveis, sense que es causessin interferència entre ells.

Després es va prosseguir amb la creació del bot de Telegram, i va ser necessari iniciar una conversació amb el bot oficial anomenat *BotFather.* Per crear un nou bot, se li ha d'enviar la comanda "/newbot" i proporcionar-li un nom i un nom d'usuari per al nou bot (vegeu figura 50).

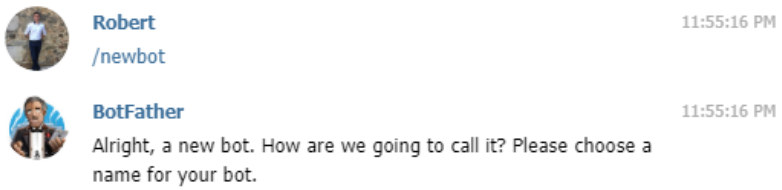


Figura 50. *Creació d'un bot de Telegram.*

Seguidament es va instal·lar la llibreria *TelegramBot* per a *NodeJS,* per poder enviar i rebre missatges programàticament. Rebre missatges de *Telegram* només era possible si es creava un *WebHook*, al que *Telegram* pogués enviar els missatges que rep el bot. Per això, es va crear una ruta al servei web per poder rebre els esdeveniments enviats per *Telegram* i d'aquesta manera poder atendre a les peticions enviades pels usuaris:

```
1. // Configurem el codi que s'executarà en rebre una petició POST a la ruta
   `/telegram`
2. app.post('/telegram', (req, res) => {
3.     // Missatge rebut
4.     let message = req.body.message.text.toLowerCase();
5.     // ID de la conversació
```

```
6.      let chatId = req.body.message.chat.id;
7.      // ID de l'usuari emissor
8.      let userId = req.body.message.from.id;
9.      // Nom de l'usuari emissor
10.     let userName = req.body.message.from.first_name;
11.
12.     // Enviem un missatge a la conversació
13.     bot.sendMessage(chatId, "Hola!");
14.
15.     // Responem la petició POST amb una cadena de text buida i un estat HTTP 200
16.     res.send('');
17. });
18.
19. // Creem l'objecte bot
20. let bot = new TelegramBot(TELEGRAM_TOKEN);
21. // Configurem la URL del WebHook perquè apunti a la nostra aplicació
22. bot.setWebHook('https://ripoll-alarm-api.herokuapp.com/telegram');
```

A partir d'aquest instant va haver-se d'implementar la lògica darrere la recepció dels missatges, de manera que s'interpretessin correctament, es comprovés si l'usuari estava registrat i si la comanda era vàlida.

Abans de continuar amb la implementació de la lògica dels missatges de *Telegram* i la lògica dels missatges rebuts a través dels *WebSockets*, va fer falta instal·lar *PostgreSQL* a l'aplicatiu (vegeu figura 51).

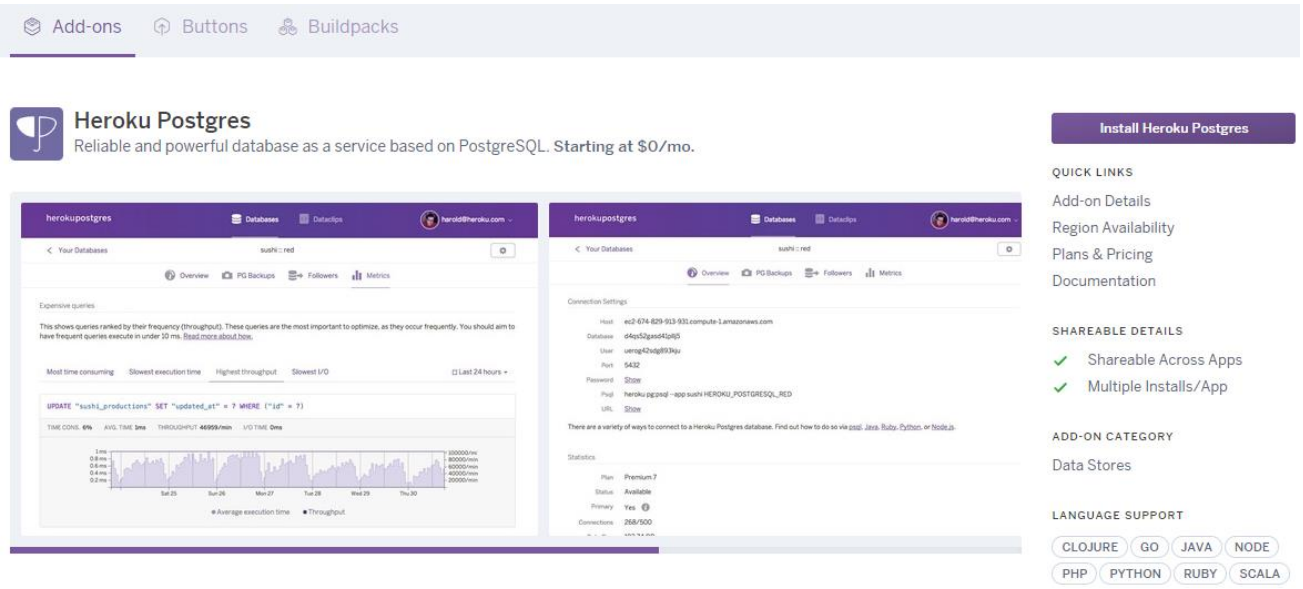


Figura 51. *Pantalla d'activació de l'addon Heroku Postres.*

Seguidament van crear-se les taules del model de dades exposat en apartats anteriors.

Fet això, va procedir-se a implementar la lògica dels missatges de Telegram, i fet això va haver-se d'implementar la lògica dels *WebSockets*, per tal de processar adequadament els missatges rebuts pel sistema d'alarma i el sistema de reconeixement facial. Per afegir un mecanisme de seguretat que rebutgés qualsevol connexió no autoritzada (i per tant, només permetre connexions legítimes del sistema d'alarma i el

sistema de reconeixement facial), va modificar-se lleugerament l'esdeveniment de connexió dels WebSockets. Van generar-se dues claus aleatòries diferents per a cada component, i aquestes claus haurien d'anar a l'URL utilitzada per connectar-se al WebSocket. A més a més, a causa del problema exposat en apartats anteriors, va ser necessari crear un WebSocket separat per a cada direcció (d'entrada i de sortida). La direcció del WebSocket que es connectava també es trobava a l'URL:

```
1. // Event `upgrade` quan un client s'intenta connectar al servidor WebSocket
2. server.on('upgrade', (req, socket, head) => {
3.     let obj = verifySocketKey(req, socket);
4.
5.     if (obj != null) {
6.         // Passem l'event `upgrade` al servidor WebSocket
7.         wss.handleUpgrade(req, socket, head, function done(ws) {
8.             wss.emit('connection', { websocket: ws, alarm: (obj.component ===
   'alarm'), mode: obj.mode }, req);
9.         });
10.    }
11. });
```

```
1. function verifySocketKey(req, socket)
2. {
3.     // La clau d'autenticació i el mode s'especifiquen a la URL (ws://url-
   api.com/clau/mode)
4.     let splitted = req.url.split('/');
5.     let key = splitted[1]; // Clau d'autenticació
6.     let mode = splitted[2]; // Mode del WebSocket
7.
8.     // Si la clau especificada no coincideix amb la clau de l'alarma o la clau del
   reconeixement facial, finalitzem l'intent de connexió
9.     if ((key != global.ALARM_KEY && key != global.FACEREC_KEY) || (mode != 'rx' &&
   mode != 'tx'))
10.    {
11.        socket.write('HTTP/1.1 401 Web Socket Protocol Handshake\r\n' +
12.            'Upgrade: WebSocket\r\n' +
13.            'Connection: Upgrade\r\n' +
14.            '\r\n'
15.        );
16.        socket.destroy();
17.        return null;
18.    }
19.
20.    return { component: (key === global.ALARM_KEY ? 'alarm' : 'facerec'), mode:
   mode };
21. }
```

Després va ser necessari establir quin protocol de comunicació s'utilitzaria entre el servidor WebSocket i els sistemes d'alarma i de reconeixement facial. Van definir-se els següents missatges:

- **Enviats pel servidor WebSocket (els missatges s'envien a un sistema o a l'altre en funció del component sobre el qual es vulgui executar la comanda):**
  - Activació de sistema (alarma o reconeixement facial)

```
{
    "action": "ON",
    "data": {
        "user_id": 1 // ID del resident que ha activat el sistema
    }
}
```

- Desactivació de sistema (ídem)

```
{
    "action": "OFF",
    "data": {
        "user_id": 1 // ID del resident que ha desactivat el sistema
    }
}
```

- Consulta d'estat del sistema (ídem)

```
{
    "action": "STATUS",
    "data": {
        "user_id": 1 // ID del resident que ha consultat l'estat del sistema
    }
}
```

- **Enviats pel sistema d'alarma i pel sistema de reconeixement facial:**
  - Confirmació d'activació/desactivació del sistema

```
{
    "type": "ON", // ON = confirmació d'activació || OFF = confirmació de desactivació
    "data": {
        "user_id": 1 // ID del resident que ha modificat l'estat
    },
    "raised_at": 1567628419 // Timestamp UNIX del moment del canvi d'estat
}
```

  - Resposta a consulta de l'estat

```
{
    "type": "STATUS",
    "data": {
        "status": "ON", // Estat actual del sistema (ON o OFF)
        "user_id": 1 // ID del resident que ha sol·licitat l'estat
    },
    "raised_at": 1567628419 // Timestamp UNIX del moment de la creació del missatge
}
```

- **Enviats pel sistema d'alarma:**

- Intrusió

```
{
    "type": "INTRUSION",
    "data": {},
    "raised_at": 1567628419 // Timestamp UNIX del moment de la intrusió
}
```

- **Enviats pel sistema de reconeixement facial:**
  - Reconeixement

```
{
  "type": "RECOGNITION",
  "data": {
          "image": "IMATGE", // Imatge codificada en base64
          "person": "Robert" // Nom de la persona reconeguda (o "desconegut")
  },
  "raised_at": 1567628419 // Timestamp UNIX del moment del reconeixement
}
```

Amb el protocol definit, van adaptar-se els programes del sistema d'alarma i del sistema de reconeixement facial perquè cadascun dels sistemes establissin dues connexions unidireccionals amb el servidor web. En ambdós sistemes, va implementar-se un mecanisme de paral·lelisme, de manera que les connexions WebSocket no afectessin al funcionament dels dos sistemes ni els sistemes afectessin l'enviament i recepció de missatges. Per aquest motiu, es van dissenyar i implementar els següents components:

- **Receptor de missatges**

Aquest subprocés estava escoltant permanentment al WebSocket de direcció entrant per afegir els missatges rebuts a una cua de processament, que seria iterada pel subprocés processador. D'aquesta manera, el processament es realitzava en un altre subprocés i d'aquesta manera no es bloquejava la recepció d'altres missatges.

```
async def websocket_listener(send_queue, process_queue):
    # Connexió al servidor WebSocket
    websocket = await connect(WEBSOCKET_URL)

    while True:
        try:
            # Esperem a rebre un missatge
            message = await websocket.recv()

            # Convertim el missatge JSON a un diccionari Python
            json_message = json.loads(message)
            # Inserim el diccionari a la cua de missatges per processar
            process_queue.put(json_message)
```

```
14.
15.         # La connexió s'ha tancat inesperadament
16.         except websockets.exceptions.ConnectionClosedError:
17.             # Reintentem la connexió
18.             websocket = await connect(WEBSOCKET_URL)
19.
20.         # Error de parsing del missatge
21.         except ValueError:
22.             pass
23.
24.         # Interrupció de teclat (Control-C)
25.         except KeyboardInterrupt:
26.             await websocket.close()
27.             break
```

- **Emissor de missatges**

Aquest subprocés consultava una cua de missatges a enviar, que era compartida i emplenada pels processos de processament de missatges i del sistema (d'alarma o bé de reconeixement facial). Disposava d'una cua interna per missatges que no s'han pogut enviar, ordenats de forma que s'enviessin primer els missatges que feia més estona que s'haurien d'haver enviat, utilitzant una cua de prioritat. Per tant, prioritzava la cua de missatges d'enviament fallat respecte a la cua compartida entre processos de missatges a enviar. En conseqüència, quan l'enviament d'un missatge fallava, aquest missatge s'inseria a la cua interna de prioritat, i s'anava reintentant l'enviament d'aquest missatge fins que era satisfactori, i seguidament s'enviaven els altres missatges de la cua interna fins a quedar buida. Un cop la cua interna quedava buida es passaven a enviar els missatges que es trobaven a la cua compartida.

El subprocés emissor de missatges del sistema de reconeixement facial és el següent:

```
1.  async def websocket_sender(send_queue):
2.      # Ens connectem al websocket remot
3.      websocket = await connect(WEBSOCKET_URL)
4.      pending_queue = PriorityQueue() # Cua de missatges pendents per enviar un cop
    hi hagi connexió
5.
6.      while True:
7.          try:
8.              # No tenim missatges pendents prioritaris
9.              if pending_queue.empty():
10.                 # Agafem el primer missatge de la cua compartida d'enviament
11.                 message = send_queue.get()
12.
13.                 # El tipus de missatge a enviar és de reconeixement
14.                 if message['type'] == 'RECOGNITION':
15.                     image = message['data']['image']
16.
17.                     # Transformem la imatge a una cadena de text en base64
18.                     ret, buffer = imencode('.jpg', image)
19.                     base64_image = b64encode(buffer).decode()
20.
21.                     message['data']['image'] = base64_image
```

```
22.
23.             # Tenim missatges pendents prioritaris, que ja s'haurien d'haver
    enviat fa estona
24.             else:
25.                 message = pending_queue.get()[1]
26.
27.             # Enviem el missatge
28.             await websocket.send(json.dumps(message))
29.
30.         # Error enviant el missatge
31.         except websockets.exceptions.ConnectionClosedError:
32.             # Inserim el missatge que hem intentat enviar per ordre de creació del
    missatge
33.             pending_queue.put((message['raised_at'], message))
34.             # Intentem reconnectar al WebSocket
35.             websocket = await connect(WEBSOCKET_URL)
36.
37.         # Error transformant el diccionari a string JSON
38.         except ValueError:
39.             pass
40.
41.         # Interrupció de teclat (Control-C)
42.         except KeyboardInterrupt:
43.             await websocket.close()
44.             break
```

- **Processador de missatges rebuts**

Aquest procés varia en funció dels missatges que pot rebre el sistema, és a dir, si el processador s'ha d'implementar pel sistema d'alarma, s'executaran unes accions o es respondrà d'una manera concreta, però diferent de la manera i a les accions que s'executarien si el processador s'ha d'implementar pel sistema de reconeixement facial. Com que és possible modificar remotament l'estat dels sistemes, va haver-se d'introduir un *lock* per evitar que hi hagués conflictes amb la variable d'estat, que és compartida pel procés que processa els missatges entrants i pel mateix procés del sistema. A continuació, el codi del processador pel sistema del reconeixement facial:

```
1.  def message_processor(process_queue, send_queue, is_active, is_active_lock):
2.      while True:
3.          # Agafem el primer missatge de la cua de processament
4.          message = process_queue.get()
5.
6.          # S'ha rebut un missatge preguntant per l'estat del sistema
7.          if message['action'] == 'STATUS':
8.              # Esperem a obtenir el recurs (la variable `is_active`)
9.              is_active_lock.acquire(timeout=1)
10.             current_status = is_active.value
11.             is_active_lock.release()
12.
13.             # Posem el missatge de resposta a la cua d'enviament
14.             send_queue.put_nowait(
15.                 {
16.                     "type": "STATUS",
```

```
                    "data": {
                        "status": ('ON' if current_status else 'OFF'),
                        "user_id": message['data']['user_id']
                    },
                    "raised_at": int(time())
                }
            )

        # S'ha rebut un missatge de canvi d'estat (apagar/encendre) del sistema
        elif message['action'] == 'ON' or message['action'] == 'OFF':
            # Esperem a obtenir el recurs (la variable `is_active`) per poder-ne canviar el valor
            is_active_lock.acquire(timeout=1)
            is_active.value = True if message['action'] == 'ON' else False
            is_active_lock.release()

            # Posem el missatge de confirmació de canvi d'estat a la cua d'enviament
            send_queue.put_nowait(
                {
                    "type": message['action'],
                    "data": {
                        "user_id": message['data']['user_id']
                    },
                    "raised_at": int(time())
                }
            )
```

- **Sistema d'alarma o sistema de reconeixement facial**

Aquest procés és el que conté el sistema en qüestió, sigui l'alarma o el reconeixement facial. El component tindrà accés a la cua d'enviament de missatges per poder notificar d'esdeveniments produïts i també disposarà d'accés al *lock* utilitzat per limitar l'accés a la variable global de l'estat del component.

El codi del component del sistema d'alarma és el següent:

```
def alarm(send_queue, is_active, is_active_lock):
    ...

    old_is_open = None
    is_open = None

    try:
        while True:
            old_is_open = is_open # Guardem l'estat anterior
            is_open = GPIO.input(DOOR_SENSOR_PIN) # Lectura de l'estat actual

            # La porta actualment està oberta
            if is_open:
                # S'ha produït un canvi d'estat (la porta abans estava tancada)
                if old_is_open != is_open:
                    send_queue.put_nowait({ "type": "INTRUSION", "data": {}, "raised_at": int(time()) })

```

```
18.                 # Intentem obtenir el lock per llegir el valor de `is_active`,
    sense bloquejar l'execució
19.                 locked = is_active_lock.acquire(block=False)
20.
21.                 # Hem aconseguit el lock
22.                 if locked:
23.                     # Obtenim el valor i alliberem el lock
24.                     is_active_value = is_active.value
25.                     is_active_lock.release()
26.                 # No hem pogut aconseguir el lock
27.                 else:
28.                     # Valor per defecte
29.                     is_active_value = True
30.
31.                 # Si l'alarma està activada
32.                 if is_active_value:
33.                     # Emetem so
34.                 # Si l'alarma està desactivada
35.                 else:
36.                     # No emetem so
```

El següent pas va ser implementar al servidor la lògica dels missatges enviats a través de WebSockets, i unificar-ho amb el bot de *Telegram*.

Posteriorment va enllaçar-se el funcionament del servidor web amb la base de dades, perquè es guardessin els esdeveniments que es produïen, per persistir els residents i per guardar les imatges capturades pel sistema de reconeixement facial.

L'estructura de directoris i fitxers d'aquest sistema va quedar de la següent forma:

```
│   .gitignore
│   package-lock.json
│   package.json
│   server.js
│
└───app
    │   constants.js
    │   telegram.js
    │
    ├───controllers
    │       apiController.js
    │
    ├───db
    │       db.js
    │       event.js
    │       pool.js
    │       resident.js
    │       telegramChat.js
    │       variable.js
    │
    ├───helpers
    │       utils.js
    │       validator.js
    │       webSocket.js
    │
    ├───middleware
```

```
    │       verifyApiKey.js
    │       verifySocketKey.js
    │
    └──routes
            apiRoutes.js
            telegramRoutes.js
```

Al fitxer «server.js» és on s'inicialitza el servidor web, el servidor *websocket,* el bot de *Telegram* i es defineixen les rutes que haurà de servir el servidor web.

Dins la carpeta «app» es troben tots els fitxers necessaris per al funcionament de l'aplicació. Al fitxer «constants.js» de dins aquesta carpeta es defineixen variables i valors constants utilitzats a tot el programa, i a "telegram.js" hi ha definida la classe "Telegram", per permetre enviar missatges, imatges i conté la lògica del bot quan es reben missatges.

Al fitxer «apiController» de dins la carpeta "controllers" s'hi troben les funcions de cadascuna de les rutes relatives a l'API, que serà utilitzada pel portal web.

A la carpeta «db» s'hi troba tot el codi relacionat amb la interacció amb la base de dades, a més de les entitats amb les quals es treballen ("event", "resident", "telegramChat" i "variable").

Seguidament, a la carpeta «helpers» es troben components i funcions miscel·lànies que s'utilitzen arreu de l'aplicació. El fitxer «webSocket» conté tota la lògica relativa a les connexions *websocket*, que inclou l'enviament de missatges i recepció de missatges dels components.

Al directori «middleware» s'hi troba el fitxer "verifySocketKey", que conté la funció que comprova la clau d'autenticació i la direcció dels missatges de les connexions *websocket* (d'acord amb els paràmetres especificats a l'URL de connexió). Al fitxer «verifyApiKey» s'hi troba la funció que verificarà que les crides API contenen la clau que ha d'utilitzar el portal web per interactuar amb l'API.

Per acabar, al directori «routes» es troben totes les definicions de les rutes de l'API, que seran servides pel servidor web.

Finalment, es va provar tota la unificació del sistema per validar que el funcionament fos correcte.

10.1.5. Portal web

Per desenvolupar aquest component, va crear-se una aplicació *Heroku* nova, anomenada «ripoll-alarm-web».

Es va començar per la instal·lació de dependències: *ExpressJS* (com a servidor web), *Passport* (per l'autenticació) i *Pug* (el motor de plantilles).

El pas posterior va ser la implementació dels dissenys de les interfícies, que van quedar de la següent manera (vegeu figures 52, 53, 54 i 55):

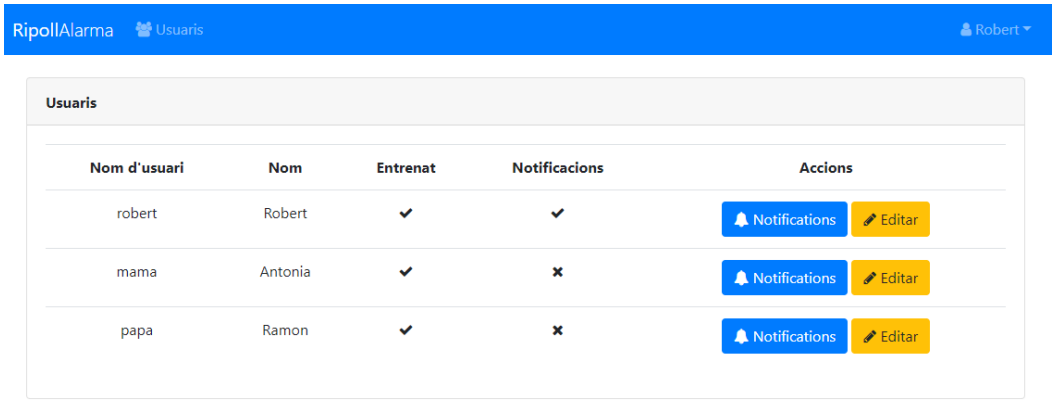


Figura 52. *Captura de pantalla de la pàgina de gestió d'usuaris.*

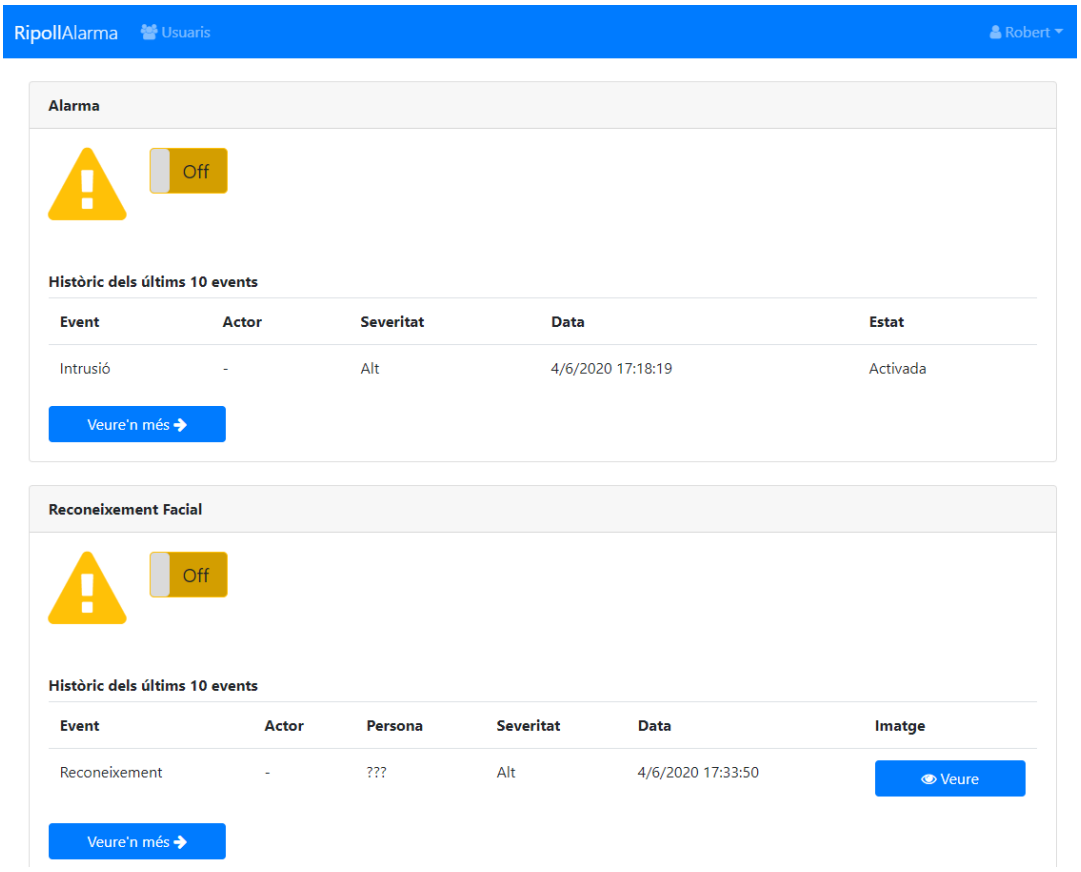


Figura 53. *Captura de pantalla de la pàgina principal del portal web.*

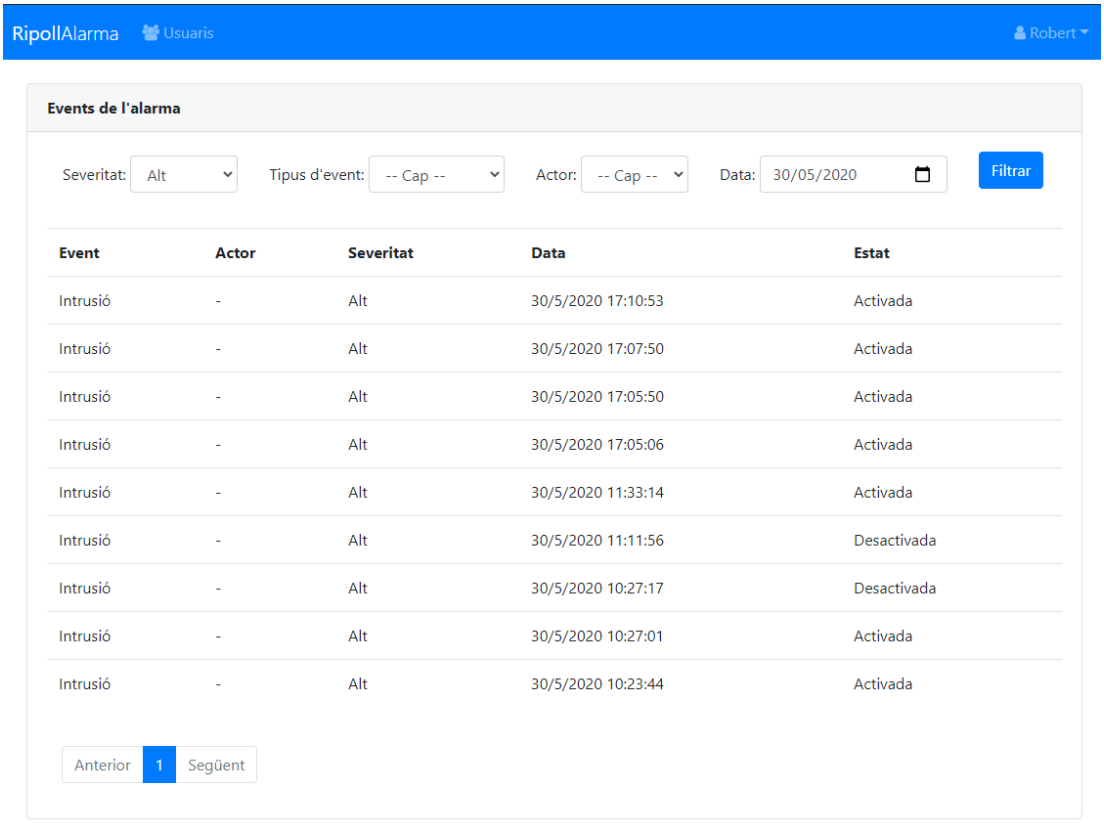


Figura 54. *Captura de pantalla del llistat d'esdeveniments del sistema d'alarma.*

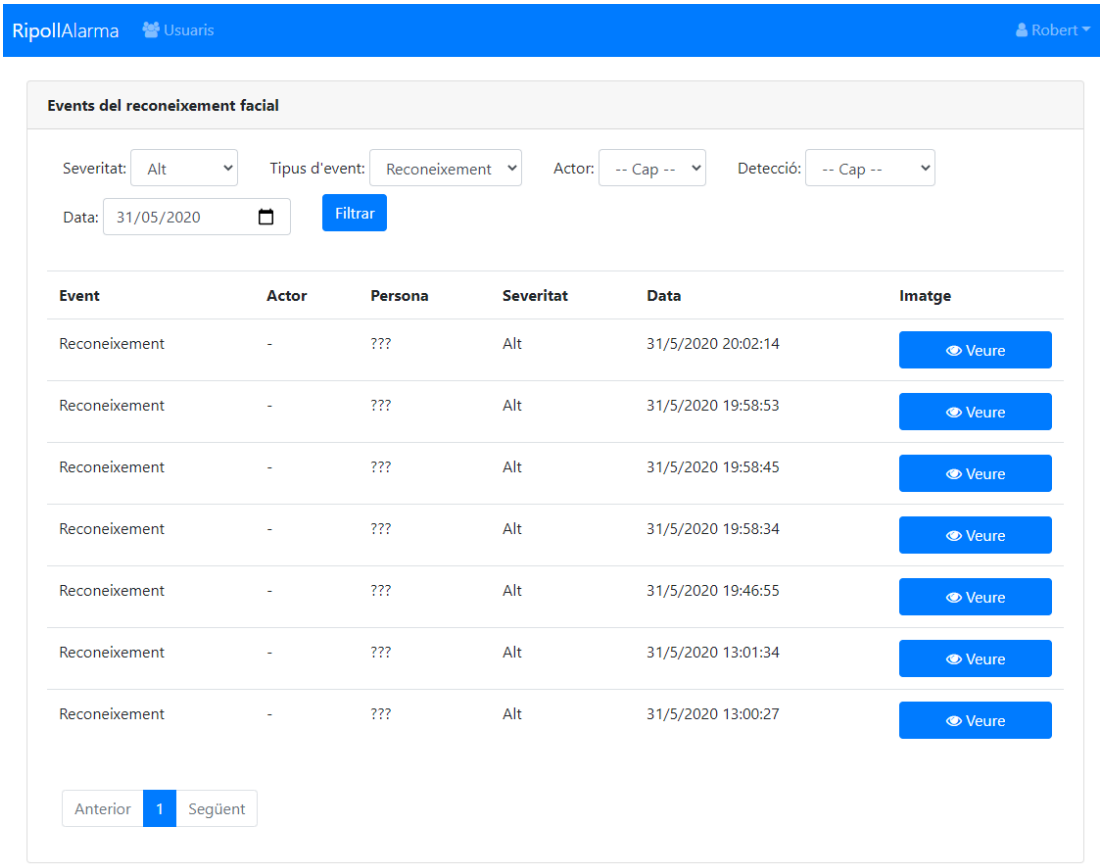


Figura 55. *Captura de pantalla del llistat d'esdeveniments del reconeixement facial.*

Després d'implementar les interfícies, es va procedir a implementar el codi que realitzaria peticions a l'API del sistema d'esdeveniments.

Posteriorment, es va prosseguir amb la implementació del sistema d'autenticació.

Finalment va provar-se la totalitat del sistema i tot funcionava degudament, d'acord amb els requisits definits a la fase inicial del projecte.

11. Conclusions

L'objectiu inicial d'aquest projecte era el de desenvolupar una alarma domèstica amb reconeixement facial, capaç de reconèixer els rostres dels residents i enviar notificacions en temps real als residents, a més de permetre la gestió de la totalitat del sistema des d'un portal web.

Després d'haver desenvolupat el projecte i d'acord amb els resultats de les proves, pot afirmar-se que s'han complert satisfactòriament tots els requisits definits inicialment.

En l'àmbit de la càmera van produir-se desviacions, ja que la idea inicial era aprofitar l'espiell ja instal·lat a la porta del domicili, però no va acabar sent factible per culpa de la baixa qualitat del borescopi, entre altres complicacions. Malgrat això, s'ha acabat implementant un reconeixement facial òptim i funcional capaç de distingir els rostres dels residents dels de possibles intrusos, utilitzant una càmera funcional en situacions de baixa lluminositat.

Perquè el producte desenvolupat al llarg del projecte pogués comercialitzar-se, caldria implantar mesures de seguretat físiques, especialment amb la càmera, ja que actualment podria ser manipulada de forma fàcil. Això també és aplicable a la placa, ja que caldria fabricar a mida una carcassa de metall antimanipulacions.

Aquest treball ha sigut idoni per aprendre conceptes de seguretat, multiprocessat, visió per computador, reconeixement facial, intel·ligència artificial, PaaS, components electrònics, ordinadors monoplaca (concretament *Raspberry Pi*), comunicacions a través de *websocket*. També ha ajudat a consolidar aspectes explicats al llarg de la carrera, com: computació paral·lela, xarxes i comunicacions, bases de dades i sistemes de gestió de bases de dades, electrònica i intel·ligència artificial.

12. Treball futur

A continuació es presenta un llistat (no ordenat) de les accions que es podrien dur a terme per millorar la fiabilitat, seguretat i qualitat del sistema d'alarma i el sistema de reconeixement facial:

- Utilitzar *Twilio*[8] per realitzar trucades telefòniques automatitzades al propietari de l'habitatge per informar-lo d'una intrusió al domicili, útil en el cas que aquest es trobi en una zona sense Internet (per exemple, a l'estranger).
- La càmera utilitzada al projecte pot capturar imatges en situacions de baixa i alta lluminositat. Tanmateix, quan la lluminositat és quasi nul·la (cap de les llums de l'espai comú on es troba la porta del domicili està oberta), només captura foscor, i en aquestes situacions el sistema de reconeixement facial no és capaç de detectar ni reconèixer rostres. Un altre inconvenient d'aquesta càmera és que l'angle de visió que té és de només 62°, mentre que un espiell tradicional de porta pot arribar a tenir un camp de visió de fins a 200°. La solució seria adquirir una càmera compatible amb la *Raspberry Pi* que disposi d'una lent d'ull de peix (amb un camp de, per exemple, 160°) i d'un LED infraroig, i d'aquesta manera el reconeixement facial seria capaç de funcionar degudament en situacions de lluminositat nul·la i deixaria de tenir tanta dependència amb l'altura on estigués col·locada la càmera.
- Adquisició d'un sistema d'alimentació ininterrompuda que garantís l'execució de l'alarma en cas que la font d'alimentació primària de la placa deixés de proporcionar corrent (ja sigui per un tall elèctric, avaria, etc.).
- Desenvolupar i muntar una interfície d'usuari física, que disposés de tecles (com les alarmes tradicionals) per controlar l'estat de l'alarma de forma presencial, en comptes d'haver de gestionar l'alarma a través del portal web. A més a més, també podria tenir llums informatives de l'estat actual de l'alarma, i fins i tot un lector d'empremta dactilar perquè en posar-hi el dit es desactivés automàticament l'alarma.
- Implementar una emissió de vídeo en temps real al portal web, de forma que es permetés als residents veure en temps real què és el que està succeint a l'exterior del domicili, útil en cas que algun individu toqui el timbre i l'habitant vulgui visualitzar qui és. Addicionalment, també podria instal·lar-se una pantalla digital que permetés visualitzar aquesta emissió sense necessitat d'accedir al portal web.
- Connectar un mòdul de connectivitat 4G a la placa, que fos utilitzat pel sistema en detectar la pèrdua de connectivitat a la xarxa Wi-Fi del domicili.

[8] Consulteu l'annex 2 per llegir informació sobre la plataforma *Twilio*.

- Fabricar una estructura física de metall a prova de manipulacions externes, on al seu interior s'hi trobés la placa. Amb aquesta estructura, podrien disminuir-se danys intencionats amb la finalitat de destruir o aturar l'alarma o altres manipulacions dels intrusos. Per altra banda, també seria convenient protegir la integritat de la càmera que capturés els rostres, ja que l'assaltant podria inutilitzar-la.
- Adaptar el sistema de reconeixement facial perquè el propietari pogués entrenar-lo per tal que fos capaç de reconèixer nous rostres. En aquest projecte només s'ha considerat la captura de vídeo d'entrenament i el mateix entrenament com a activitats desenvolupades pel fabricant de l'alarma.
- Adquirir una altra placa amb més potència computacional per tal d'evitar perdre qualitat d'imatge a l'hora de detectar i reconèixer rostres. El sistema només captura en qualitat 480p per minimitzar el temps de processament dels *frames* captats per la càmera. Si es disposés de més potència computacional, es podrien processar *frames* de més resolució, sense incrementar el temps de processament.
- Implementar un sistema de control de presència, per saber en tot moment si algun dels residents es troba a casa i a quina hora ha arribat. De forma addicional també podria guardar un històric amb les arribades i sortides de cadascun dels residents del domicili.
- Implementar un sistema per detectar falsificacions en els rostres capturats per la càmera, és a dir, determinar la «realitat» de la cara capturada, de manera que pugui determinar-se si prové de la pantalla d'un mòbil (Rosebrock, Liveness Detection with OpenCV, 2019).

13. Bibliografia

Acebo, E. d. (2019). *Aprenentatge automàtic: Problemes de classificació. El model KNN (K Nearest Neighbours).* Recollit de UdGMoodle.

Agencia Española de Protección de Datos. (2019). *Guía sobre el uso de videocámaras para seguridad y otras finalidades.* Consultat el 5 / març / 2020, a https://www.aepd.es/sites/default/files/2019-09/guia-videovigilancia.pdf

Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (25 / gener / 2016). *Free and open source face recognition with deep neural networks*. Recollit de OpenFace: https://github.com/cmusatyalab/openface

ArduCam. (21 / maig / 2020). *NO IR Camera.* Recollit de ArduCam: https://www.arducam.com/docs/cameras-for-raspberry-pi/noir-camera/

Arkbauer. (2019). *How Much Does Custom Software Development Cost?* Recollit de Arkbauer: https://arkbauer.com/blog/custom-software-development-cost/#europe

Arrow. (14 / agost / 2018). *What is a Reed Switch and How Does it Work?* Recollit de Arrow: https://www.arrow.com/en/research-and-events/articles/the-reed-switch-ingeniously-simple-sensing

Axarnet. (4 / desembre / 2018). *¿Qué es IaaS, PaaS e Saas? Conóce la diferencias*. Recollit de Axarnet: https://axarnet.es/blog/saas-paas-iaas#paas

Barney, B. (9 / juny / 2020). *Introduction to Parallel Computing*. Recollit de Lawrence Livermore National Laboratory: https://computing.llnl.gov/tutorials/parallel_comp

Barrios, J. (sense data). *Redes Neuronales Convolucionales.* Recollit de Juan Barrios: https://www.juanbarrios.com/redes-neurales-convolucionales

Chetu. (24 / agost / 2018). *¿Qué es Twilio y Cómo Puede Funcionar Para su Negocio?* Recollit de Medium: https://medium.com/@simonbrady85/qu%C3%A9-es-twilio-y-c%C3%B3mo-funciona-para-su-negocio-e96d67be3bb6

Código Facilito. (sense data). *¿Qué es el GIL en Python?* Recollit de CódigoFacilito: https://codigofacilito.com/articulos/gil-python

De Máquinas y Herramientas. (26 / setembre / 2013). *¿Qué es un Boroscopio?* Recollit de De Máquinas y Herramientas: https://www.demaquinasyherramientas.com/herramientas-de-medicion/boroscopio

Desconegut. (9 / febrer / 2016). *raspberry-gpio-python*. Recollit de SourceForge: https://sourceforge.net/p/raspberry-gpio-python/wiki/Inputs/

ExpressJS. (1 / abril / 2020). *Hello world example*. Recollit de ExpressJS: https://expressjs.com/en/starter/hello-world.html

Franco, L. (26 / febrer / 2008). *Modelos Computacionales.* Recollit de Universidad de Málaga: http://www.lcc.uma.es/~lfranco/Clase2-modelos-de-neuronas.pdf

Frías, E. R. (sense data). *Heroku - Una plataforma para la creación de aplicaciones*. Recollit de Esteban Romero: https://estebanromero.com/herramientas-emprender-desarrollar-proyectos/heroku-una-plataforma-para-la-creacion-de-aplicaciones/

Gavrailova, G. (8 / maig / 2019). *¿Qué es un webhook y por qué lo necesitas?* Recollit de Mailjet: https://es.mailjet.com/blog/news/que-es-webhook/

Geeky Theory. (sense data). *Qué es una API REST y para qué se utiliza*. Recollit de Geeky Theory: https://geekytheory.com/que-es-una-api-rest-y-para-que-se-utiliza

Geitgey, A. (24 / juliol / 2016). *Modern Face Recognition with Deep Learning*. Recollit de https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Geitgey, A. (13 / desembre / 2017). *Install dlib and face_recognition on a Raspberry Pi*. Recollit de GitHub: https://gist.github.com/ageitgey/1ac8dbe8572f3f533df6269dab35df65

Geitgey, A. (2020). *The world's simplest facial recognition api for Python and the command line.* Recollit de GitHub: https://github.com/ageitgey/face_recognition

Gómez, E. (12 / abril / 2017). *TWILIO: Servicios telefónicos en la nube.* Recollit de Kabel: https://www.kabel.es/twilio-servicios-telefonicos-en-la-nube/

González, A. (sense data). *¿Qué es Machine Learning?* Recollit de Cleverdata: https://cleverdata.io/que-es-machine-learning-big-data/

Haverbeke, M. (2018). Asynchronous Programming. A M. Haverbeke, *Eloquent Javascript* (p. 181-183). 4: No Starch Press. Recollit de Eloquent Javascript.

Heroku. (13 / setembre / 2019). *HTTP Routing*. Recollit de Heroku Dev Center: https://devcenter.heroku.com/articles/http-routing#timeouts

Heroku. (14 / agost / 2019). *WebSockets on Heroku*. Recollit de Heroku Dev Center: https://devcenter.heroku.com/articles/websockets#timeouts

Heroku. (6 / juny / 2020). *HTTP Routing*. Recollit de Heroku Dev Center: https://devcenter.heroku.com/articles/http-routing

Heroku. (23 / maig / 2020). *Using WebSockets on Heroku with Node.js*. Recollit de Heroku Dev Center: https://devcenter.heroku.com/articles/node-websockets

Heroku. (sense data). *How do I set the timezone on my dyno?* Recollit de Heroku: https://help.heroku.com/JZKJJ4NC/how-do-i-set-the-timezone-on-my-dyno

Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (9 / gener / 2018). *Labeled Faces in the Wild Home*. Recollit de Labeled Faces in the Wild: http://vis-www.cs.umass.edu/lfw/

Malik, D. (3 / juliol / 2019). *Asynchronous Programming 101*. Recollit de Hackernoon: https://hackernoon.com/asynchronous-programming-101-5ac0fa4e84e8

Manorshi. (16 / agost / 2019). *What's the difference between active buzzers and passive buzzers?* Recollit de Manorshi: https://www.manorshi.com/What-s-the-difference-between-active-buzzers-and-passive-buzzers-id3333285.html

Ministerio del Interior. (2011). *Balance de Criminalidad 2011.* Consultat el 1 / març / 2020, a http://www.interior.gob.es/documents/10180/1209011/Balance+de+Criminalidad+2011.pdf/f5945060-0cf1-4e0e-9a69-f2953d2c70d4

Ministerio del Interior. (2018). *Balance de Criminalidad: Cuarto Trimestre 2018.* Consultat el 1 / març / 2020, a http://www.interior.gob.es/documents/10180/8736571/informe+balance+2018+4%C2%BA%20trimestre.pdf/fb51653e-77f5-44da-9d23-535dbf4b2edd

Ministerio del Interior. (2019). *Balance de Criminalidad: Cuarto trimestre 2019.* Consultat el 1 / març / 2020, a http://www.interior.gob.es/documents/10180/9814700/Balance+de+Criminalidad.+Cuarto+trimestre+2019.pdf/279d1ebf-026f-4bea-8b4b-eccbab8c3430

Mozilla. (23 / març / 2019). *WebSockets.* Recollit de MDN Web Docs: https://developer.mozilla.org/es/docs/Web/API/WebSockets_API

Node.js. (10 / juliol / 2019). *Example*. Recollit de Node.js Documentation: https://nodejs.org/api/synopsis.html#synopsis_example

NTS. (29 / setembre / 2019). *¿Qué es Twilio? ¿En qué consiste su apuesta para democratizar el desarrollo de dispositivos IoT?* Recollit de NTS: https://www.nts-solutions.com/blog/twilio-que-es.php

PCE Instruments. (sense data). *Boroscopio.* Recollit de PCE Instruments: https://www.pce-instruments.com/espanol/instrumento-medida/medidor/boroscopio-kat_70030.htm

Project Management Institute, Inc. (2017). *La guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)* (Sisena ed.). Pennsylvania: Independent Publishers Group.

Pusher. (sense data). *What are WebSockets?* Recollit de Pusher: https://pusher.com/websockets

Python. (7 / juny / 2020). *asyncio - Asynchronous I/O*. Recollit de Python Standard Library: https://docs.python.org/3/library/asyncio.html

Python. (7 / juny / 2020). *multiprocessing - Process-based parallelism*. Recollit de Python Standard Library: https://docs.python.org/3/library/multiprocessing.html#contexts-and-start-methods

Python Software Corporation. (2 / juny / 2020). *multiprocessing - Process-based parallelism.* Recollit de Python: https://docs.python.org/3/library/multiprocessing.html

Raspberry Pi. (3 / abril / 2020). *Power Supply*. Recollit de Raspberry Pi: https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

Raspberry Pi Foundation. (sense data). *What is a Raspberry Pi?* Recollit de https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

Raspberry Pi. (sense data). *Getting started with the Camera Module*. Recollit de Raspberry Pi: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/1

Rodríguez, T. (31 / agost / 2012). *Entendiendo la nube: el significado de SaaS, PaaS y IaaS*. Recollit de Genbeta: https://www.genbeta.com/desarrollo/entendiendo-la-nube-el-significado-de-saas-paas-y-iaas

Rosebrock, A. (16 / setembre / 2019). *Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster*. Recollit de PyImageSearch: https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/

Rosebrock, A. (11 / març / 2019). *Liveness Detection with OpenCV*. Recollit de PyImageSearch: https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/

Rusev, K., & Gadzhev, D. (15 / maig / 2018). *What is Heroku Used For?* Recollit de MentorMate: https://mentormate.com/blog/what-is-heroku-used-for-cloud-development/

Santos, P. R. (16 / novembre / 2017). *Tipos de aprendizaje en Machine Learning: supervisado y no supervisado*. Recollit de Blog ThinkBig: https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/

Silva, S., & Freire, E. (23 / novembre / 2019). *Intro a las redes neuronales convolucionales.* Recollit de Medium: https://medium.com/@bootcampai/redes-neuronales-convolucionales-5e0ce960caf8

Single-board computer. (Maig / 2020). Recollit de Wikipedia: https://en.wikipedia.org/wiki/Single-board_computer

Techopedia. (25 / gener / 2017). *Single-Board Computer (SBC).* Recollit de Techopedia: https://www.techopedia.com/definition/9266/single-board-computer-sbc

Tena, M. (Novembre / 2018). *¿Qué es la metodología 'agile'?* Recollit de BBVA: https://www.bbva.com/es/metodologia-agile-la-revolucion-las-formas-trabajo/

Ubl, M., & Kitamura, E. (20 / octubre / 2010). *Introducing WebSockets: Bringing Sockets to the Web*. Recollit de HTML5 Rocks: https://www.html5rocks.com/en/tutorials/websockets/basics/

WebSocket. (sense data). *About HTML5 WebSocket*. Recollit de WebSocket: https://www.websocket.org/aboutwebsocket.html

Wikipedia. (13 / abril / 2020). *Bootstrap (framework).* Recollit de Wikipedia: https://ca.wikipedia.org/wiki/Bootstrap_(framework)

Wikipedia. (26 / maig / 2020). *Multi-core processor*. Recollit de Wikipedia: https://en.wikipedia.org/wiki/Multi-core_processor

Wikipedia. (29 / maig / 2020). *Node.js.* Recollit de Wikipedia: https://en.wikipedia.org/wiki/Node.js

Wikipedia. (3 / juny / 2020). *Red neuronal artificial.* Recollit de Wikipedia: https://es.wikipedia.org/wiki/Red_neuronal_artificial

Wikipedia. (31 / maig / 2020). *Sistema de reconocimiento facial*. Recollit de Wikipedia: https://es.wikipedia.org/wiki/Sistema_de_reconocimiento_facial

Wikipedia. (3 / juny / 2020). *Tiempo universal coordinado*. Recollit de Wikipedia: https://es.wikipedia.org/wiki/Tiempo_universal_coordinado

Xeredia. (16 / setembre / 2019). *Redes Neuronales artificiales: Qué son y cómo se entrenan.* Recollit de Xeridia: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

14. Annexos

Annex 1. Comparativa de models de *Raspberry Pi*

A la taula comparativa que s'exposa a continuació s'han omès els models que no disposen de port GPIO ni connectivitat Wi-Fi de fàbrica. Els preus que apareixen s'han extret d'un distribuïdor oficial de *Raspberry Pi* a Espanya.

CARACTERÍSTIQUES	Raspberry Pi 4 Model B	Raspberry Pi 3 Model A+	Raspberry Pi 3 Model B+	Raspberry Pi Zero WH	Raspberry Pi 3 Model B
PREU	37,95 € - 58,95 €	26,50 €	39,95 €	14,60 €	35,24 €
Data de llançament al mercat	juny 2019	novembre 2018	març 2018	gener 2018	febrer 2016
SoC (System on a Chip)					
Model SoC	Broadcom BCM2711	Broadcom BCM2837B0	Broadcom BCM2837B0	Broadcom BCM2835	Broadcom BCM2837
Model CPU	Cortex-A72 (ARM v8) 64-bit	Cortex-A53 64-bit	Cortex-A53 64-bit	ARM1176JZF-S	Cortex-A53 64-bit
Freqüència CPU	1,5 GHz	1,4 GHz	1,4 GHz	1 GHz	1,2 GHz
Núm. cores	4	4	4	1	4
RAM	1 GB, 2 GB, 4 GB	512 MB	1 GB	512 MB	1 GB
Connectivitat per cable					
Ports USB	2x USB 3.0 + 2x USB 2.0	1x USB 2.0	4x USB 2.0	1x Micro-USB 2.0	4 x USB 2.0
Ethernet	✓ Gigabit	×	✓ Gigabit, per USB 2.0	×	✓ 10/100 M
GPIO	✓	✓	✓	✓	✓
Càmera	✓	✓	✓	✓	✓
Connectivitat inalàmbrica					
Wi-Fi	2,4 GHz i 5GHz 802.11 b/g/n/ac	2,4 GHz i 5GHz 802.11 b/g/n/ac	2,4 GHz i 5GHz 802.11 b/g/n/ac	802.11 b/g/n	802.11 b/g/n
Dimensions					
Altura	85,6 mm	65 mm	85,6 mm	65 mm	85,6 mm
Amplada	56 mm	56,5 mm	56 mm	30 mm	56 mm
Profunditat	21 mm	11 mm	21 mm	5,4 mm	21 mm
Pes	45 g	45 g	45 g	9 g	45 g
	LOSER	WINNER	LOSER	LOSER	LOSER
Memòria RAM	No cal tanta RAM (amb menys n'hi ha suficient)	Mida de RAM correcta: no s'executaran processos que en necessitin més RAM	No cal tanta RAM (amb menys n'hi ha suficient)	Mida de RAM correcta: no s'executaran processos que en necessitin més RAM	No cal tanta RAM (amb menys n'hi ha suficient)
Ethernet	No s'utilitzarà port Ethernet	No s'utilitzarà port Ethernet	No s'utilitzarà port Ethernet	No s'utilitzarà port Ethernet	No s'utilitzarà port Ethernet
Ports USB	No es faran servir els ports USB (no fa falta 3.0 ni 4 ports USB)	No s'utilitzaran els ports USB (per tant no en fan falta més)	No es faran servir els ports USB (no fan falta 4 ports USB)	No s'utilitzaran els ports USB (per tant no en fan falta més)	No es faran servir els ports USB (no fan falta 4 ports USB)
Dimensions	Dimensió de mida gran (respecte els demés models)	Dimensió de mida mitjana (ni molt gran ni molt petita)	Dimensió de mida gran (respecte els demés models)	Dimensió de mida petita i pes molt reduït	Dimensió de mida gran (respecte els demés models)
CPU	No fa falta més freqüència de CPU (amb menys n'hi ha suficient)	Bon nombre de cores (separació core sistema de core web) i bona freqüència	Bon nombre de cores (separació core sistema de core web) i bona freqüència	Nombre de cores insuficient (no seria possible separació de cores sistema i web)	Bon nombre de cores (separació core sistema de core web) i bona freqüència
Wi-Fi	Bona connectivitat Wi-Fi (5GHz i b/g/n) per streaming de la càmera	Bona connectivitat Wi-Fi (5GHz i b/g/n) per streaming de la càmera	Bona connectivitat Wi-Fi (5GHz i b/g/n) per streaming de la càmera	Connectivitat Wi-Fi insuficient (només 2,4GHz) per streaming de la càmera	Connectivitat Wi-Fi insuficient (només 2,4GHz) per streaming de la càmera
Preu	Preu elevat d'acord amb el que realment es necessita	Bon preu d'acord amb les prestacions i necessitats	Preu lleugerament superior d'acord amb les necessitats	Bon preu d'acord amb les prestacions	Altres models més nous tenen una millor relació preu/prestacions

Annex 2. *Convolutional Neural Network*

Xarxes neuronals

Les xarxes neuronals són models computacionals que es basen en el funcionament de les xarxes neuronals biològiques, i estan compostos per neurones (nodes) connectades entre sí (arestes), que s'utilitzen per emmagatzemar coneixement i que treballen en conjunt (Wikipedia, 2020).

L'estructura d'una neurona artificial és la següent (vegeu figura 56):

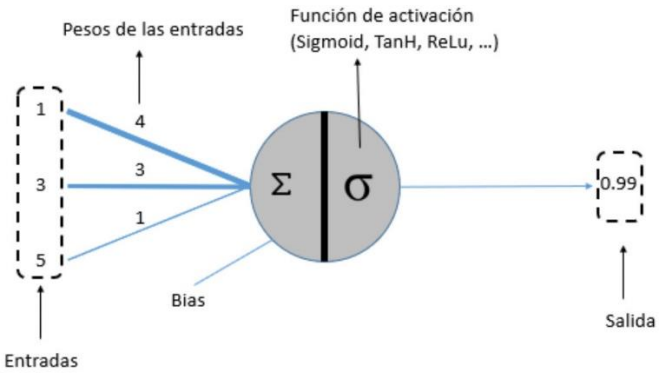


Figura 56. *Representació esquemàtica d'una neurona artificial.* Extret de *Xeridia*. Font: https://www.xeridia.com/sites/default/files/contenidos/blog/neurona_artificial.jpg

La suma de les entrades multiplicat pel respectiu pes de cadascuna de les entrades determina l'impuls que rep la neurona, on el pes és el factor que determina l'excitació o inhibició de les connexions (és a dir, de les entrades). A partir de la funció d'activació i del valor d'entrada es calcula el valor que retorna la neurona (Xeredia, 2019).

A les xarxes neuronals artificials, aquests nodes solen estar agrupats en diferents nivells anomenats capes. Aquestes capes són agrupacions de neurones que tenen com entrada les sortides de la capa anterior i la sortida és l'entrada de la capa posterior (vegeu figura 57 per visualitzar l'exemple d'una xarxa neuronal).

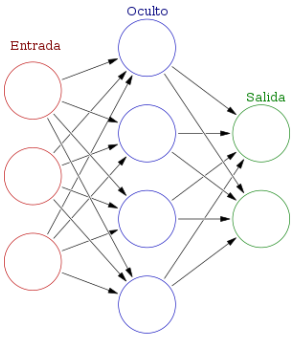


Figura 57. *Exemple d'una xarxa neuronal amb capa d'entrada (en vermell), capa oculta (en blau) i capa de sortida (en verd).* Extret de *Wikipedia*. Font: https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Colored_neural_network_es.svg/300px-Colored_neural_network_es.svg.png

D'acord a la forma com les capes perceben les dades, es classifiquen en tres tipus: d'entrada, que és la capa que rep les dades reals (de l'exterior) que alimenten a la xarxa neuronal; ocultes, que no tenen connexió amb l'exterior, estan situades entre capes d'entrada i de sortida, i els valors d'entrada i de sortida es desconeixen; i finalment de sortida, que són el resultat visible de la xarxa, que transmet dades a l'exterior.

Les xarxes neuronals sempre estan formades per una capa d'entrada i una capa de sortida, i addicionalment de capes ocultes. El concepte *deep learning* sorgeix d'utilitzar xarxes neuronals amb una gran quantitat de capes ocultes.

L'entrenament de les xarxes neuronals consisteix en regular els pesos de les entrades de totes les neurones de manera que els resultats calculats per la capa de sortida s'ajustin el millor possible a les dades conegudes, i per reduir l'error. Cada modificació dels pesos comporta guany d'experiència, i quelcom après "queda" a la xarxa neuronal.

Existeixen tres tipus d'entrenaments que s'apliquen a les xarxes neuronals (Franco, 2008):

- **Predictiu o supervisat:** es coneixen les sortides desitjades pel conjunt d'entrenament usat.
- **Descriptiu o no supervisat:** no es coneixen les sortides desitjades pel conjunt d'entrenament utilitzat. La xarxa serà la que busqui el seu comportament més adequat segons un criteri.
- **Per reforç:** la xarxa aprendrà amb exemples i es corregiran els pesos a través d'un senyal avaluatiu, la qual serà positiva si millora el comportament de la xarxa i serà negativa en cas contrari.

El principal avantatge d'aquests models és la d'evitar desenvolupar un programa i haver d'especificar manualment les regles que determinen els valors de sortida, ja que les xarxes neuronals ajusten els seus paràmetres automàticament per identificar les característiques de les dades que circulen a través d'elles.

Xarxes neuronals convolucionals

Les xarxes neuronals convolucionals són una branca de les xarxes neuronals que estan orientades al tractament i processament d'imatges, que és el que rep com entrada. Les xarxes convolucionals consisteixen en múltiples capes de filtres convolucionals d'una o més dimensions, i després de cada capa se sol afegir una funció de mapeig casual no lineal. Les tasques d'aquestes xarxes són: detectar i/o categoritzar objectes, classificar escenes i classificar imatges en general (Silva & Freire, 2019).

La fase d'extracció de característiques està composta per capes alternes de neurones convolucionals, seguides per reducció per mostreig i al final es troben neurones més senzilles per realitzar la classificació final a partir de les característiques extretes (Barrios).

Com entrades, s'agafen els píxels de la imatge a processar, per tant si una imatge en escala de grisos té una resolució de 1280x720 píxels, la xarxa neuronal haurà de tenir un total de 921.600 neurones d'entrada. En canvi, si fos d'una imatge a color, serien necessaris tres canals per cada color primari (vermell, verd i blau), i el nombre de neurones d'entrada passaria a ser 1280x720x3 = 2.764.800.

Per extreure determinades característiques importants o patrons d'una imatge, les xarxes convolucionals realitzen convulsions, que consisteix en prendre un conjunt de píxels propers de la imatge d'entrada i calcular el producte escalar contra una matriu anomenada *kernel*. A continuació es pot veure un exemple del producte escalar que es calcula a partir d'un *kernel* (vegeu figura 58):

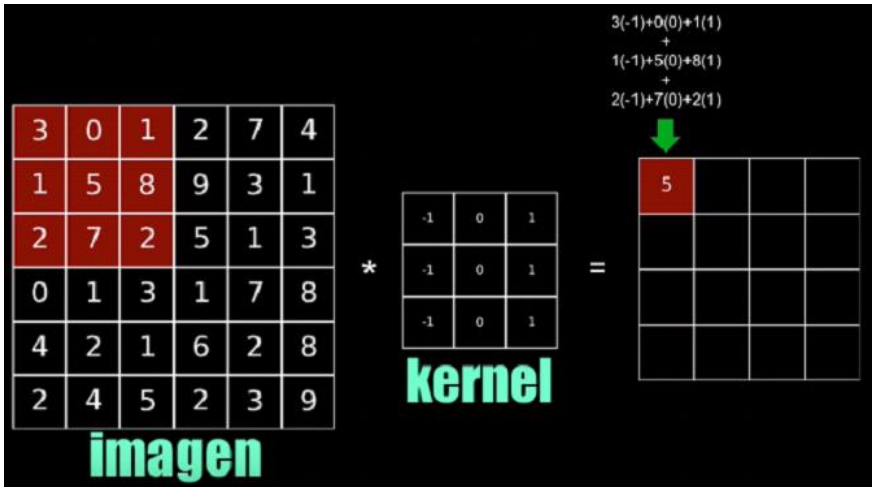


Figura 58. *Exemple de càlcul del producte escalar pel primer conjunt de píxels 3x3.* Extret de *Bootcamp AI*. Font: https://miro.medium.com/max/1400/1*67NJ1OXILVMUx-tCqaWUZg.png

El conjunt de píxels seleccionat es va desplaçant d'esquerra a dreta (primer columnes) i de dalt a baix (després files) per seguir calculant el producte escalar i omplir la matriu resultant d'aplicar el *kernel*. Els valors que formen part de la matriu *kernel* s'inicialitzen de forma aleatòria, i s'aniran ajustant de forma automàtica durant l'entrenament.

No obstant, en una xarxa convolucionals s'aplicaran diversos *kernel* (un conjunt de *kernel* s'anomena filtre), i la matriu resultant d'aplicar aquests filtres s'anomena *feature mapping*. Aquesta matriu serà la que determini característiques concretes de la imatge original, i ajudarà a la xarxa neuronal a distingir un objecte d'un altre.

Després d'aplicar una convolució amb un *kernel* s'aplica la funció d'activació ReLu (*Rectifier Linear Unit*), que en aquest cas és $f(x) = \max(0, x)$.

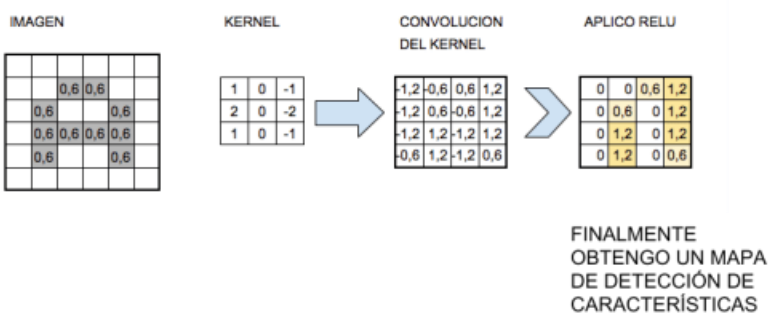


Figura 59. *Flux d'operacions per obtenir un mapa de característiques.* Extret de *Juan Barrios*. Font: https://i1.wp.com/www.aprendemachinelearning.com/wp-content/uploads/2018/11/CNN-04.png?resize=768%2C355

Seguidament es pren una mostra de les neurones més representatives abans de tornar a fer una convolució, per tal de reduir la mida de la pròxima capa de neurones, ja que si apliquem una convolució amb 32 filtres sobre una imatge de 28x28 s'obtindrà una capa oculta de 25.088 nodes. La tècnica que s'utilitza s'anomena *max-pooling*, amb la qual es redueix proporcionalment el mapa de característiques agafant els valors més alts.

Amb una primera convolució, la xarxa seria capaç de detectar elements primitius com línies o corbes. A mesura que es vagin creant més capes a base de convolucionar, els *feature maps* seran capaços de reconèixer formes més complexes.

Finalment, després d'aplicar convolucions successives es connecta l'última capa oculta amb una xarxa neuronal tradicional.

Annex 3. *Twilio*

Twilio és una plataforma de desenvolupament que ofereix serveis de missatgeria de veu, videotrucades, missatges de text i autenticació. La plataforma ha aconseguir virtualitzar la infraestructura global de telecomunicacions de forma transparent, accessible des d'una API que permet que tots els serveis que ofereix siguin accessibles als programadors que desenvolupen aplicacions (Chetu, 2018).

Abans de que existís aquesta plataforma, els dispositius d'Internet de les Coses (IoT) havien de disposar d'interfícies o mòduls que connectessin el dispositius amb la xarxa mòbil, de manera que fos possible pel dispositiu realitzar trucades o enviar SMS (NTS, 2019). Amb l'arribada d'aquesta plataforma ja no és necessari, sinó que creant un compte a la plataforma i configurant les dades de facturació ja es pot començar a utilitzar de forma molt fàcil i intuïtiva.

Els serveis que ofereix la plataforma són els següents (Gómez, 2017):

- **Veu i vídeo**
    - Phone to Phone: control, realització i recepció de trucades telefòniques en tot el món.
    - App to Phone: realització de trucades des d'una aplicació a telèfons mòbils o fixes.
    - App to App: establiment d'una comunicació *punt a punt* per trucades i videotrucades HD, orientada a aplicacions mòbils i webs.
    - Many to Many: integració per aplicacions de conferències d'àudio amb controls de moderador.
- **Missatgeria**
    - Text Messaging (SMS): enviament i recepció de missatges de text a través de la xarxa d'un operador mòbil, a qualsevol part del món.
    - Picture Messaging (MMS): intercanvis de missatges d'imatge (MMS) entre telèfons mòbils a través de la xarxa d'un operador mòbil.
    - In-app Chat: possibilitat de xat en temps real a través d'aplicacions mòbils i/o web.
    - Notifications: enviament d'alertes i notificacions en diversos canals.

Twilio utilitza un llenguatge de marcat propi basat en XML, que s'anomena *TwiML* i està compost per una sèrie d'instruccions que permeten informar al sistema sobre com processar trucades o SMS (entrants i sortints). A continuació es pot observar un exemple d'un *TwiML* que dictarà per veu el text "Hello Kabel" (vegeu figura 60):

```
<?xml version="1.0" encoding="UTF-8" ?>
    <Response>
        <Say> Hello Kabel <Say/>
    </Response>
```

Figura 60. *Exemple d'un TwiML que determina que la resposta haurà de ser el dictat per veu de "Hello Kabel".*

Extret de *Kabel*. Font: https://www.kabel.es/wp-content/uploads/2017/03/ejemplo_TwiML3.jpg

El format que ha de seguir aquest conjunt d'instruccions utilitzat per *Twilio* ha de contenir estrictament l'etiqueta que especifica la versió d'XML i la codificació, i totes les accions han d'anar contingudes dins de l'etiqueta "*<Response>*". Les accions que poden utilitzar-se per a trucades de veu són les següents:

- ***<Say>*:** llegir un text.
- ***<Play>*:** reproduir un fitxer d'àudio.
- ***<Dial>*:** marcar un número de telèfon per incloure'l a la trucada.
- ***<Record>:*** enregistrar la veu de la persona implicada en la trucada.
- ***<Gather>:*** capturar els dígits marcats al teclat de la persona implicada.

I per controlar el flux de la trucada hi ha les següents etiquetes:

- ***<Hangup>***: penjar la trucada.
- ***<Enqueue>***: posar la trucada a la cua.
- ***<Leave>***: treure la trucada de la cua.
- ***<Pause>***: espera.
- ***<Redirect>***: redirecció de la trucada a un altre document *TwiML*.
- ***<Reject>***: rebutjar una trucada entrant sense que es facturi.

A continuació es presenta un exemple en *Python* que utilitza *Twilio* per realitzar una trucada sortint, especificant, amb el paràmetre "url", les accions que s'hauran d'executar a l'establir la trucada:

```
1. from twilio.rest import Client
2.
3. account_sid = 'ACXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
4. auth_token = 'your_auth_token'
5. client = Client(account_sid, auth_token)
6.
7. call = client.calls.create(
8.     url='http://demo.twilio.com/docs/voice.xml',
9.     to='+15558675310',
10.    from_='+15017122661'
11.)
12.
13.print(call.sid)
```

Les accions que s'executaran a la trucada realitzada seran les següents:

```
1. <Response>
2.     <!-- Dictar per veu el text a continuació -->
3.     <Say voice="alice">Thanks for trying our documentation. Enjoy!</Say>
4.     <!-- Reproduir àudio -->
5.     <Play>http://demo.twilio.com/docs/classic.mp3</Play>
6. </Response>
```

Com pot apreciar-se, aquesta plataforma seria ideal per implementar en aquest projecte, fet que ja s'ha mencionat a l'apartat de futures millores. D'aquesta manera, podria simular-se la trucada tradicional que fa l'empresa de l'alarma al client, informant de que s'ha produït una intrusió.