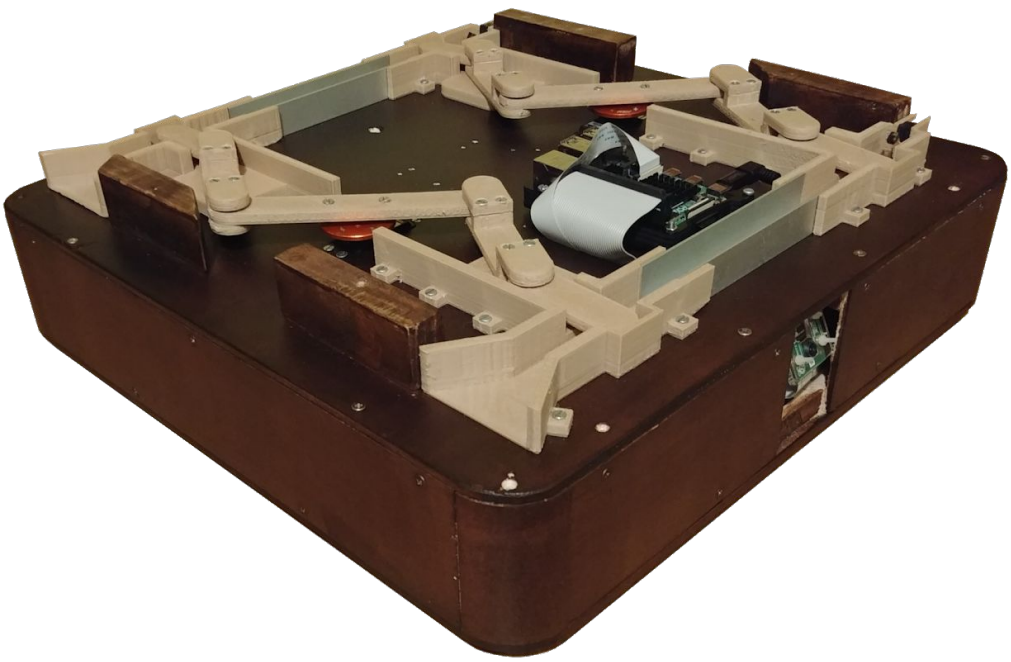
# ROOBY, EL MILLOR AMIC PER LA LOGÍSTICA

*Disseny, construcció i programació d'un robot de baix cost*



Aida Benítez Bagué
Tutora: Maria Masoliver
2n Batxillerat D
Departament de Tecnologia
09/11/2020

# RESUM

Durant els últims anys, els avenços tecnològics han aportat grans millores en el món de la robòtica. Actualment, la creació i programació de tants tipus de robots ens està obrint les portes cap a un nou futur. Grans empreses fan ús de robots logístics per tal de facilitar la seva feina. Els enginyers que construeixen aquests robots els hi donen capacitats humanes com la vista, la interpretació d'imatges i el moviment autònom. Però ho fan a canvi de grans beneficis. Per aquest motiu, és possible, construir i programar un robot autònom de baix cost per totes aquelles petites empreses?

Al llarg d'aquest treball s'investigarà sobre la robòtica de logística i la visió artificial amb l'objectiu de dissenyar, construir i programar un robot de baix cost capaç de moure's autònomament. Serà possible aconseguir-ho?

# ABSTRACT

In recent years, technological advances have brought great improvements in the robotics world. Currently, the creation and programming of so many types of robots is leading us to a new future. Big enterprises use logistics robots to make their work easier. The engineers who build these robots give them human abilities such as sight, image interpretation and autonomous movement. But they do it in exchange for big profits. For this reason, is it possible to build and program a low-cost autonomous robot for all those small businesses?

Throughout this work, research will be done on logistics robotics and artificial vision with the aim of designing, building, and programming a low-cost robot capable of moving autonomously. Will it be possible to achieve it?

# AGRAÏMENTS

Vull agrair la col·laboració de totes aquelles persones que han fet possible la realització d'aquest projecte.

En primer lloc, a la meva tutora Maria Masoliver per haver confiat en mi des del dia que li vaig presentar la idea del projecte. Durant aquests mesos m'ha recolzat en totes les decisions preses i m'ha ajudat a poder-les dur a terme.

En segon lloc, m'agradaria donar les gràcies al meu mentor de robòtica, Marc Camacho, per haver-me proposat aquest projecte i endinsar-me al món de la robòtica. A més, m'ha brindat coneixements claus tant en l'àmbit de la robòtica com sobretot en el de la programació que m'han ajudat en la realització el projecte.

Agrair també, l'ajuda d'en Joan Font, company de robòtica i amic, per dedicar-me part del seu temps, brindar-me els seus coneixements en Solidworks i ajudar-me en la realització de totes les peces 3D.

També, vull agrair al meu veí, Josep Batchelli, per deixar-me treballar al seu taller i ajudar-me en la part mecànica del robot. Per ensenyar-me a utilitzar tot tipus d'eines i donar-me moltes idees que m'han ajudat a avançar quan estava més encallada.

Finalment, m'agradaria acabar els agraïments donant les gràcies a la meva família. En primer lloc, a la meva mare Mònica Bagué i al meu germà Joel Benítez per viure amb intensitat el procés de realització del projecte i animar-me quan les coses no sortien bé. Però, especialment, donar les gràcies al meu pare, Quim Benítez, per ser el suport incondicional d'aquest projecte. Gràcies a la seva dedicació, il·lusió i paciència, ha estat possible aquest resultat.

# ÍNDEX DE CONTINGUTS

# ÍNDEX DE FIGURES

# ÍNDEX DE TAULES

# 1. INTRODUCCIÓ

Des de l'inici de la història les persones hem necessitat la tecnologia. Tant és així, que constantment buscàvem com millorar-la. Al principi eren el foc, la roda i altres invents que facilitaven la vida a les persones i any rere any, aquesta ha anat adquirint importància fins a arribar a l'actualitat. Avui en dia, s'han potenciat molt les noves tecnologies, i les tenim tant presents que no sabríem sobreviure sense elles. D'entre tots els avenços que ens han aportat al llarg dels anys, la robòtica és un dels més importants. Convivim amb ella tant a casa, com a la feina ja que ens pot substituir i fer aquelles tasques difícils que ens costa realitzar estalviant-nos així, molt de temps. Cada vegada podem veure més aplicacions de sistemes robotitzats en els àmbits que van des de la salut, la logística o, fins i tot, l'oci.

Amb la Revolució Industrial, totes les fàbriques es van mecanitzar aconseguint una major eficiència i creant més productes amb menys temps. D'aquesta manera, retallaven costos de producció i s'adaptaven a una societat que volia augmentar el consum. En l'actualitat, un dels reptes que hi ha, ja no és només produir molt, sinó que la societat demana accedir als productes d'una manera molt més ràpida. D'aquesta manera, grans empreses com Amazon s'han hagut d'adaptar a aquest canvi millorant el seu sistema de logística. Per fer-ho han aprofitat l'avenç de la robòtica creant magatzems on els robots porten les estanteries fins on es troba el treballador. A més, per facilitar-li la feina, la resta de tasques també estan automatitzades de tal manera que la zona on es troba aquest, hi ha tot de màquines que li diuen quin tipus de caixa ha d'utilitzar, com l'ha de tancar i quins objectes hi ha de posar. Tots els paquets es mouen a través de cintes que els transporten d'un treballador a un altre augmentant així la velocitat del procés. Tant és així que avui en dia són capaços de fer arribar un producte a casa teva en tan sols dues hores. Aquesta robòtica de logística permet a les empreses realitzar els processos d'una manera més senzilla i còmoda a més d'aconseguir fer tots

els trasllats de manera automàtica. També s'estalvien temps, diners i esforços, augmentant així la productivitat i la flexibilitat del magatzem.

Aquesta innovació és molt útil per grans empreses que tenen molts diners, però, quan parlem de les més petites, aquestes no tenen suficient pressupost per poder aconseguir aquests robots i, per tant, no poden millorar la logística dels seus magatzems. És aquí quan no només són necessaris aquests robots, sinó que a més, es necessiten uns que siguin de baix cost. Alguns exemples d'altres àmbits que no poden aconseguir-los són els següents:

- En l'àmbit de l'educació, tots els professors han necessitat alguna vegada sortir de la classe per anar a buscar aquell paper que han imprès o bé, han enviat a un alumne a fer-ho. A més, a cada canvi d'hora han de portar els llibres i les carpetes necessàries per la següent classe carregant així molt de pes durant moltes hores del dia. Així, amb un robot de logística podrien transportar tot aquell material necessari d'una classe a una altra sense fer-se mal a l'esquena ni perdre temps durant les classes.

- En l'àmbit de la salut, les infermeres mouen els carros a través dels passadissos transportant els medicaments i altres instruments que necessiten per poder avaluar l'estat dels pacients o fins i tot els diferents àpats per cada pacient. Amb un robot de logística podrien fer que aquest carro anés d'habitació en habitació ja sigui seguint a l'infermer o pel seu compte, facilitant-los així la seva feina.

- En el comerç local, actualment hi ha grans superfícies comercials que malgrat no tenir la necessitat de la immediatesa d'Amazon, tenen les seves limitacions. El manteniment del seu magatzem és primordial i necessiten persones que hagin de fer tota aquesta feina que, en una empresa multinacional ho faria un robot. Per aquest motiu, amb un robot de logística, podrien realitzar tots aquests processos d'una manera més ràpida i menys costosa.

Per aquest motiu he decidit centrar-me en aquests àmbits on no poden accedir, degut al pressupost, als robots i mecanismes de les grans empreses com Amazon. Però en canvi, també els hi seria molt útil aconseguir un robot de logística per poder facilitar la feina a aquells que han de traginar dia rere dia pesos que es podrien estalviar. Així doncs, vull aconseguir dissenyar, muntar i programar un robot de baix cost capaç de transportar material d'un costat a l'altre de manera autònoma.

## 1.1. Motivacions

Després de realitzar durant quatre anys la coescolar de robòtica a l'escola FEDAC Sant Narcís i d'estar diàriament en contacte amb robots, aquest projecte em permetrà saber si realment aquest és l'àmbit que m'agrada i al que em vull dedicar. D'aquesta manera comprovaré si puc superar el repte d'adquirir coneixements sobre programació, visió per computador, mecànica i electrònica de manera autònoma i aplicar-los en la construcció d'un robot. A més, podré resoldre el problema que se'm va plantejar a la mateixa escola que és la dificultat de portar objectes d'un costat a un altre aplicable també en altres sectors com el de la sanitat i el comerç.

## 1.2. Hipòtesis i objectius

La meva hipòtesis inicial del projecte és:

> "Pot una estudiant de batxillerat construir un robot de baix cost que es desplaci autònomament mitjançant la visió per computador?"

Per tal d'aconseguir-ho caldrà superar els següents objectius:

1. Construir un robot que pugui controlar el seu moviment per odometria.
2. Programar un robot perquè sigui capaç de desplaçar-se autònomament mitjançant uns punts de guia.

3. Programar un robot perquè sigui capaç de detectar una persona i seguir-la mitjançant la visió per computació.

## 1.3. Com ho he investigat

Per tal d'arribar a fer aquest projecte es van seguir les següents pautes en el procés d'aprenentatge i d'investigació:

- Aprenentatge sobre què és i els usos de la Raspberry Pi.
- Curs en línia de la Cisco Network Academy sobre Linux per a moure'm amb agilitat a través de la Raspberry Pi.
- Consulta a enginyers i autoformació per a escollir els millors components pel robot.
- Curs de OpenCV i Raspberry per a la detecció i seguiment d'objectes a través de les imatges d'una càmera.

A partir d'aquí, en el marc pràctic, es va buscar gent de referència per obtenir consells sobre com progressar.

## 1.4. Planificació

La planificació és una eina fonamental per tal de poder assolir tots els objectius d'un projecte. Permet agrupar les tasques similars i ubicar-les en el temps per tal d'obtenir una major efectivitat.

El primer que cal tenir present per a realitzar una bona planificació són les dates claus del projecte (veure **Figura 1**). Una vegada es tenen clares, es pot iniciar el procés.

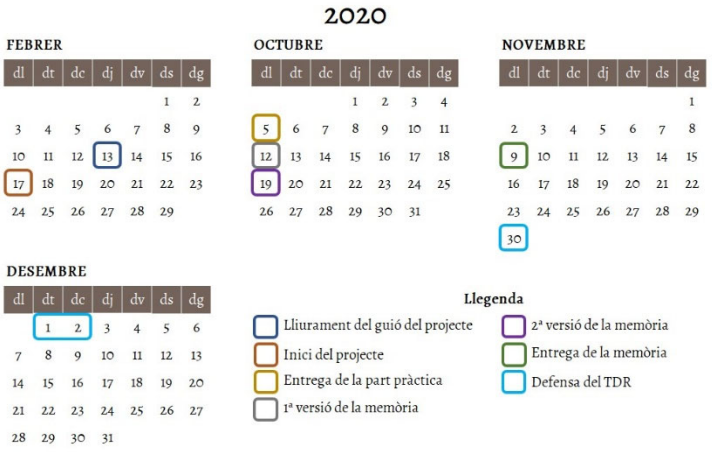



***Figura 1.*** *Calendari amb les dates a tenir en compte pel projecte. Elaboració pròpia*

Per la planificació del treball es va optar per utilitzar el diagrama de Gantt ja que et permet distribuir totes les tasques d'una manera més clara. A la **Figura 2** es poden veure les diferents tasques repartides en el temps (de febrer a novembre) i classificades en quatre grups diferents: aprenentatge, robot, programació i memòria i defensa. Com es pot observar, hi ha tasques fetes paral·lelament gràcies a la planificació. A continuació s'expliquen cadascuna de les etapes.

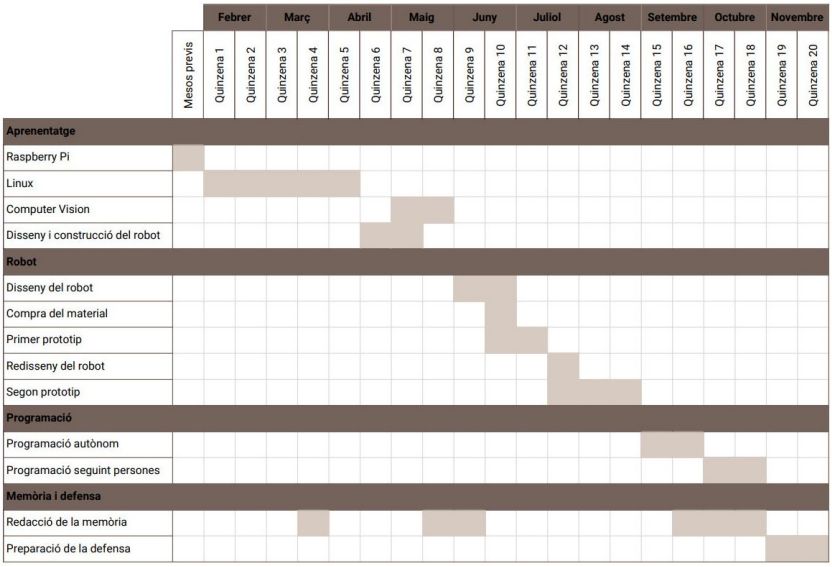



***Figura 2.*** *Temporització, per etapes, en un diagrama de Gantt. Elaboració pròpia*

La primera etapa de totes és la de l'aprenentatge. Al llarg d'aquest període de temps es farà recerca i es realitzaran diferents cursos per tal d'adquirir els coneixements

necessaris i poder continuar amb el treball. Aquest serà un període clau del projecte perquè d'ell depèn tota la part pràctica.

Tot seguit apareix l'etapa del robot. Dins d'ella hi trobem els processos de disseny i de construcció del robot. Aquesta vindria a ser la part mecànica i electrònica del treball i s'hi hauran de prendre totes les decisions necessàries per poder obtenir com a resultat final el robot.

Conseqüentment ve l'etapa de la programació. Una vegada s'hagi acabat de muntar el robot, s'haurà de fer funcionar. Durant aquest període es durà a terme la realització de tots els codis necessaris per aconseguir assolir els objectius proposats a l'inici del treball

Finalment hi ha l'etapa de memòria i defensa on es prepararan les entregues del treball tant escrites com orals. Aquest procés es durà a terme des de l'inici del treball fins al final. En les darreres setmanes serà supervisat pel tutor i pel propi projectista. Finalment, després d'haver entregat el treball definitiu, es prepararà la defensa del treball.

# 2. MARC TEÒRIC

En el marc teòric es van estudiar tots els conceptes previs per la realització d'aquest treball. En primer lloc, sabent que es treballaria amb Raspberry Pi i que el seu sistema operatiu és el Linux, calia entendre el seu funcionament. En segon lloc, va caldre profunditzar en un camp que, malgrat no ser nou per mi, era necessari consolidar. Per últim era necessari comprendre què era la Visió per Computador i endinsar-se en les bases per tal de poder aplicar els coneixements en el marc pràctic per aconseguir que el robot seguís línies i persones d'una manera autònoma.

Així doncs, a continuació es troba un resum de tots els continguts d'aprenentatge teòric treballats.

## 2.1. Introducció al Linux

La informàtica va sorgir al llarg de la dècada dels 40 després de la II Guerra Mundial. Al principi, els programadors interactuaven directament amb el hardware treballant en llenguatge binari. En aquell temps es desconeixia el concepte de sistema operatiu fins que, l'any 1956 es va crear el primer per a un ordinador IBM 704 (veure **Figura 3**). Aquest executava únicament els programes en funció de quan acabaven els anteriors. Va ser als anys 60 quan va haver-hi una revolució dels sistemes operatius i va aparèixer UNIX, la base de la gran majoria de sistemes operatius d'avui en dia.

***Figura 3.*** *IBM 4, primer ordinador amb sistema operatiu. (Sulbaran, 2012)*

Actualment, la base de molts sistemes operatius a passat a ser Linux. Aquest va ser inspirat en UNIX i creat per Linus Torvalds l'any 1991. En altres paraules, és el nucli o kernel en anglès. Es va dissenyar per cobrir la necessitat que tenia el món de la tecnologia

de comunicar de manera segura el programari, anomenat software, i la maquinària, conegut també per hardware. Per fer-ho, el sistema ha de poder gestionar el disc dur, on s'emmagatzema tota la informació; la memòria RAM, on s'executen diferents aplicacions al mateix temps; i el processador o CPU, que s'encarrega de realitzar les diferents operacions. A més, ha de permetre la comunicació entre aquests tres dispositius i tots els perifèrics que connectem a la placa base com en són el ratolí, el teclat o la pantalla.

La importància que té aquesta gestió és molt gran per poder realitzar les ordres de manera correcta. El motiu és que si tenim aquesta comunicació segura, les ordres que s'enviaran de la memòria RAM al disc dur, o de la RAM al processador, seran les que realment volem executar i obtindrem els resultats esperats. En canvi, si la comunicació entre aquests elements no és bona, podem alterar el procés que estem intentant realitzar obtenint resultats contraris als que esperàvem.

Si no tenim una comunicació segura podem assegurar que les aplicacions fallaran. Per aquest motiu aquestes han de seguir diferents protocols anomenats API (Application Programming Interface) per poder-se comunicar amb el sistema operatiu. Perquè el kernel pugui fer ús de les diferents aplicacions realitza processos, és a dir, execucions de les dades i instruccions. Una mateixa aplicació pot necessitar múltiples processos per funcionar i és el kernel qui s'encarrega d'executar-los engegant-los i parant-los segons el que se li ha demanat.

Linux és un sistema operatiu de software lliure, és a dir, que no és propietat de cap persona o empresa. Al tenir un codi obert, el seu codi font és accessible per a qualsevol persona i aquesta el pot modificar. Després de la seva primera versió, el sistema va ser modificat per milers de programadors sota la coordinació del seu creador. Està basat principalment en el sistema UNIX tot i que el projecte rep el nom de GNU ja que el sistema es distribueix sota la llicència GNU GPL (General Public License). Les distribucions s'encarreguen de configurar l'emmagatzematge d'informació i d'instal·lar

el kernel juntament amb la resta del software. Igual que UNIX, Linux consta de diferents centenars de distribucions de les quals destaquen Ubuntu, Debian, Red Hat, Open SUSE, Scientific i Linux Mint.

### 2.1.1. El seu funcionament

Durant els primers anys de les computadores, el control de l'execució dels programes es feia en llocs especialitzats on l'administrador manipulava directament la màquina. S'escrivia una ordre en el teclat i es rebia la resposta a través d'un full imprès. Va ser així com es va començar a interactuar amb les computadores, amb les línies de text simples. Més endavant, als inicis de la dècada de 1970, Unix va popularitzar l'ús de la línia de comandes creant així les interfícies de text. Quan es va popularitzar la computadora personal a la dècada de 1980, les aplicacions van començar a prendre relleu i gràcies a Alan Kay i al seu equip, van aparèixer les interfícies gràfiques sent, des de llavors, les interfícies dominants.

La CLI (Command Line Interface) es va començar a popularitzar l'any 1970 permetent als usuaris escriure comandes a una consola per comunicar-se amb un sistema operatiu. Va ser molt útil perquè va permetre una programació més ràpida i elaborada. Aquesta interfície, tot i que va ser una gran invenció i actualment encara destaca per la velocitat amb què s'escriuen les comandes, requereix un gran coneixement de totes les opcions de comandes que hi ha. A la **Figura 4** es pot veure un exemple de CLI.

En canvi, la GUI (Graphical User Interface), utilitza gràfiques per permetre als usuaris intercomunicar-se amb el sistema operatiu o l'aplicació. Proporciona entre d'altres finestres, botons i icones per facilitar l'ús als usuaris (veure **Figura 5**). És per aquest motiu que quan va aparèixer aquesta interfície a la dècada de 1980 es va convertir en la dominant. El motiu era senzill, tothom entén millor un botó que una comanda, perquè és molt més senzill interpretar una icona que llegir un codi. És molt còmode per aquells

que l'utilitzen però no tant per aquells que la dissenyen ja que es tracta d'un procés llarg i lent.

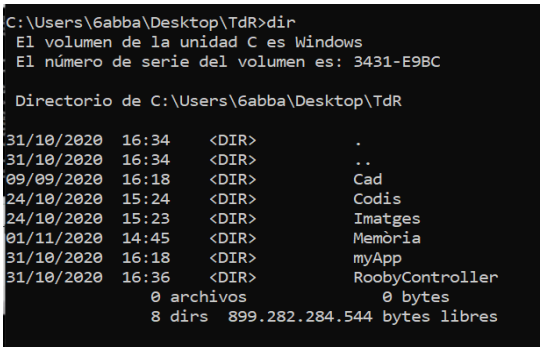



***Figura 4.*** *Exemple de CLI (Command Line Interface). Elaboració pròpia*

***Figura 5.*** *Exemple de GUI (Graphical User Interface). (Linux, 2013)*

Per la seva senzillesa d'ús, actualment s'aconsella a tots els principiants utilitzar la GUI perquè té una comprensió molt senzilla. De fet, és el que estem acostumats a veure tant en els nostres ordinadors, com en els mòbils. Pràcticament tothom, en el seu dia a dia, fa servir entorns gràfics amb finestres, pestanyes i aplicacions preparades per això.

A la **Taula 1** apareix un resum de les diferències més notables entre el CLI i la GUI.

**Taula 1** - Diferències entre CLI i GUI

| | **CLI** | **GUI** |
|---|---|---|
| **Dispositiu utilitzat** | Teclat | Ratolí, teclat i botons |
| **Ús** | És difícil realitzar una operació i requereix experiència | Feines fàcils de realitzar i no requereix experiència |
| **Consum memòria** | Baix | Alt |
| **Aparença** | No es pot canviar | Es poden fer canvis personalitzats |
| **Velocitat** | Ràpida | Lenta |

Font: Adaptat de (Solución, 2018)

Després de comparar les dos interfícies, la millor per a realitzar aquest treball seria la CLI per dos motius:

- Malgrat que la dificultat de treball és major, treballar per CLI permet connectar-se a la Raspberry per saber què està passant en cada moment. Aquesta connexió però, no es pot establir a través d'un cable perquè el robot es mourà contínuament. Així doncs, la tecnologia que s'utilitzarà per connectar-se a la Raspberry serà la WiFi.
- Com que l'accés serà remot des de l'ordinador i a través de comandes, no cal que la Raspberry contingui GUI. D'aquesta manera, també tindré més espai per poder guardar-hi altres programes o aplicacions més necessaris.

Una vegada decidida la interfície, cal conèixer alguns dels apartats claus dels sistemes operatius, en concret els de les distribucions de Linux, que em serviran per fer funcionar el robot.

### 2.1.2. Comandes bàsiques

En l'entorn de treball de Linux, les comandes són unes eines essencials. Aquestes són petits programes que s'envien a l'ordinador i que estan incorporades en el sistema operatiu. A més, són accessibles directament des de la consola sense conèixer-ne la localització i permeten fer les tasques més comunes de gestió. A la **Taula 2**, es llisten les que són més habituals juntament amb una petita descripció.

**Taula 2** - Comandes bàsiques de Linux

| Comanda | Definició | Comanda | Definició |
|---|---|---|---|
| `cd` | Es mou al directori principal | `ifconfig` | Informa sobre la xarxa |
| `cd..` | Es mou al directori superior | `iwconfig` | Informa sobre la xarxa sense fil |
| `ls` | Enllista arxius del directori actual | `ssh` | Obre sessió sistema remot |
| `ls -a` | Enllista arxius (inclós els ocults) | `route -n` | Enllista ruta |
| `touch` | Crear arxiu | `passwd` | Canviar contrasenya usuari |
| `mkdir` | Crear directori | `visudo` | Edita arxiu de configuració |

| | | | |
|---|---|---|---|
| `cp` | Copia arxiu o directori al destí | `sudo` | Executa comanda sense permís |
| `mv` | Mou un arxiu | `su -` | Canvia a la conta de root |
| `rm` | Elimina un arxiu | `chown` | Defineix usuari i grup |
| `tar` | Comprimeix | `chmod` | Defineix permisos |

Font: Elaboració pròpia.

### 2.1.3. Permisos i drets

Tots aquells que tenim ordinador, en algun moment hem tingut la necessitat de limitar l'accés de documents o carpetes per evitar que certes persones vegin, eliminin o inclús modifiquin el contingut dels arxius. Això mateix passa amb el sistema operatiu Linux. Aquest, com que és un sistema dissenyat fonamentalment pel treball en xarxa, la seguretat de la informació és essencial ja que molts usuaris podran tenir accés als recursos tant de software com de hardware que hi ha gestionats als ordinadors. Per aquest motiu, Linux consta d'un sistema de permisos que evita aquesta falta de protecció.

Els permisos que poden tenir els usuaris es classifiquen en tres nivells diferents, els permisos del propietari (`u`), els permisos del grup (`g`) i els permisos de la resta d'usuaris també anomenats "els altres" (`o`). En aquests casos, és l'administrador qui s'encarrega de crear i donar de baixa als usuaris. Però, per poder establir permisos, primer s'han de saber diferenciar els tots tipus d'arxius que pot tenir un sistema.

Es poden tenir permisos per dos àmbits diferents, un per arxius i l'altre, per directoris o carpetes. En ambdós casos, els permisos només varien en el sentit que, tal i com el seu nom indica, un serà exclusiu per arxius i l'altre, en canvi, serà per directoris.

Cada arxiu és identificat per 10 caràcters anomenats "màscara". D'entre aquests caràcters, el primer sempre fa referència al tipus d'arxiu, ja sigui un directori (`d`), un arxiu de blocs especials (`b`), un arxiu de caràcters especials (`c`), un arxiu d'enllaç (`l`), o bé un arxiu sense determinar (-). Els següents 9 nombres que venen a continuació es

divideixen en blocs de tres fent referència als permisos que es concedeixen. Aquests segueixen aquest ordre: propietari, grup i altres. Els caràcters que defineixen aquests permisos són: `r`, pel permís de lectura; `w`, pel permís d'escriptura; `x`, pel permís d'execució o bé el símbol "-", per identificar que no hi ha cap permís.

Podem agafar com exemple un arxiu anomenat "CSS_Avançat.pdf" que apareix de la manera següent: `-rw-r--r--`... CSS Avançat.pdf. Aquest mateix arxiu analitzat tindria una forma similar al resum de la **Taula 3**.

**Taula 3 -** Anàlisis de l'arxiu "OpenCV.pdf"

| Tipus | Usuari | Grup | Altres | Nom de l'arxiu |
|---|---|---|---|---|
| - | rw- | r-- | r-- | OpenCV.pdf |

Font: Recuperat de (Perseo, 2011)

En altres paraules, tothom pot visualitzar l'arxiu però només el pot modificar l'usuari.

La comanda `chmod` modifica la màscara afegint o eliminant drets a cada tipus d'usuari i s'utilitza per assignar els diferents permisos. Tots els paràmetres apareixen resumits a la **Taula 4**.

**Taula 4 -** Paràmetres per modificar els permisos dels usuaris

| Paràmetre | Nivell | Descripció |
|---|---|---|
| u | Propietari | propietari de l'arxiu o directori |
| g | Grup | grup al que pertany l'arxiu |
| o | Altres | tota la resta d'usuaris |
| **Permís** | **Identifica** | |
| r | Permís de lectura | |
| w | Permís d'escriptura | |
| x | Permís d'execució | |

Font: Adaptat de (Perseo, 2011)

En el cas que el tipus d'usuari no estigui especificat, l'ordre donada afectarà a tots els usuaris de forma simultània. L'única manera de fer un bon ús d'aquesta comanda és recordant a qui volem donar o treure permisos i quin és el permís que ens interessa utilitzar fent ús dels paràmetres següents ja mencionats anteriorment.

Tot i així, existeix una altra manera d'utilitzar la comanda `chmod` amb la que molts usuaris afirmen que, tot i ser més còmoda, és més complicada d'entendre. Aquesta segona forma és utilitzada amb un format numèric octal ja que la combinació de valors de cada grup d'usuaris en forma un. Per aquest mètode cal tenir en compte el següent canvi:

- `r` = 4: el bit és 20, per tant, 1
- `w` = 2: el bit és 21, per tant, 2
- `x` = 1: el bit és 22, per tant, 4

Aquesta combinació de bits ens porta fins als 8 possibles valors que, quan les combinem amb els permisos d'usuari, grup i altres, obtenim nombres de tres xifres (veure **Taula 5**).

**Taula 5 -** Combinacions de bits en el format numèric octal

| Permís | Valor octal | Descripció |
|---|---|---|
| `---` | 0 | No té cap permís |
| `--x` | 1 | Només té permís d'execució |
| `-w-` | 2 | Només té permís d'escriptura |
| `-wx` | 3 | Permisos d'escriptura i execució |
| `r--` | 4 | Només té permís de lectura |
| `r-x` | 5 | Permisos de lectura i execució |
| `rw-` | 6 | Permisos de lectura i escriptura |
| `rwx` | 7 | Tots els permisos, lectura, escriptura i execució |

Font: Recuperat de (Perseo, 2011)

Existeixen tres tipus de permisos més que són classificats com a especials perquè no són gaire habituals. Tot i així és necessari conèixer-los i són els següents:

- **Sticky bit**: S'utilitza sobre directoris fent que, els elements de cadascun només puguin ser eliminats i modificats de nom únicament pel propietari
- **SUID** (Set User ID): L'usuari que executa el fitxer té els mateixos permisos que el que el va crear
- **SGID** (Set Group ID): Fa la mateixa funció que el SUID però aquest no serveix per usuaris, sinó per grups

En conclusió, Linux va decidir crear els permisos per poder protegir la informació que distribuïa a través de les xarxes. Per a fer-ho va dividir tres tipus de paràmetres: els propietaris, els grups i la resta; i tres tipus de permisos: un per lectures, un altre per escriptures i un últim per execucions. Aquests permisos s'utilitzen amb la comanda `chmod` i es poden expressar tant anomenant-los, com a través del format numèric ortogonal. Finalment, Linux consta de tres permisos més que se'ls coneix com a especials degut al seu poc ús.

### 2.1.4. La importància de les llicències

Per altra banda, la importància de les llicències en el món de la informàtica és molt gran ja que estableixen un contracte entre aquell que posseeix els drets d'autor i l'entitat que l'adquireix. En aquest contracte es defineixen quins són els drets i les obligacions que ha de complir cadascú juntament amb la vigència de la llicència.

Els programaris es poden classificar segons el seu grau de llibertat en funció de quins dels següents criteris compleixen i en quina mesura ho fan:

- La llibertat per a executar el programa
- La llibertat d'estudiar com treballa el programa, i adaptar-lo a les necessitats pròpies

- La llibertat de redistribuir còpies
- La llibertat per a millorar el programa i alliberar aquestes millores al públic perquè tothom pugui beneficiar-se'n

En funció dels criteris anteriors podem trobar-nos des de les llicències més privatives fins a les lliures. Les privatives són aquelles que no són lliures i que, per tant, no et deixen ni veure el codi ni reutilitzar-lo. En canvi, les lliures et permeten executar, copiar, distribuir, estudiar, canviar i millorar el programari. A part, existeixen punts entremitjos entre unes i altres que t'ofereixen els beneficis d'ambdós.

Actualment les tres llicències més importants i que engloben la resta són: la llicència BSD (Berkeley Software Distribution), la llicència GPL (GNU General Public License) i la llicència MPL (Mozilla Public License).

Les llicències del tipus BSD es poden considerar de les més permissives del software lliure ja que no tenen cap mena de restricció. Aquestes llicències es caracteritzen per la llibertat de poder comercialitzar el software i el dret de no haver de compartir el codi font. És a dir, si tens una BSD, pots modificar el programa deixant la llicència que hi havia anteriorment, pots modificar-lo i posar una llicència privativa o pots canviar del codi obert al tancat. Amb altres paraules, pots fer tot el que vulguis i necessitis amb el programa.

Les llicències del tipus GPL són un exemple de software lliure amb copyleft fort. El copyleft és una versió del copyright però per a softwares. Aquesta legislació s'encarrega d'impedir que s'utilitzi el codi font sense autorització definint les condicions amb les que es pot fer ús. Per tant, aquesta llicència és lliure ja que et dona la llibertat de modificar el programari però a canvi l'has de tornar a publicar amb la mateixa llicència que ja hi havia abans. Per exemple, si estàs modificant un programa on el codi és obert, quan l'acabis hauràs de deixar aquest codi obert.

Per últim, les llicències MPL són semblants a les BSD tot i que aquestes són molt menys permissives i es troben en el punt entremig de les dues llicències anteriors. La Mozilla Public License es considera de programari lliure amb un copyleft feble, és a dir, s'ha de mantenir la llicència però pots combinar el codi amb fitxers de programari privatiu que no siguin d'aquest mateix codi.

Linux està col·locat sota la llicència GNU Public License versió 2 (GPLv2) on el codi font és disponible per a tothom i es poden fer els canvis que vulguis mentre que, a l'hora de distribuir-ho, deixis que els altres també puguin modificar-lo beneficiant a tothom.

A la **Taula 6** es llisten els avantatges i desavantatges de tres tipus de llicències segons el seu grau de llibertat.

**Taula 6 -** Avantatges i inconvenients de les llicències BSD, GNU i privada

| | **Avantatges** | **Inconvenients** |
|---|---|---|
| **BSD**<br>**Llicència pública** | • Accés al codi font<br>• Independència del proveïdor<br>• Evolució tecnològica | • Manca de garanties<br>• Falta de seguretat<br>• Absorcions d'empreses |
| **GNU**<br>**Llicència de terme mig** | • Preu baix<br>• Llibertat de còpia<br>• Seguretat i fiabilitat<br>• Copyleft fort | • Sense garantia<br>• Poca flexibilitat<br>• Menys compatibilitat amb el hardware |
| **Llicència privada** | • Major compatibilitat amb el hardware<br>• Facilitat d'adquisició | • No es poden fer còpies<br>• No es pot modificar<br>• Restriccions d'ús<br>• Preu alt |

Font: Elaboració pròpia.

Després de fer la comparació penso que la millor opció seria utilitzar una llicència de terme mig com la GPLv3 perquè, d'aquesta manera, cap usuari podria modificar el meu programa i després canviar-lo per una llicència privada i beneficiar-se'n personalment. En canvi, una gran avantatge seria que la gent podria millorar el codi i publicar-lo fent que, tothom pugui beneficiar-se d'ell de forma equitativa.

## 2.2. Els robots mòbils

Durant molt de temps, la robòtica ha estat sempre estàtica i s'ha centrat únicament en el rendiment, la precisió de repetició i la velocitat. Els robots obtinguts fins als anys 90 tenien grans qualitats com la força però mancaven de la del moviment. Això els impedia realitzar totes les operacions desitjades i és per aquest motiu que van veure la necessitat de crear la robòtica mòbil. Gràcies a ella s'ha aconseguit una important transformació industrial i s'ha portat l'ús dels robots més enllà de les fàbriques.

Com el seu propi nom indica, la robòtica mòbil es basa en la construcció de robots que es poden moure. Però aquesta no només es centra en la mobilitat. Quan es va iniciar el desenvolupament d'aquest nou repte, els enginyers que hi treballaven es van adonar que el seu objectiu no només s'havia de basar en el moviment, sinó que també havia d'incloure l'autonomia. Perquè per exemple, com pot un robot netejar el terra d'un pis seguint ordres com moure's 2 metres endavant o girar 90 graus a la dreta? Tots els pisos són diferents i, per poder mantenir-lo tot net, cada robot haurà de realitzar uns moviments diferents. El problema és que no podem inventar un robot especial per cada pis, per això necessitem que aquest robot autònom.

Des de que es va inventar el primer robot, tots han depès dels éssers humans per poder funcionar. Quan parlem de la seva autonomia ens referim a que aquest no ens necessita, és més, a través de sensors és capaç de situar-se en l'espai i moure's cap a on nosaltres desitgem. L'ús de la robòtica mòbil no només ha suposat un gran avenç en la construcció de robots, sinó que també ho ha sigut per la programació. Actualment ja no li diem al robot quins moviments ha de seguir, sinó que li donem ordres que ha de complir segons el que es troba al llarg del seu recorregut. Amb el sensor de llum, per exemple, al robot li interessa buscar línies, i quan les troba, realitzar les comandes del programa. Gràcies a aquesta autonomia aconseguim que el robot resolgui els problemes per ell mateix i no necessiti una ajuda externa per a solucionar-los.

Actualment hi ha una gran varietat de robots mòbils terrestres i és per aquest motiu que s'han creat tres grups per poder-los classificar. Aquests grups tenen un nom genèric que defineix les seves característiques mòbils i són els següents:

- Els robots amb rodes s'utilitzen per moure material o objectes d'un costat a l'altre. Gràcies a les rodes aconsegueixen una gran mobilitat sempre i quan el terreny no sigui irregular.
- Els robots eruga o amb cadena s'utilitzen per explorar àrees noves que no estan preparades per la circulació. Tenen unes cadenes als laterals que els dóna aquesta agilitat per moure's per terrenys irregulars.
- Els robots zoomòrfics també s'utilitzen per explorar àrees desconegudes. Es diferencien dels eruga per la seva forma ja que aquests últims estan inspirats en els diferents animals que ens envolten.

### 2.2.1. Els robots amb rodes

Avui en dia, els robots que es mouen a través de rodes són els més comuns. És així perquè no estem envoltats de terrenys irregulars i el que realment necessitem són màquines àgils que es moguin amb facilitat. Una altra raó de perquè són tan comuns és per la programació. Al tenir diferents rodes individuals per moure's, permet un codi més simple tant pels moviments com pels girs. L'últim punt a destacar és la seva cinemàtica. Quan parlem de la cinemàtica d'un robot ens referim a la distribució de les seves rodes motrius. A diferència dels altres robots terrestres, els que es desplacen amb rodes presenten diferents configuracions cinemàtiques ja existeixen més possibles distribucions.

Dins de les configuracions cinemàtiques ens trobem amb tres tipus de rodes que tenen funcions diferents. El primer grup són les rodes motrius. Aquestes rodes són les que estan connectades al motor i, per tant, es mouen endavant i enrere depenent de la potència que doni el motor. Tot seguit hi ha les rodes directrius. Tal i com el seu nom

indica, aquestes dirigeixen la direcció del robot. No estan connectades a cap motor i es poden moure cap als costats. En els cotxes, per exemple, les rodes directrius són les que estan connectades al volant. Per aquest motiu, quan girem el volant cap a un costat, hi ha dos rodes que segueixen el seu moviment. Finalment, ens trobem amb les rodes passives. Aquestes tampoc estan connectades a cap motor i, en comptes de dirigir el cotxe, actuen com a suport per al robot. Quan parlem d'aquestes rodes ens referim a les dels carros de la compra, per exemple. Aquestes segueixen la direcció del robot i es poden moure en tots els sentits.

A continuació s'expliquen detalladament quines són les diferents configuracions cinemàtiques que poden presentar els robots terrestres que es mouen amb rodes.

- **Diferencial**: Aquesta configuració és la més vista perquè és la més senzilla de muntar. Consta de dos rodes motrius col·locades als laterals del robot (veure **Figura 6**). Aquestes només és poden moure endavant o enrere depenent de la potencia i, per girar, s'ha d'activar un sol motor.

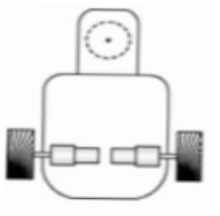


***Figura 6**. Configuració diferencial (Barrientos Sotelo & García Sánchez, 2007)*

- **Ackerman**: Aquesta configuració és molt popular en l'àmbit automobilístic. Aquesta té dos rodes motrius i dos de directrius (veure **Figura 7**). Avui en dia, es pot entendre fàcilment aquesta configuració si s'estudia el moviment d'un cotxe.

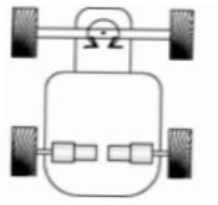


***Figura 7**. Configuració Ackerman (Barrientos Sotelo & García Sánchez, 2007)*

- **En tricicle**: Aquesta configuració consta de tres rodes. Una d'elles és directriu i motriu i, les altres dues, són passives (veure **Figura 8**). D'aquesta manera es duu a terme tot el moviment del robot des d'un sol motor i les altres només fan de suport.

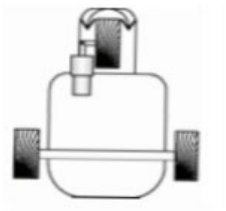


***Figura 8**. Configuració en tricicle (Barrientos Sotelo & García Sánchez, 2007)*

- **Skid-steer**: Tot i ser molt semblant a la configuració diferencial, la Skid-steer consta de més de dos rodes motrius (veure **Figura 9**). Aquesta configuració utilitza el mateix moviment que el del robot eruga diferenciant-se per la quantitat de rodes.

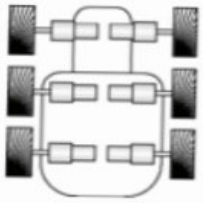


***Figura 9**. Configuració Skid-steer (Barrientos Sotelo & García Sánchez, 2007)*

- **Síncrona**: En aquesta configuració hi ha tres rodes connectades entre elles que estan col·locades a cada extrem del robot. A més, només una d'elles està connectada al motor (veure **Figura 10**). Com que es mouen de manera síncrona, permet que el robot realitzi un moviment lliure, és a dir, en totes direccions.

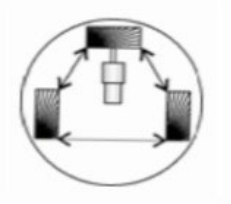


***Figura 10**. Configuració Síncrona (Barrientos Sotelo &García Sánchez, 2007)*

- **Holonòmic**: Molt semblant a la configuració síncrona tot i que es diferencia per la quantitat de motors. En aquesta configuració totes les rodes són motrius i, per aconseguir la llibertat de moviment, s'utilitzen unes rodes especials. Aquestes rodes presenten unes altres rodes petites a dins anomenades corrons i s'utilitzen en els robot que necessiten moure's en totes les direccions (veure **Figura 11**)

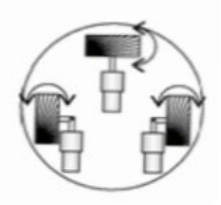


***Figura 11**. Configuració omnidireccional (Barrientos Sotelo & García Sánchez, 2007)*

## 2.3. Ordinadors de placa

Una vegada havia començat la revolució tecnològica i ja s'havien creat els primers ordinadors, els enginyers que treballaven en aquest projecte van voler anar més enllà. Es van adonar que no era del tot còmode treballar amb plaques plenes de circuits i cablejats. Necessitaven inventar algun objecte molt més petit que fos capaç de realitzar les mateixes funcions. Va ser així com, durant la dècada dels 70, es va crear la primera placa SBC (Single Board Computer). Una placa que destacava per ser igual de gran que el palmell de les nostres mans. Aquesta constava d'un sol circuit que incloïa la memòria RAM, l'entrada i la sortida de corrent, un microprocessador i totes les altres característiques necessàries.

Va ser amb la venta del primer ordinador de placa que es va veure l'èxit que tenia aquell producte. Així doncs, a mesura que la tecnologia anava progressant, les plaques milloraven al mateix ritme sent més petites i més barates. És gràcies a tot el que es va aconseguir que, avui en dia hi ha diverses empreses que competeixen entre elles per crear la millor placa SBC amb relació al preu. D'entre aquestes empreses en destaquen Banana Pi (veure **Figura 12**), BeagleBone Black (veure **Figura 13**), Raspberry Pi (veure **Figura 14**) i Arduino (veure **Figura 15**). Tot i que les dues primeres siguin de les més conegudes, els dos ordinadors de placa més venuts actualment són Raspberry i Arduino.

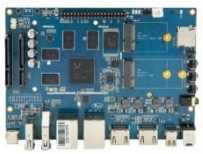


***Figura 12**. SBC Banana Pi (BananaPi, 2018)*

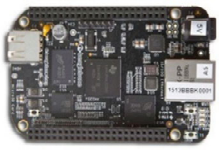


***Figura 13**. SBC BeagleBone Black (BeagleBoard, 2019)*

***Figura 14**. SBC Raspberry Pi (RaspberryPi, 2020)*

***Figura 15**. SBC Arduino (Arduino, 2020)*

La història de la Fundació Raspberry Pi va començar l'any 2006 a la universitat de Cambridge, al Regne Unit. Tant els estudiants com els enginyers buscaven la creació d'una placa SBC per poder fomentar l'ensenyament de la ciència i de la computació als

nens. No va ser fins a l'any 2012 que van oferir al públic el seu primer ordinador de placa però, des de llavors, tots els seus materials han sigut els més venuts per varis motius. El primer de tots és que permet un ús tant en l'àmbit educatiu com en l'àmbit professional, donant així la possibilitat de realitzar projectes molt variats. Si ens centrem en el software, Raspberry Pi consta d'un sistema operatiu basat en Debian però, com a avantatge, també permet utilitzar altres sistemes operatius com Windows o Mac. El mateix passa amb el llenguatge, tot i que aquesta fundació fomenta l'ús de Python, també és compatible amb C, Perl i Rubi, entre d'altres.

Tot i que la Fundació Raspberry Pi s'hagi situat en un lloc tan alt dins del mercat, té una empresa rival que li suposa una gran competitivitat. En aquest cas parlem d'Arduino. Aquest projecte el va iniciar un jove estudiant d'Itàlia l'any 2006 amb un propòsit similar al dels creadors de Raspberry Pi, facilitar als estudiants l'aprenentatge de computació i d'electrònica. Va ser amb l'evolució del projecte que es van anar incorporant enginyers i van acabar creant un grup de treball que actualment s'ha convertit en una empresa. Igual que la Raspberry Pi, Arduino també permet el seu ús a diversos àmbits oferint així a les empreses o qualsevol persona crear les seves pròpies plaques. A part, si parlem del software, consta d'un programari lliure amb C++ com a llenguatge principal.

A la **Taula 7**, es resumeixen les característiques de la Raspberry Pi 4 i Arduino UNO.

**Taula 7 -** Comparativa entre els models Raspberry Pi 4 i Arduino UNO

| | **Raspberry Pi 4B** | **Arduino Uno** |
|---|---|---|
| **Processador** | 1,5 GHz | 16 MHz |
| **Nucli** | Quadcore | Monocore |
| **RAM** | 4 GB | 2 KB |
| **Ports USB** | 4 | 1 |
| **Pins GPIO** | 40 | 20 |

Font: Adaptat de (Pérez Martín, 2016)

Després d'haver fet aquesta comparació he vist que la Raspberry Pi em proporciona tot el que necessito tant en quantitat com en qualitat. Un robot logístic necessita un processador i una RAM potents per poder realitzar bons càlculs i guardar tota la informació necessària. A part, també és necessari l'Ethernet per poder connectar-te des d'altres ordinadors a la placa SBC. Respecte als altres punts com el nombre de GPIOS o la quantitat de ports USB, Raspberry Pi en conté un major nombre i, per tant, tenint en compte que no hi ha cap punt on Arduino sigui superior, l'elecció final per dur a terme el meu treball de recerca és la Raspberry Pi 4.

## 2.4. Visió per computador

Nosaltres, els humans, sempre hem tingut la capacitat innata de mirar, de poder saber què és tot el que estem veient, amb molta facilitat. Constantment interpretem imatges, les entenem i en sabem detectar totes aquelles parts que més ens interessen. Però i els computadors? Fins fa poc, aquests no sabien llegir cap mena d'imatge, tan sols veien que aquesta estava formada per infinitat de nombres. Ha sigut amb el pas dels anys que, gràcies a la visió per computador, cada vegada són més capaços d'entendre aquestes imatges que nosaltres els mostrem.

La visió per computador no deixa de ser una branca de la intel·ligència artificial en la que destaca l'estudi del procés, l'anàlisi i la interpretació de les imatges. Actualment, un computador és capaç de determinar quin és l'objecte que li mostres a la imatge, és capaç de detectar quan una persona està somrient o fins i tot diferenciar dos objectes molt semblants. Aquesta branca consta de varies funcions on la més destacable és la detecció d'objectes.

Tot i que nosaltres trobem una feina molt fàcil reconèixer objectes, per la història dels computadors ha portat molta feina. Com he dit anteriorment, aquests només veuen una infinitat de nombres en una fotografia i no saben què fer amb tots ells. És per aquest motiu que, amb el pas del temps, s'han anat aconseguint moltes millores en aquest

aspecte. Totes aquestes millores ens han portat a l'actualitat, on s'està aconseguint que aquesta detecció d'objectes segueixi semblant senzilla tot i que hi hagi un llarg procés al darrere. Aquest es divideix en diferents fases que ens faciliten a arribar al resultat final i desitjat: la diferenciació de l'objecte escollit envers els altres.

### 2.4.1. Procés de la visió per computador

Per obtenir la informació necessària a partir de les imatges, s'ha de dur a terme un procés per poder-ne extreure aquesta informació. Aquest procés de la visió artificial consta de sis fases (veure **Figura 16**): la digitalització, el preprocessament, la segmentació, la parametrització, la classificació i la interpretació. A continuació s'explicaran, amb detall, cadascuna de les etapes.

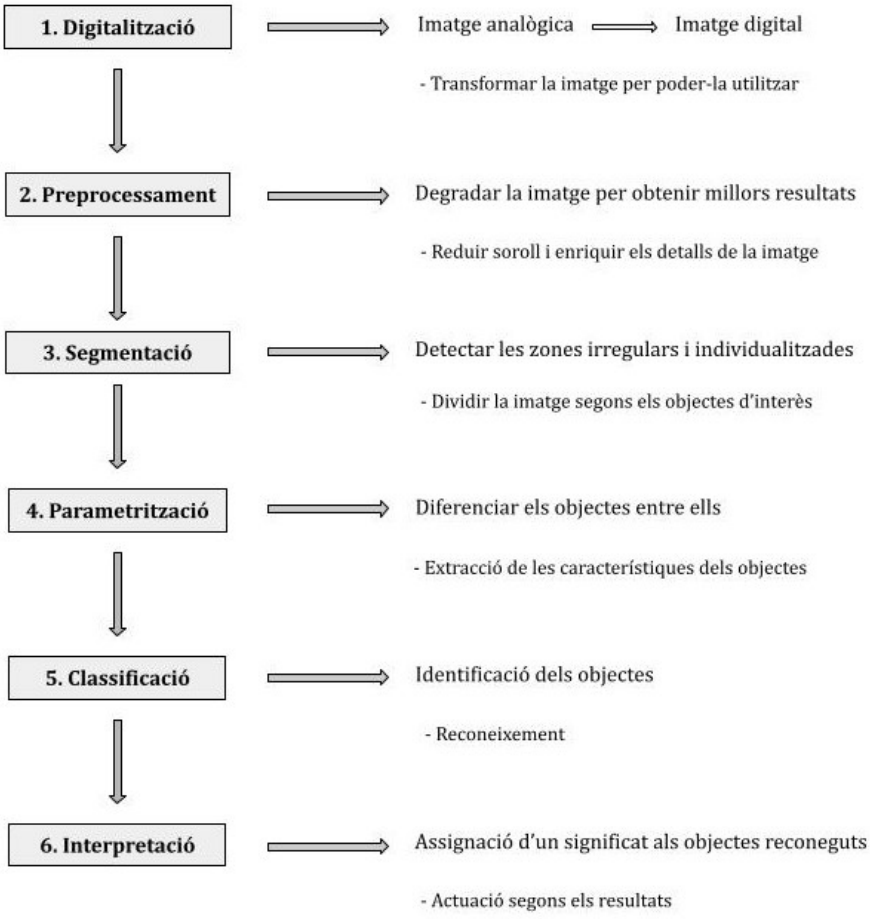



***Figura 16**. Etapes que formen la visió per computador. Elaboració pròpia.*

#### 2.4.1.1. Digitalització

Per poder interpretar una imatge de manera adequada, és essencial que aquesta estigui en les millors condicions. És per aquest motiu que la primera fase és la digitalització. En aquesta etapa es tracta d'aconseguir capturar les imatges més adients per poder continuar realitzant les següents etapes amb èxit i obtenir el resultat desitjat.

A l'hora de realitzar la fotografia cal tenir en compte els dos grups de factors que existeixen dins de la digitalització. El primer és el sistema de hardware de la visió artificial on s'hi inclouen tots els aparells que s'utilitzen a l'hora de capturar la imatge. Aquests són la càmera, la seva òptica i el dispositiu on es guarda la imatge. Si ens centrem en la càmera, és important la decisió que prenem quan escollim el model perquè, depenent del que vulguem capturar, necessitarem que aquesta tingui unes característiques determinades. Si per exemple volem detectar el color d'un objecte, necessitarem una càmera de color, però, en canvi, si el que volem detectar és la seva forma, amb una de monocromàtica (escala de grisos) en podem tenir suficient.

Per capturar una imatge, però, no només hem de tenir en compte què utilitzem, si no que també ens hem de fixar amb quines condicions ho fem. Aquests factors formen part del segon grup que té en compte l'entorn amb aspectes com la il·luminació o el fons, i el posicionament de la càmera, elements i el soroll òptic extern. La il·luminació és un dels factors més importants ja que, si fem la fotografia en unes condicions on hi ha massa llum, quan iniciem les següents fases, serà complicat diferenciar l'objecte que nosaltres busquem perquè hi apareixerà una taca blanca per l'excés d'aquesta. I el mateix ens passaria a la inversa, si una fotografia està feta amb escassa il·luminació, tampoc seria fàcil determinar-ne l'objecte perquè, en aquest cas, obtindríem una taca negra.

A continuació, a la **Taula 8**, es determinen quins són els dos grups i els seus factors que apareixen a la digitalització.

**Taula 8 -** Factors necessaris dins del procés de digitalització

| Tecnologia | Condicions |
|---|---|
| • Càmera<br>• Òptica de la càmera<br>• Dispositiu d'emmagatzematge | • Il·luminació<br>• Fons<br>• Posicionament de la càmera<br>• Elements i soroll òptic extern |

Font: Elaboració pròpia.

Una vegada tenim la fotografia definitiva, hem de transformar la imatge analògica, que és la captada per la càmera i que encara no està llesta per poder ser a editada, a una imatge digital. És a dir, de la que disposem en el computador i hi podem aplicar filtres o transformacions, en altres paraules, la podem modificar. D'aquesta manera, una vegada disposem de la segona, podrem passar a la següent fase tenint en compte que ja tindrem la imatge llesta per manipular.

Perquè el computador sigui capaç de manipular la imatge resultant necessitem realitzar aquesta transformació d'analògica a digital. Una imatge digital està formada per un conjunt de matrius numèriques basades en nombres binaris i, per poder donar per acabada aquesta fase, la càmera ha de seguir diferents passos fins a obtenir la imatge desitjada.

Primer de tot la càmera captura la llum de la imatge que està enfocant. Aquesta llum travessa alguns filtres i arriba al sensor d'imatge també anomenat CCD o CMOS que, format per multituds de fotodíodes, dispositiu que deixa anar càrregues elèctriques proporcionals a la intensitat de llum que rep, permet que la llum en realitzi una a cada receptor. Mitjançant aquesta càrrega elèctrica, el conversor ADC (Analog to Digital Converter) realitzarà la transformació desitjada convertint les dades analògiques a dades digitals i aconseguint així la imatge digitalitzada.

### 2.4.1.2. Preprocessament

Quan llegim una imatge des de qualsevol dispositiu hi podem veure certs defectes com el soroll, és a dir, imperfeccions inesperades a la imatge, o la pèrdua de definició. A aquest fet l'anomenem degradació i és provocat pels sensors de captura o les imprecisions d'enfocament i el moviment de la càmera. Com és d'esperar, aquests defectes no ens interessen per poder treballar adientment amb la imatge i, per tal de fer-los desaparèixer el màxim possible, realitzem la fase del preprocessament amb l'objectiu d'aconseguir enriquir tots els detalls de la imatge.

Per solucionar aquest problema i intentar millorar el resultat final de la imatge, existeixen diversos algoritmes de preprocessat que ens permeten modificar-la i eliminar-ne tant el soroll com el mal contrast. De totes maneres, no es recomana abusar d'aquests recursos perquè pot repercutir la imatge indicant així que la fase de digitalització no ha sigut la més adequada.

Aquest però, no és l'únic problema que hem d'intentar millorar. Malauradament, al captar una imatge, existeixen diferents característiques que ens empitjoren la seva lectura i, per tal de poder facilitar-la, fem ús de les operacions de millora de la imatge. Aquestes operacions es basen en diverses tècniques com eliminar lluentors o bé, millorar les textures i el contrast. D'aquesta manera, amb la solució de la degradació i l'aplicació d'aquests procediments, tindrem la imatge llesta per poder passar a la següent fase.

A continuació s'expliquen quins són els grups als que pertanyen les diferents funcions mencionades anteriorment per poder aclarir conceptes.

#### 2.4.1.2.1. Espais de colors

Una vegada tenim la imatge capturada i a punt per modificar hem de decidir quines dades en volem guardar. En els espais de colors és on decidim el format de la imatge, ja sigui tenint en compte els colors, com el vermell, el verd i el blau o bé tenint en compte el seu matís, la saturació i el valor. Per cada dada que guardem, es crea una matriu que

anomenem canal tenint així, en els casos anteriors, una imatge amb tres canals diferents (veure **Figura 17**). Una vegada hi ha creades les diferents matrius, aquestes es combinen creant així la imatge final. Cal saber que, com més canals tingui la imatge, més varietat de colors obtindrem.

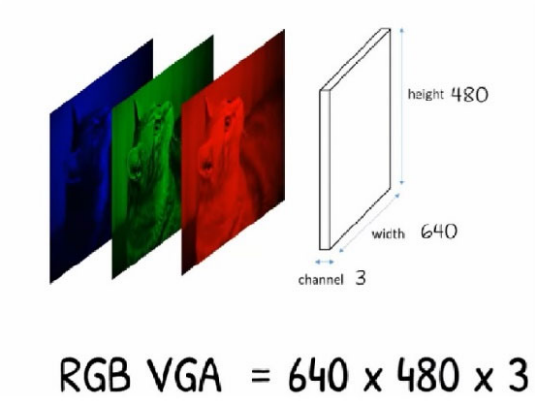



***Figura 17**. Imatge amb espai de color RGB. (Hassan, 2020)*

Tot seguit, a la **Figura 18**) es fa una comparació dels diferents espais de colors juntament amb l'original utilitzant una mateixa imatge:

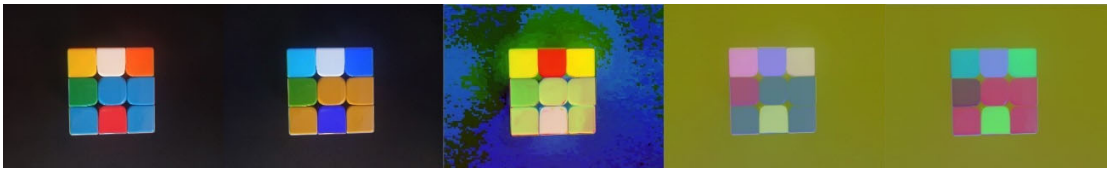
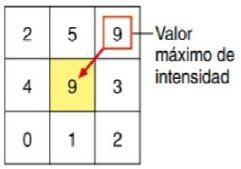
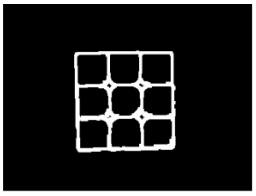
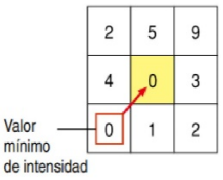
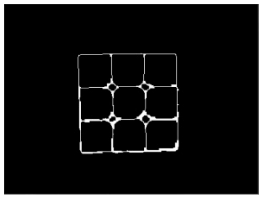


***Figura 18**. Relació dels tipus d'espais de colors. D'esquerra a dreta: BGR (Original), RGB, HSV, LAB i YCrCB. Elaboració pròpia.*

### 2.4.1.2.2. Filtres

Quan s'aplica un filtre a una imatge s'estan modificant-ne els píxels per tal d'aconseguir una altra imatge semblant però amb més facilitat per aplicar-hi modificacions com eliminar-ne el soroll o ressaltar-ne les vores i les cantonades. Aquest és l'objectiu del filtre, facilitar la feina a les altres funcions. Per fer-ho es crea una matriu anomenada kernel que es dedica a modificar aquests píxels depenent dels criteris que tinguis per canviar la imatge. És a dir, depenent dels nombres que tingui al voltant, agafarà el més gran o el més petit i en canviarà la resta. A la **Taula 9** s'explica gràficament aquest procés.

**Taula 9 -** Explicació i aplicació de dos dels filtres més coneguts de la llibreria OpenCV

| Kernel | Imatge | Kernel | Imatge |
|---|---|---|---|
|  |  |  |  |
| ***Figura 19**. Kernel de la funció Dilate. (Valcare, 2014)* | ***Figura 20**. Resultat de la funció Dilate. Elaboració pròpia.* | ***Figura 21**. Kernel de la funció Erode. (Valcare, 2014)* | ***Figura 22**. Resultat de la funció Erode. Elaboració pròpia.* |

Font: Elaboració pròpia.

### 2.4.1.2.3. Soroll

Definim el soroll d'una imatge com alteracions aleatòries de la llum o del color. Quan adquirim la imatge digital, moltes vegades ens apareixen puntets no desitjats que fan que aquesta perdi qualitat. Aquests puntets en realitat són píxels que no han guardat correctament el seu valor i han provocat un canvi d'il·luminació o de color molt exagerat. Com és d'esperar, aquest fenomen ens perjudica perquè el que destaca de la imatge són aquestes alteracions i no els objectes que a nosaltres ens interessen. Per això, per tal d'eliminar aquest soroll, el que es fa a la visió per computador és difuminar la imatge de tal manera que els puntets desapareguin ressaltant així els espais que a nosaltres ens interessen. A la **Figura 23** es compara la imatge original amb la imatge obtinguda després d'aplicar la funció *Gaussian Blur*.

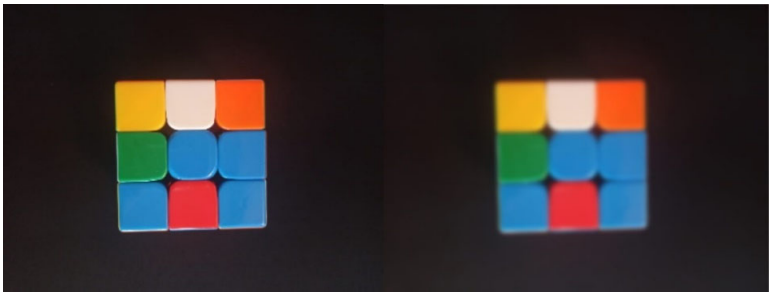


***Figura 23**. Eliminació del soroll a través del filtre Gaussian Blur. Elaboració pròpia.*

### 2.4.1.2.4. Vores i cantonades

Quan parlem de vores i cantonades ens referim als canvis bruscos en la intensitat del color, és a dir, tots aquells punts on passem d'un color clar a un de fosc. En aquest cas, les vores i les cantonades fan de frontera entre aquestes diferents àrees i per tal de detectar-les només ens cal buscar aquesta diferència de tons tan brusca. A la **Figura 24** es compara la imatge original amb la imatge obtinguda després d'aplicar la funció *Canny*.

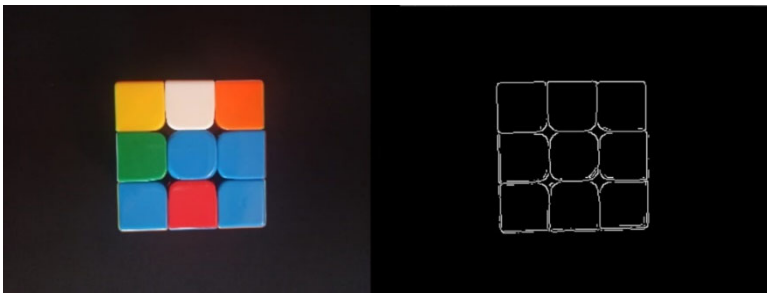


***Figura 24**. Detecció de vores i cantonades a través del filtre Canny. Elaboració pròpia.*

## 2.4.1.3. Segmentació

Una vegada hem aconseguit que la imatge sigui més senzilla de llegir, passem a la fase més important del procés de visió artificial, la segmentació. En aquesta etapa s'agrupen tots els píxels que tenen característiques idèntiques o similars en diferents zones. Un cop estan tots classificats, obtenim una imatge dividida en varis grups de píxels semblants entre ells. D'aquesta manera s'aconsegueixen diferenciar els objectes respecte el fons i detectar a quina part de la imatge es troben.

En aquesta fase ens trobem amb tres tècniques molt utilitzades. Generalment, els mètodes de segmentació es basen en les propietats de similitud i de discontinuïtat, però també existeix una tercera que és molt útil quan busques un rang de dades, és a dir, llindars. Quan parlem de similitud ens referim a les regions, aquelles àrees de la imatge on hi ha conjunts de píxels de característiques semblants. En canvi, la discontinuïtat són totes aquelles línies que separen els diferents objectes, és a dir, les vores. Finalment, la

tècnica dels llindars, s'utilitza quan hi ha una clara diferència entre les característiques dels diferents objectes que volem detectar.

Després d'haver-se aplicat aquests mètodes de segmentació, la imatge ja està preparada.

#### 2.4.1.4. Parametrització

Per poder realitzar un bon reconeixement de la imatge és necessari saber diferenciar quins són els diferents objectes que en formen part. Si tu vols detectar una poma, per exemple, no buscaràs un quadrat lila, sinó que et centraràs en les formes rodones i que, a més, són de color vermell. D'això és el que es tracta la parametrització, de diferenciar els diferents objectes de la imatge entre ells per poder-los reconèixer amb més facilitat.

És per aquest motiu que, després d'haver-se aplicat aquests mètodes de segmentació, ja es poden extreure els diferents trets de cada grup. Per fer-ho, s'etiqueten cadascuna de les zones segons les seves característiques per poder-les diferenciar més endavant. En aquesta fase mirarem la forma de l'objecte i el seu color, entre d'altres. El més complicat d'identificar és la forma i, la millor manera de reconèixer-la és fixant-se en la quantitat de vèrtexs i costats que té. És molt senzill diferenciar un quadrat d'un triangle, d'un cercle i d'un rectangle. A diferència del triangle i del cercle, el quadrat té quatre vèrtexs i, per distingir-lo del rectangle, podem mirar que tots els seus costats siguin iguals. D'aquesta manera obtindrem una correcta identificació.

Una vegada totes les etiquetes estan determinades podem passar a la següent fase i dur a terme el reconeixement de cada zona extraient així la informació més important de cada objecte.

#### 2.4.1.5. Classificació

Per poder realitzar la última fase de forma exitosa ens cal classificar d'una manera adequada els diferents objectes determinats a la imatge. Per aquest motiu és tan important aquesta penúltima etapa, perquè depenent del que assignem obtindrem els

resultats desitjats o no. Però abans de parlar dels resultats finals, és necessari centrar-se en el reconeixement de la imatge, en altres paraules, en la determinació dels objectes. Una vegada tenim extretes les característiques de les diferents zones, només ens queda anomenar cada zona, és a dir, canviar les etiquetes llargues per unes que siguin més simples i entenedores.

És a la classificació on determinem un triangle, un quadrat, un cercle o un rectangle com a tals (veure **Figura 25**). Cadascuna d'aquestes formes geomètriques tenen unes característiques diferents i, per això, ens pot ser fàcil identificar-les. En el procés anterior hem determinat el nombre de vèrtexs i costats i, en aquesta fase només ens queda determinar què són. Si ens centrem en l'exemple anterior de la poma, com ja he mencionat aquesta és rodona i vermella. En el cas que llegim una forma de quatre vèrtexs i de color vermella, sabrem que no és una poma perquè estem identificant un quadrat.

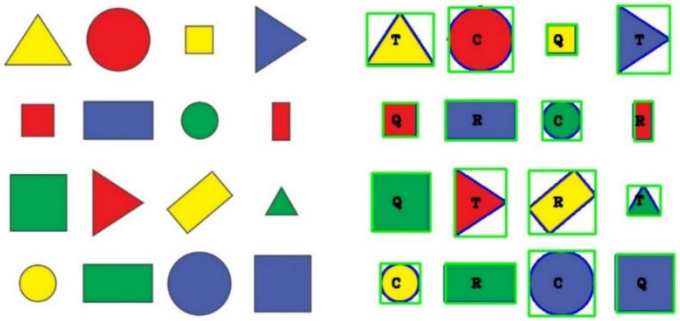



***Figura 25**. Resultat del procés de classificació. Elaboració pròpia*

Aquest és l'objectiu d'aquesta etapa, determinar què és cada zona de la imatge. Així, d'aquesta manera ja sabrem què hi ha a la imatge i si apareix o no l'objecte que estem buscant.

### 2.4.2. Llibreria OpenCV

Als inicis de la visió artificial era molt complicat programar totes les funcionalitats que se'ls hi volia donar als computadors. Hi havia tantes operacions a calcular, tantes funcions a realitzar, que els programes quedaven extensos i gairebé incomprensibles.

Per poder facilitar la feina als programadors, Intel va crear la biblioteca OpenCV l'any 1999. A partir de l'any 2000, després de poques versions, aquesta s'ha convertit en la biblioteca més popular del món de la visió artificial.

A l'inici del treball, haver de triar quina biblioteca utilitzar per dur a terme la programació de la visió per computador no va ser gens complicat. Després d'haver-me informat sobre quines llibreries hi havia, vaig tenir clar que la millor opció era OpenCV. Va ser així perquè al tenir una llicència BSD significava que podia utilitzar-la de manera gratuïta. A més, com que és la més popular, és més senzill trobar informació i cursos per aprendre'n les bases. Finalment, el que em va ajudar a decidir-me va ser el llenguatge. Des de que he començat a programar, sempre he utilitzat el Python i que OpenCV estigui adaptat a aquest llenguatge em facilita la realització del meu treball perquè no ha fet falta dedicar cert temps a aprendre un llenguatge nou.

#### 2.4.2.1. Funcions bàsiques

Al llarg del meu aprenentatge amb la llibreria OpenCV he conegut varies funcions que m'han sigut molt útils per realitzar les fases de la visió artificial. En **l'Annex 1 – Funcions bàsiques de OpenCV** es pot trobar un resum de les funcions indispensables per treballar amb la llibreria.

# 3. MARC PRÀCTIC

Un dia un professor em va plantejar la problemàtica que tenia per transportar material d'un costat a l'altre. Traginar portàtils d'un extrem del passadís a l'altre, aturar-se a la sala de professors per recollir material o transportar en una maleta els llibres de cada curs. Aquest, és un problema extrapolable a moltes altres situacions. Els petits negocis amb magatzems plens d'objectes o els carros de menjar i de medicines dels hospitals en són altres exemples.

Els robots logístics són cars i sovint, el pressupost de les empreses amb aquestes necessitats no poden fer-hi front. Després de realitzar els meus estudis obligatoris a l'escola FEDAC Sant Narcís i d'haver participat durant cinc anys a la coescolar de robòtica. Volia dissenyar, construir i programar un robot que fos capaç de transportar tot tipus de material de manera autònoma. Tot això en el cost més baix possible.

Després de parlar amb diversos sectors que podrien tenir aquestes necessitats, vaig establir dos problemes a resoldre:

- Com transportar material sense tenir en compte el pes.
- Com aconseguir realitzar un moviment autònom.

En els següents apartats es descriuen les decisions preses en el procés de disseny com ara l'estructura, la forma, l'electrònica i la programació del robot per tal de complir amb els requisits. També s'hi poden observar tots els passos seguits per construir-lo.

## 3.1. Disseny mecànic

El primer que calia decidir era el disseny que havia de tenir el robot. Una de les primeres decisions que calia prendre era com moure una certa quantitat de pes sense haver de destinar molts recursos als motors. Després de buscar com funcionaven robots logístics

com els d'Amazon, es va decidir crear diferents carros amb rodes que encaixessin amb el robot i que suportessin el pes extra que se li volgués afegir. D'aquesta manera, el robot només havia de fer força per fer moure un carro amb rodes enlloc de suportar tot el pes directament. Amb aquesta premissa, el disseny del robot es simplificava.

A continuació s'expliquen els passos seguits per prendre les decisions sobre els components necessaris, la seva distribució, el tipus de moviment, com transportar el carro i la seva forma final.

### 3.1.1. La forma i el tipus de moviment

El robot havia de permetre l'ús de diferents carros que poguessin subjectar-s'hi. Per aquest motiu, es va decidir que la millor forma era la quadrada, perquè permet encaixar-lo fàcilment amb el carro.

Les dimensions del robot havien de ser el més petites possibles per tal de reduir la mida útil del carro i fer que cabés a tot arreu. És per això que, després de tenir en compte l'espai i la distribució de tots els components, les seves mesures son de 400 x 400 x 125 mm. La decisió d'arrodonir les cantonades no va ser només per una raó estètica, sinó que també aporta autonomia al robot perquè és més complicat que quedi encallat a cantonades o llocs petits. A la **Figura 26** es pot veure el procediment seguit per aconseguir-ho.

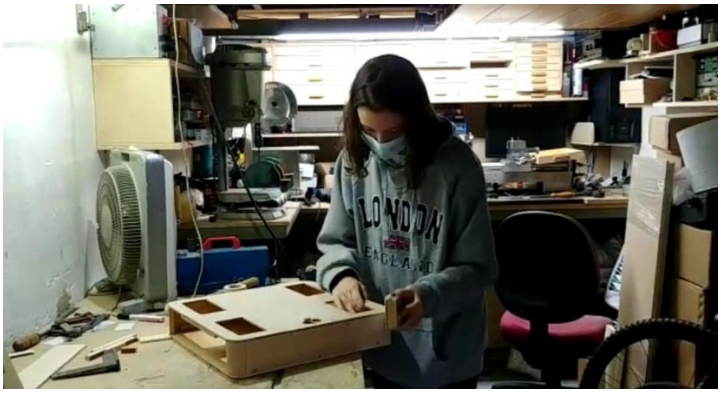



***Figura 26**. Procediment d'arrodoniment de cantonades del robot. Elaboració pròpia*

El seu exterior està construït amb fusta però la majoria de peces han estat dissenyades per mi amb el programari Solidworks i impreses amb PLA amb una Creality CR-10S Pro. Cadascun dels dissenys està explicat més endavant.

## 3.1.2. Enginyeria de requeriments, l'electrònica

A continuació s'expliquen els elements electrònics necessaris per fer funcionar el robot així com la justificació de la seva tria.

### 3.1.2.1. Sistema de control i alimentació

#### 3.1.2.1.1. Raspberry pi

Es va escollir una Raspberry Pi 4 com la de la **Figura 27** com a sistema de control pel robot. Les seves característiques, enfront a les versions velles, permeten que sigui capaç de capturar i processar vídeos en 4K, controlar els motors i estar pendent d'altres sensors sense veure'n afectat el rendiment. A diferència d'altres sistemes com l'Arduino, és capaç de fer funcionar un sistema Linux aportant molta més versatilitat al projecte. Pel què fa al pressupost, permet competir amb sistemes de control similars amb un pressupost molt més ajustat.

***Figura 27.*** *Raspberry Pi 4. Elaboració pròpia*

#### 3.1.2.1.2. Alimentació

L'alimentació del robot necessita dues fonts diferents. Per un costat, necessita 5V per la Raspberry Pi 4 i, per l'altre, necessita 12V pels motors. Per no haver de disposar de dues bateries diferents, es va optar per agafar una bateria de la marca DSK de 12 V i 1.3 Ah juntament amb un transformador per adaptar la sortida a 5V per la Raspberry Pi. El convertidor de la marca Haofy consta d'un convertidor Micro USB que, amb un

adaptador a USB C permet alimentar el sistema de control . A la **Figura 28** i a la **Figura 29** es poden veure la bateria i el transformador.


***Figura 28.*** *Bateria de la marca DSK de 12 Volts. Elaboració pròpia*


***Figura 29.*** *Convertidor de la marca Haofy. Elaboració pròpia*

D'aquesta manera amb una sola bateria s'alimenten els motors i la Raspberry Pi estalviant diners, sistemes de càrregues i guanyant espai.

#### 3.1.2.1.3. Interruptor

Per tal de poder engegar i apagar el sistema elèctric del robot sense haver-lo de desmuntar, calia un interruptor. En aquest cas, tal i com es pot veure a la **Figura 30**, es va escollir un interruptor de tres posicions que permet triar si el robot està engegat, apagat o s'està carregant. D'aquesta manera, s'eviten sobrecàrregues innecessàries al sistema i s'aïllen els diferents circuits electrònics. Per poder carregar el robot sense haver de desmuntar-lo es va comprar un connector pel carregador com el de la **Figura 31.**

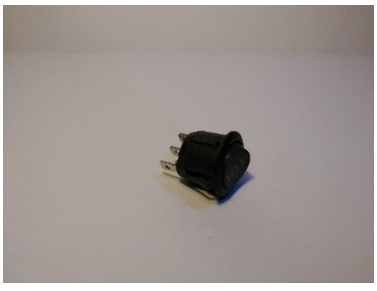


***Figura 30****. Interruptor de tres posicions. Elaboració pròpia*

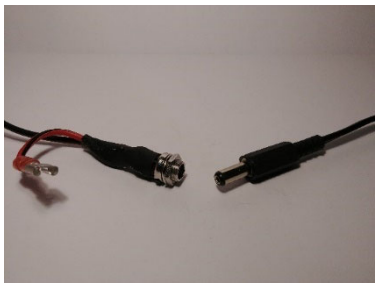


***Figura 31****. Connector pel carregador. Elaboració pròpia*

### 3.1.2.2. Sistema Motriu

#### 3.1.2.2.1. Motors

El moviment del robot es duu a terme a través de quatre motors de corrent contínua que inclouen un codificador que permet calcular els centímetres que s'ha mogut el robot.

Els motors que es van escollir són de la marca CQRobot, amb 122 RPM i un parell motor de 36 Kg.cm (veure **Figura 32**). Com que només ha de ser capaç de moure's a ell mateix i desplaçar un carro, el pes que ha de carregar és inferior als 10 Kg. Així doncs, amb aquests motors, es cobreixen les necessitats per aconseguir un robot relativament ràpid i amb capacitat de moure pes.

***Figura 32.*** *Motor de la marca CQRobot. Elaboració pròpia*

#### 3.1.2.2.2. Drivers

Engegar els motors provoca pics de tensió. Això fa que calgui separar els circuits del sistema de control del de la Raspberry Pi dels motors. Per fer-ho, es va decidir fer ús de dues plaques controladores L298N amb sortides PWM (veure **Figura 33**).

El què fa, és actuar de barrera física per evitar curtcircuits entre el sistema de control i el motriu. El seu funcionament electrònic consta d'un pont en forma de H (veure **Figura 34**) que permet invertir el sentit de la corrent.

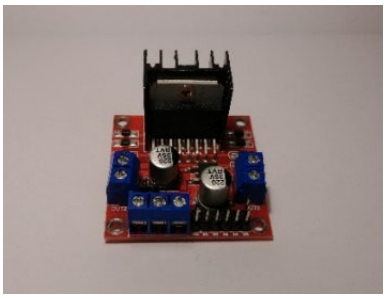


***Figura 33****. Controladora L298N. Elaboració pròpia*

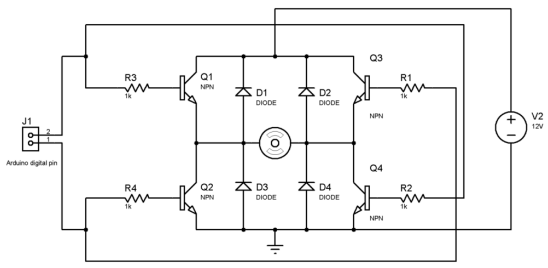



***Figura 34****. Esquema connexió pont H. (González, 2014)*

### 3.1.2.3. Sistemes d'informació

#### 3.1.2.3.1. Sensor de llum

L'ús d'un sensor de llum és necessari per poder fer un seguiment de línies. En aquest projecte, es va escollir per tal d'ajudar a fer el seguiment de rutes autònomes del robot fins a un espai concret.

***Figura 35**. Sensor de llum del paquet ELEGOO Robot Car Kit. Elaboració pròpia*

Es va decidir aprofitar els sensors de llum que venien amb el paquet ELEGOO Robot Car Kit. El dispositiu consta de tres sensors de llum amb una sortida digital per cadascun d'ells, col·locats de costat (veure **Figura 35**). Malgrat que al ser digitals no es pot fer ús d'un sistema PID, la col·locació estratègica dels sensors fa que, establint el gruix de la línia de 47 mm, es pugui fer el seguiment d'una manera suau i controlada.

#### 3.1.2.3.2. Càmera

Per tal de poder dur a terme tant el seguidor d'usuaris com la lectura de codis QR, es va decidir fer ús d'una càmera Raspberry Pi.

Com que la intenció era construir un robot econòmic, es va comprar una càmera diferent a l'oficial de la Raspberry Pi. En aquest cas, es va decantar per la marca MakerHawk (veure **Figura 36**) ja que proporcionava una resolució de 1080 p i unes condicions de llum i temps de resposta adequades. Finalment, com que el producte escollit obtenia imatges molt fosques, es va decidir comprar la Raspberry Pi Camera v2.0 (veure **Figura 37**) guanyant així fiabilitat en la programació.

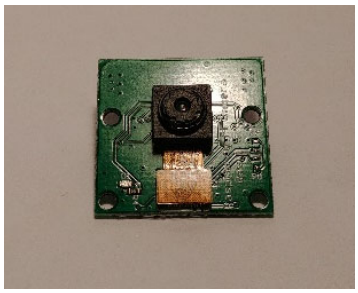


***Figura 36**. Càmera per Raspberry Pi de la marca MakerHawk.* Elaboració pròpia

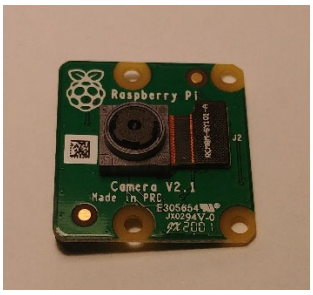



***Figura 37.** Càmera Raspberry Pi v2.0. Elaboració pròpia*

### 3.1.2.3.3. Servomotors

El robot té dues necessitats per les quals es va decidir fer ús de servomotors. La primera era l'activació d'un sistema de subjecció pel carro. Consisteix en dos biela-manovella controlades cadascuna pel servomotor LD-20MG de la marca LewanSoul (veure **Figura 38**. Els servomotors són de corrent continu i permeten controlar posicions de 0 a 180º. Per la seva construcció, permeten suportar pesos de fins a 20 Kg motiu pel qual es van escollir.


***Figura 38.** Servomotor de la marca LewanSoul. Elaboració pròpia*

La segona de les necessitats era el control de la posició de la càmera. Com que es tracta d'un element necessari tant per dur a terme el mode autònom com el del seguiment d'usuaris, cal poder moure-la perquè enfoqui en paral·lel al terra o en una posició de 35º respectivament. Per aquest motiu, es va decidir utilitzar la marca Longruner perquè, juntament amb el servo, venia un suport per la càmera que permet fer rotacions del tipus Tilt (veure **Figura 39**).

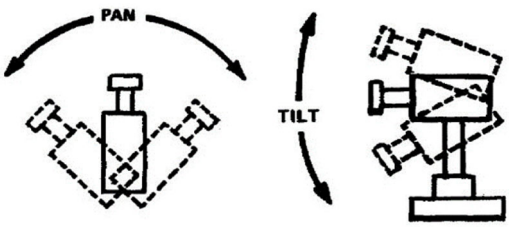



***Figura 39**. Moviment Pan i Tilt de la càmera. (Studiomaven, 2014)*

### 3.1.3. La Raspberry Pi 4 com a sistema de control

Per dur a terme el projecte del robot, es va decidir que la millor opció era que el seu sistema de control fos la Raspberry Pi 4. Com s'ha mencionat anteriorment, compleix totes les meves necessitats. La velocitat del processador i els 4Gb de RAM permeten tenir varis processos oberts a la vegada sense veure'n afectat el rendiment. A més, el fet que disposi de connexió per WiFi permetia connectar-se remotament i fer les proves d'una manera més ràpida.

Per últim, la connectivitat de sensors i actuadors externs era suficient pel meu projecte. A continuació, es detallen els diagrames de connexió emprats.

#### 3.1.3.1. Connexions i control del pins GPIO

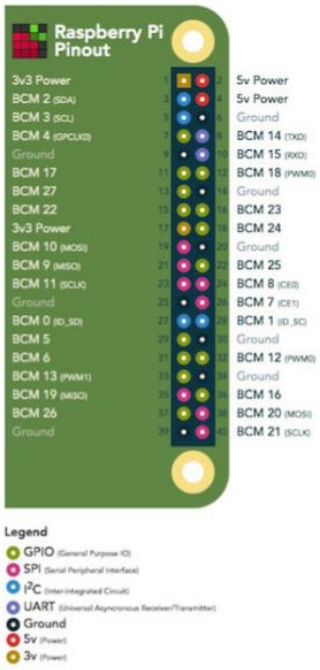
Un cop seleccionats tots els elements que es necessitaven, calia veure com es podien connectar a la Raspberry Pi 4. L'ordinador de placa disposa d'un total de 40 ports GPIO que permeten rebre o enviar informació. En total, es necessitaven 41 pins d'alimentació, 13 d'entrada i 15 de sortida analògica i digital.

El primer problema va ser veure que no hi havia prou VCC i GND per connectar-ho directament. Així doncs, va ser necessari realitzar algunes connexions amb una regleta (veure **Figura 40**) per tal de poder disposar dels ports d'alimentació necessaris.

***Figura 40**. Regleta per les connexions. Elaboració pròpia*

El segon problema va ser no disposar de suficients GPIO de control analògic o PWM (simulació de senyals analògiques a través d'impulsos elèctrics). Es necessitaven 7 sortides analògiques per tal de controlar els quatre motors i els tres servomotors. Investigant, es va trobar que tots els pins es poden configurar com a PWM a través de programari amb la llibreria RPi.GPIO. Així doncs, configurant el tipus de GPIO es va aconseguir tenir tots els pins necessaris pel robot.

A continuació, a la **Figura 41**, es pot veure l'esquema dels pins GPIO que facilita la instal·lació del robot cada vegada que és necessari. La numeració utilitzada és la BCM, la més usada amb les llibreries RPi.GPIO i GPIO Zero.




| | |
|---|---|
| **3V3 power** - final de carrera | **5V power** - sensor de llum |
| **GPIO 2 (SDA)** - PWM servo 1 (esquerre) | **5V power** - servo |
| **GPIO 3 (SCL)** - PWM servo 2 (dret) | **Ground** - servo dalt (dret) |
| **GPIO 4** - final de carrera 1 (esquerre) | **GPIO 14** - final de carrera 2 (dret) |
| **Ground** - servo dalt (esquerre) | **GPIO 15** - motor davant esquerre 5 (groc) |
| **GPIO 17** - (PWM) driver esquerre 1 | **GPIO 18** - motor davant esquerre 6 (blanc) |
| **GPIO 27** - driver esquerre 2 (endavant) | **Ground** - final de carrera |
| **GPIO 22** - driver esquerre 3 (enrere) | **GPIO 23** - motor davant dret 5 (groc) |
| **3V3 power** - motor | **GPIO 24** - motor davant dret 6 (blanc) |
| **GPIO 10** - (PWM) driver esquerre 6 | **Ground** - final de carrera |
| **GPIO 9** - driver esquerre 5 (enrere) | **GPIO 25** - motor darrera esquerre 5 (groc) |
| **GPIO 11** - driver esquerre 4 (endavant) | **GPIO 8** - motor darrera esquerre 6 (blanc) |
| **Ground** - sensor de llum | **GPIO 7** - motor darrera dret 5 (groc) |
| **GPIO 0** - (PWM) driver dret 1 | **GPIO 1** - motor darrera dret 6 (blanc) |
| **GPIO 5** - driver dret 2 (endavant) | **Ground** - driver |
| **GPIO 6** - driver dret 3 (enrere) | **GPIO 12 (PWM0)** - servo càmera |
| **GPIO 13 (PWM1)** - driver dret 6 | **Ground** - servo càmera |
| **GPIO 19** - driver dret 5 (enrere) | **GPIO 16** - sensor de llum 1 (esquerre) |
| **GPIO 26** - driver dret 4 (endavant) | **GPIO 20** - sensor de llum 2 (mig) |
| **Ground** - motor | **GPIO 21** - sensor de llum 3 (dret) |

***Figura 41**. Esquema dels pins GPIO a la Raspberry. Elaboració pròpia*

### 3.1.3.2. Diagrama de connexions del sistema

Abans de començar a connectar tots els elements però, es va fer un esquema de connexions com el de la **Figura 42** per assegurar que totes les decisions preses anteriorment i l'esquema dels pins GPIO eren correctes.

A més de les connexions amb el sistema de control, calia tenir una perspectiva de com quedaria tot connectat. A la **Figura 43** es poden observar tots els components electrònics utilitzats i de quina manera estan connectats. Així doncs, es facilitava el procés de connexió de cables cada vegada que era necessari repetir-lo.

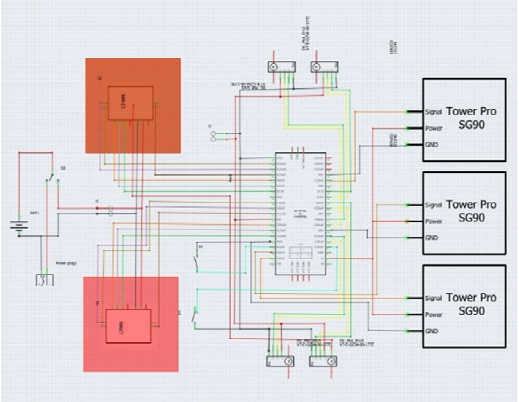



***Figura 42**. Esquema de les connexions dels elements. Elaboració pròpia*

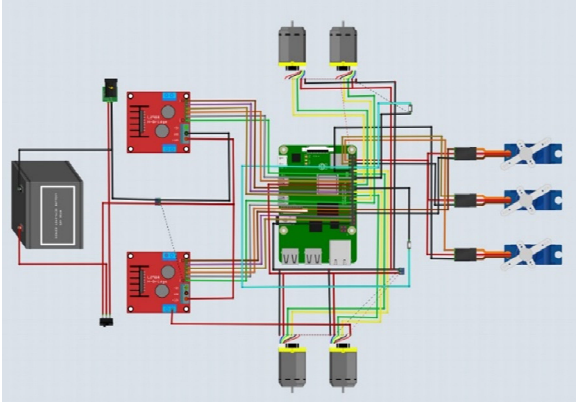


***Figura 43.** Disseny de les connexió dels component electrònics. Elaboració pròpia*

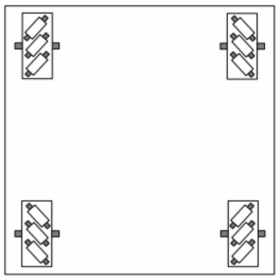
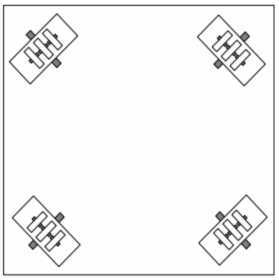
### 3.1.4. Un *drivetrain* holonòmic

Des de l'inici del projecte es va detectar que la necessitat de moviment del robot era senzilla. Per una banda, el desplaçament era per superfícies planes. Per l'altra, el moviment consistia en poder seguir tant persones com línies. En ambdós casos, el robot necessitava poder modificar la seva direcció de manera instantània. En el cas del seguiment de línies, és necessari per poder aconseguir un moviment més suau. En el cas del seguiment d'usuaris, el treballador farà canvis de sentit i això no hauria de suposar un problema pel robot.

Vistes totes aquestes necessitats i els tipus de moviments del robot analitzats al capítol 3, el millor sistema de moviment pel robot era l'holonòmic. El motiu principal és que permet desplaçar-se en qualsevol sentit, sense fer maniobres prèvies, i permetent moure's en totes direccions immediatament.

Per fer aquest tipus de moviments, existeixen principalment dues classes de rodes: les mecanum i les omnidireccionals. Les seves característiques estan resumides a la **Taula 10**.

**Taula 10 -** Comparació dels tipus de rodes holonòmiques

| | Rodes mecanum | Rodes omnidireccionals |
|---|---|---|
| **Descripció** |  Rodes amb corrons en posició de 45º |  Rodes amb corrons en posició de 90º |
| **Avantatges** | • Disseny compacte<br>• Bona capacitat de càrrega<br>• Menys velocitat i tracció quan va en diagonal | • Disseny compacte<br>• Més estètic i menys pes<br>• Menys velocitat i tracció quan va en diagonal |
| **Desavantatges** | • Alta sensibilitat a si una roda per contacte amb el terra.<br>• Alta sensibilitat al terra. No poden fer-se servir en terrenys irregulars<br>• Disseny de les rodes complicat | • Alta sensibilitat a si una roda per contacte amb el terra i als errors en el diàmetre de les rodes.<br>• Alta sensibilitat al terra. No poden fer-se servir en terrenys irregulars<br>• Menys tracció |

Font: Adaptat de (Florentina & Doroftei, 2011)

Després de l'estudi fet, la decisió que es va prendre va ser fer ús de les rodes mecanum perquè tenien més tracció que les omnidireccionals i per arrossegar el carro seria la millor opció. Pel què fa a l'estètica, com que les rodes estan amagades dins la caixa que tapa el robot, no representava un problema.

#### 3.1.4.1. Les rodes mecanum

Primer de tot, es van haver de triar el model de rodes i les mides desitjades. Com que la seva compra era més senzilla per internet, es va decidir buscar els distribuïdors més coneguts tenint en compte que el transport fos des d'Espanya o amb un temps d'enviament curt. Per aquest motiu, els va decidir decantar-se per Amazon i Robotshop.

Respecte les rodes, el diàmetre desitjat era d'entre 80 mm i 100 mm perquè eren les mides més raonables per un robot d'aquelles mides. A continuació, a la **Taula 11** es fa una comparació de les diferents rodes trobades i el seu preu.

**Taula 11** - Comparativa de preus de rodes mecanum

| Distribuidor | Model | Dimensions | Preu |
|---|---|---|---|
| Amazon | OSOYOO | 80 mm | 90,97€ |
| Amazon | Gaoominy | 60 mm | 35,31€ |
| Robotshop | Nexus Robot | 60 mm | 91,97€ |
| Robotshop | Nexus Robot | 100 mm | 123,67€ |

Font: Elaboració pròpia

Una de les premisses a la que es va arribar és que calien rodes el més gran possibles, per sobre dels 80mm de diàmetre. Aquestes eren molt costoses així que la decisió final va ser fer-les amb impressió 3D. Finalment, buscant models ja dissenyats es van escollir unes rodes de 75 mm d'en (Dejan, 2019) que, tot i no ser del diàmetre desitjat, era el model que més si acostava.

La primera iteració d'impressió només va servir per fer proves perquè la qualitat, tal com es pot veure a la **Figura 44**, va ser molt baixa i els corrons no eren prou llisos.

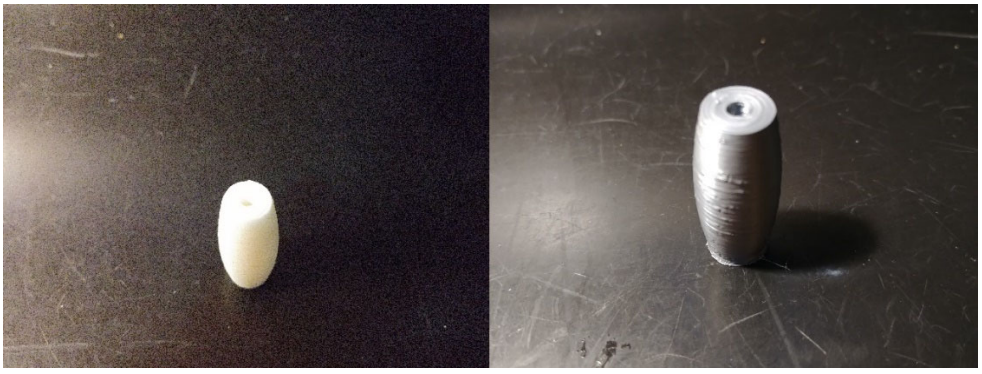


***Figura 44.*** *Diferència entre corró amb baixa i alta qualitat. Elaboració pròpia*

La següent impressió va ser amb una altura de capa de 0.2 mm i una velocitat molt lenta. Malgrat això, en la impressió va aparèixer una línia vertical a la paret exterior del corró.

Després d'investigar, es va descobrir que el problema era degut a una opció del Cura, el programari per convertir un disseny 3D en instruccions per la impressora. Per defecte, estableix que l'inici i final de capa d'un cilindre és sempre a la mateixa posició fent que sempre hi hagi un excedent de fil al mateix lloc que acaba provocant la línia vertical. Per solucionar-ho, es va haver de marcar la opció del *Z seam* com a *Random* fent que l'inici i el final sigui diferent per cada capa.

Amb aquesta última opció es va poder fer la unió dels corrons amb les seves respectives bases. Per unir-los, es va fer ús d'una vareta de 2 mm de diàmetre amb una gota de *Loctite* per evitar que es desmuntessin. Les rodes muntades van quedar com es veuen a la **Figura 45**.

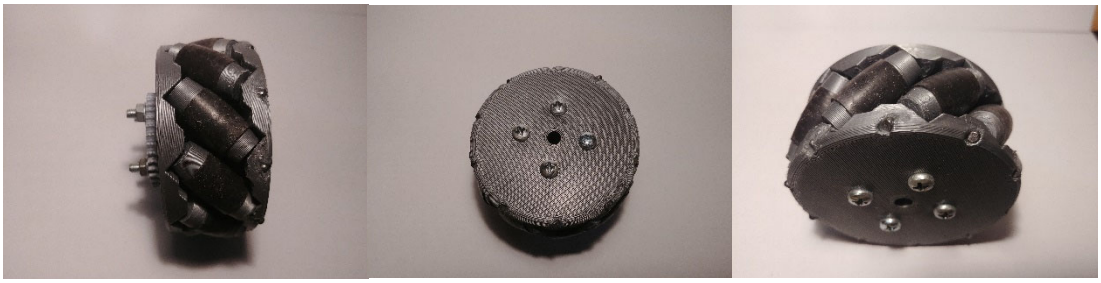


***Figura 45.*** *Rodes mecanum definitives. Elaboració pròpia*

### 3.1.4.2. Muntatge dels motors

El següent pas era unir els motors i les rodes i muntar-ho tot a la fusta. Per aconseguir el resultat final va ser necessari prendre varies decisions i realitzar moltes iteracions per tal d'aconseguir l'objectiu proposat.

Per poder transmetre el moviment es va haver d'afegir un engranatge tan als motors com a les rodes. La seva funció és evitar qualsevol dany al motor. Així doncs, si per qualsevol motiu la roda queda encallada, els engranatges saltaran i permetran que el motor continuï girant. Tot i que la idea inicial era comprar aquests engranatges, degut a la necessitat de crear una peça que encaixés amb el

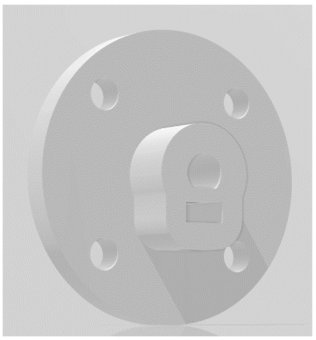


***Figura 46.*** *Shaft per l'engranatge. (Dejan, 2019)*

motor (veure **Figura 46**), es va haver d'optar per utilitzar la impressora 3D.Els dissenys de cadascuna de les peces es van treure d'internet perquè, d'aquesta manera, la feina es simplificava en la seva unió.

Escollir l'engranatge no va ser senzill ja que havia de complir uns requisits. El seu diàmetre havia de ser de màxim 25 mm si no es volia elevar els motors. Com que després de molta recerca només es va trobar un engranatge de 12.7 mm, es va decidir buscar-ne un de més gran i dur a terme l'elevació dels motors. Així doncs, l'elecció final va ser un engranatge de 36 dents i 38.1 mm de diàmetre (veure **Figura 47**).

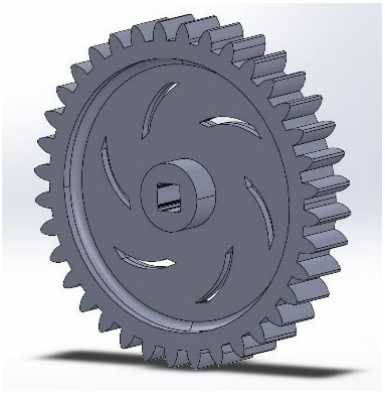


***Figura 47**. Engranatge de 36 dents i 38.1 mm de diàmetre. Elaboració pròpia*

Aquesta decisió va obligar a crear una nova peça que elevés el motor 1 cm. Tot i que amb la compra del motor ja venia inclòs un suport, aquest no era suficientment alt com per poder-lo utilitzar. Així doncs, per tal de fer-ho encaixar tot es va crear una peça com la de la **Figura 48**.

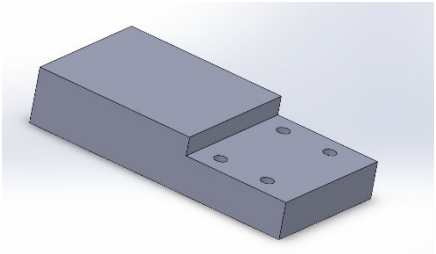


***Figura 48**. Suport pel motor. Elaboració pròpia*

Finalment es van obtenir dos engranatges diferents, un pel motor i l'altre per la roda. El primer constava del *shaft*, la peça que permetia encaixar-lo amb l'eix motriu (veure **Figura 49**). El segon però, es caracteritzava per un forat on hi cabia un rodament (veure **Figura 50**). És evident que per unir la roda amb el motor calia subjectar la roda d'alguna manera i que, per fer-la girar, s'havia d'utilitzar un eix. La funció del rodament doncs, és permetre que la roda giri sense importar el moviment de l'eix.

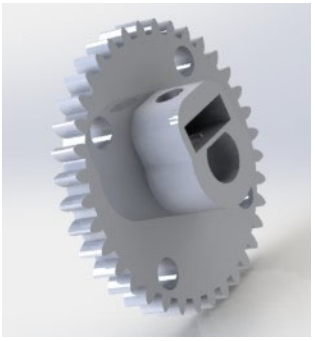


***Figura 49**. Engranatge amb shaft pel motor. Elaboració pròpia*

***Figura 50**. Engranatge amb forat per la roda. Elaboració pròpia*

### 3.1.4.3. Proves dels motors

Totes les peces dissenyades van necessitar moltes iteracions fins a convertir-se en les definitives. Hi havia un total de tres peces que suportaven el motor i la roda i que, alhora, els unien. Com que es muntaven a la fusta de forma independent, van portar molts problemes ja que calia una precisió mil·limètrica molt complicada d'aconseguir. Per una banda, si els engranatges quedaven massa junts, degut a la pressió causada, acabaven torçant-se i no girant bé. Al contrari, si quedaven massa separats, no es tocaven i per tant, la roda no girava.

En un primer moment, la idea de crear tres suports independents per dur a terme aquesta unió semblava bona. El suport de la roda estava format per dos peces idèntiques caracteritzades per un rodament que evitava la fricció amb l'eix (veure **Figura 51**). Però tot es va començar a complicar a l'hora d'enganxar-ho a la fusta. Com ja he dit anteriorment, es necessitava una precisió que les eines manuals no t'ofereixen. A falta d'una talladora làser, es van haver de realitzar diferents proves per veure si es podia aconseguir però, malauradament, es va acabar optant per buscar una altra manera per realitzar la unió.

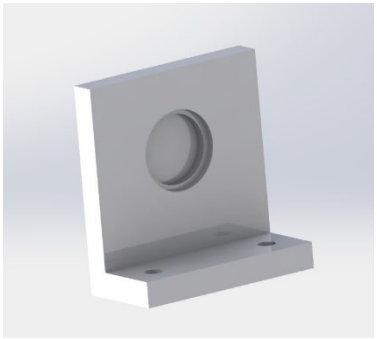


***Figura 51.** Suport per l'eix de la roda. Elaboració pròpia*

La solució consistia en ajuntar en el disseny 3D les dos peces que tanta problemàtica portaven. D'aquesta manera, la distància entre l'eix motriu i el de la roda no depenien dels forats realitzat per les eines, sinó d'un càlcul matemàtic invariable. Tot i que aquesta va ser la solució final, trobar la distància perfecta va suposar més de 10 iteracions en el disseny. A la **Figura 52** es pot contemplar el resultat final d'aquesta peça.

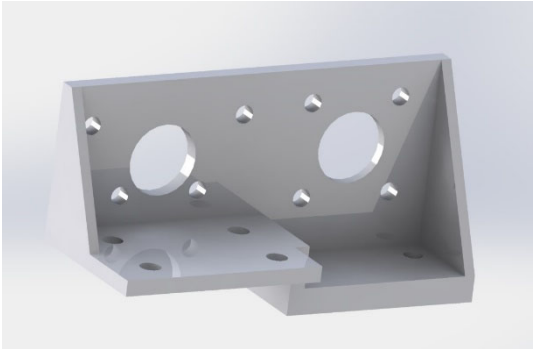


***Figura 52.*** *Suport definitiu pel motor i la roda. Elaboració pròpia*

D'aquesta manera, la unió dels motors és igual a la de la **Figura 53**.

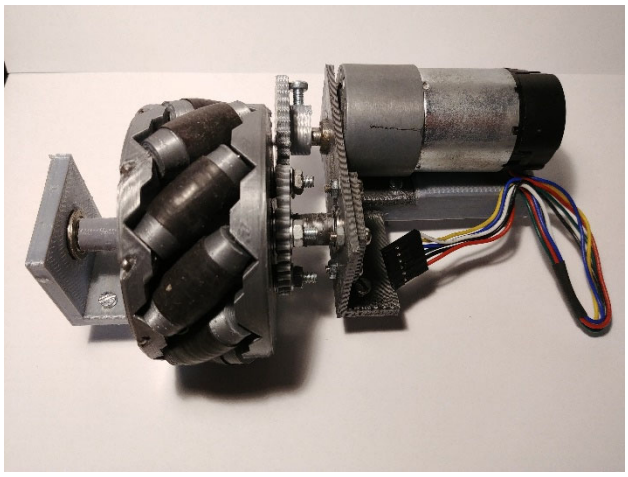


***Figura 53****. Muntatge definitiu dels motors.* Elaboració pròpia

### 3.1.5. Proves del drivetrain

Després d'haver muntat totes les peces a la base de fusta es va començar a fer les primeres proves de rendiment.

Per fer-ho, calia aprendre a programar un moviment desconegut fins al moment. Al ser les mecanum un tipus de roda tan especials, consten d'una configuració pròpia. Per aquest motiu, es va necessitar molta recerca per tenir clar el seu funcionament. Finalment, es va descobrir que en funció de la velocitat que se li dona a cada roda, el

robot es mou en una direcció diferent. A **Figura 54** es poden veure els exemples més senzills del moviment de les mecanum. Tal i com indica a la imatge, el moviment del robot dependrà de la potència que rebi cada roda.

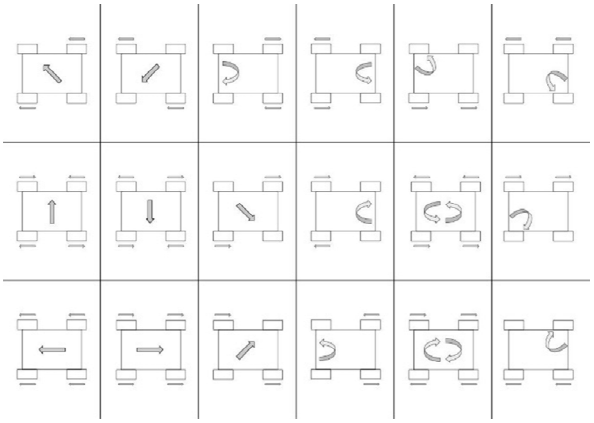


***Figura 54**. Esquema del moviment de les rodes mecanum.* Elaboració pròpia

Un cop entesa la configuració de velocitats que calia donar als motors, es va poder programar el robot perquè es mogués realitzant un quadrat. Al executar les proves, es va ser conscient d'un altre problema. El tipus de material que es va fer servir per imprimir les rodes, el PLA, era tan llis que patinava i, per tant, el robot no es movia. Per solucionar-ho, tal com es pot veure a la **Figura 55**, es van afegir tubs termoretràctils als corrons augmentant-ne així la tracció amb el terra.

***Figura 55.** Elaboració dels corrons amb el tub termoretràctil.* Elaboració pròpia

Després de desmuntar les rodes i tornar-les a assemblar, el robot va poder fer el moviment que se li havia demanat.

### 3.1.6. El sistema de subjecció del carro

El robot està pensat principalment per transportar pes i, per tant, calia crear una estanteria o qualsevol altre objecte per tal de poder transportar tot el material necessari. Com que els motors tenen un parell de bloqueig als 10 Kg, si es multiplica per quatre motors, surt que es pot arribar a suportar un total de 40 Kg i, per tant, una opció era instal·lar una estanteria damunt del robot.

***Figura 56**. Carro amb el robot encaixat. Elaboració pròpia*

Malgrat això, és una opció que es va acabar descartant perquè aportava poca flexibilitat al projecte. Tal com s'ha indicat en capítols anteriors, el fet d'arrossegar un carro feia que aquest es pogués adaptar a les necessitats del client i reduir els costos dels seus components. D'aquesta manera, es va començar a dissenyar el carro. Després de pensar-hi, la primera iteració va quedar com es pot veure a la **Figura 56** i el material que es va escollir va ser la fusta per la seva facilitat de manipulació. També es van estipular les mides del carro que serien de 500 x 550 x 750 mm.

Amb el carro dissenyat calia trobar com encaixar-lo amb el robot perquè el pogués arrossegar. El prototip final constava de dos perfils d'alumini en el carro que encaixaven amb dos carrils en el robot. Un cop els perfils eren dins els carrils, un sistema de biela-manovella activat per dos servomotors en el robot tancava el sistema. Per guanyar marge d'error, es va afegir un embut als carrils que ajudava a ressituar el carro en cas de necessitat (veure **Figura 57**). Tot aquest sistema està situat al segon pis del robot perquè els carrils no poden estar tapats.

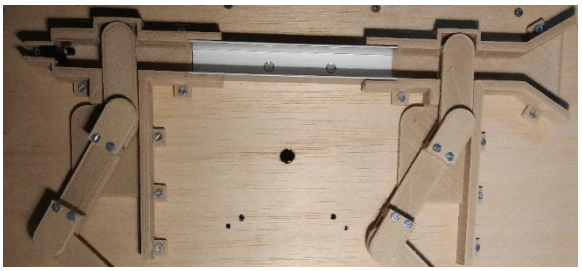


***Figura 57.*** *Carril de guia per on es fa la subjecció amb el carro.* Elaboració pròpia

Com que els carrils encaixen entre ells, el robot només caldrà que segueixi una línia i es col·loqui a sota el carro. Mentre vagi avançant, els carrils aniran coincidint i d'aquesta manera quedaran enganxats. En el moment que es vulgui deixar anar el carro caldria tornar a activar el servo perquè els sistemes de biela s'encongeixin i alliberin així el carro.

El sistema era funcional. Quan els perfils encaixaven dins els carrils feien contacte amb un polsador que indicava al robot que ja podia activar el sistema de biela-manovella. La mecanització del sistema mitjançant servomotors va resultar una manera senzilla i econòmica d'aconseguir els objectius. Per una banda, amb tan sols dos servomotors s'obtenia un moviment coordinat i segur per subjectar el carro al robot. Per altra banda, només amb peces 3D i rodaments s'aconseguia un sistema robust.

### 3.1.7. Control dels servomotors

Realitzar el moviment dels servos no era una feina extremadament complicada però si que calia indagar ja que era un tema completament desconegut. Després de molta recerca es va trobar que el servo funciona per impulsos i , per tant, a l'hora de programar no se li pot demanar que giri certs graus. En el cas del servo que s'ha utilitzat, els impulsos que rep són d'entre el 2.5% i el 12%. Per aquest motiu, per poder transformar els graus desitjats en impulsos calia seguir la següent fórmula:

$$impuls = \frac{graus}{18} + 2.5$$

### 3.1.8. Proves del sistema de subjecció

Abans d'arribar a la versió definitiva, van caldre certes iteracions de les quals en destaquen dos. Ambdues formen part de l'encaix del sistema de biela amb el carril. El primer problema va ser que el sistema de biela entrava tort i, per tant, era impossible que encaixés i que funcionés. Per solventar aquest problema es va decidir fer, tal com es pot veure a la **Figura 58**, unes parets laterals que servissin de guia pel sistema i així funcionés.

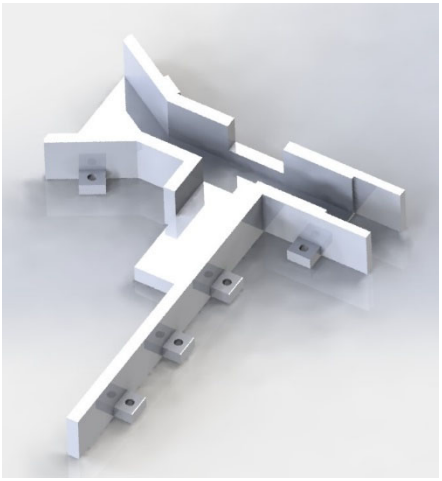


***Figura 58.*** *Embut amb parets.* Elaboració pròpia

L'altre problema era que el sistema de biela s'encallava a l'inici d'aquestes parets. Així doncs, es va crear una base que permetia reposar el sistema sobre seu i, per tant, evitar que quan avancés s'encallés.

El disseny de cadascuna de les peces que composen l'embut i els plànols queden reflectits a la **Figura 59**.

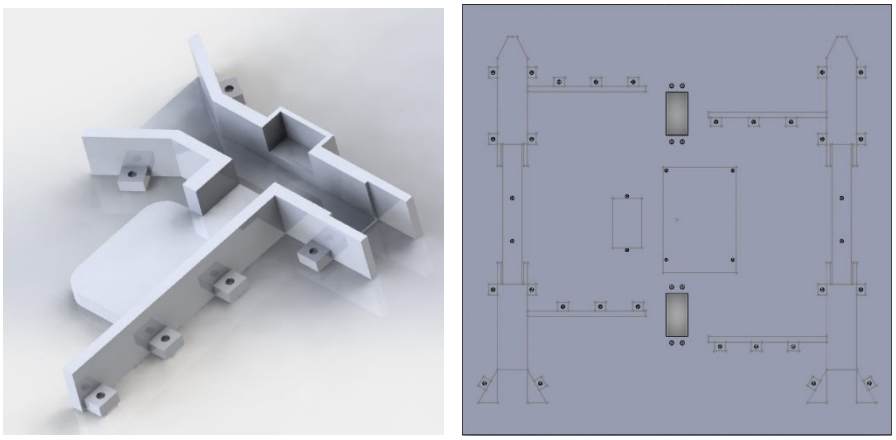


***Figura 59***. Embut amb la base més gran i plànols. *Elaboració pròpia*

### 3.1.9. El robot Rooby

Finalment, el robot Rooby queda de la següent manera. Si ens centrem primer en l'exterior i en el que es pot veure a simple vista, el robot té un total de dos pisos per poder aconseguir una millor distribució dels seus components (veure **Figura 60**). Les dues bases que hi ha són quadrades i de 40 centímetres de costat. A part de tenir en compte l'estètica, per tal d'evitar que el robot quedi encallat a les cantonades o a llocs petits, té les vores arrodonides amb un radi de 3 centímetres. D'aquesta manera, es guanya més autonomia impedint que hi hagi errors en el programa. Principalment està construït amb fusta perquè és un material fàcil de treballar i de pintar. Tot i així, quan ens centrem més en el seu interior, veurem com també s'utilitzen materials com el PLA i l'alumini. A la **Figura 60** es pot veure el procés de construcció de l'exterior del robot.

***Figura 60**. Procés de construcció d'en Rooby. Elaboració pròpia*

Tots els components que permeten el moviment del robot com son els motors, la bateria i les controladores estan situats al primer pis. Per tal d'obtenir una bona distribució dels components, calia tenir en compte tant la simetria com ocupar poc espai. D'aquesta manera s'assegurava que el pes quedava ben distribuït i que la base del robot no es xafaria. Així doncs, es va decidir posar la bateria al mig del robot ja que és l'element que més pesa i, al voltant, posar-hi els quatre motors amb les rodes. Entre els dos motors de davant i els dos de darrere hi ha col·locades les controladores i, el sensor de llum, està situat davant de tot per així poder programar un bon seguidor de línies. Un altre component essencial pel robot és la càmera. Es va decidir situar-la al pis de baix per, d'aquesta manera, aconseguir dissimular els codis QR deixant-los al marge de la paret. Aquesta decisió provocava dificultats a l'hora de realitzar la funció de seguiment

d'usuaris ja que, era impossible detectar a la gent. Així doncs, es va decidir enganxar la càmera a un servo per poder-la inclinar cap amunt solucionant així el problema i facilitant-ne el moviment. Per poder donar per acabat aquest pis, cal tenir present com s'uneix amb el segon. Al voltant de tota la base hi ha repartides un total de quatre peces que compleixen aquesta funció. A més, estan col·locades estratègicament per tal d'obtenir un robot compacte. A la **Figura 61** es pot contemplar el primer pis definitiu.

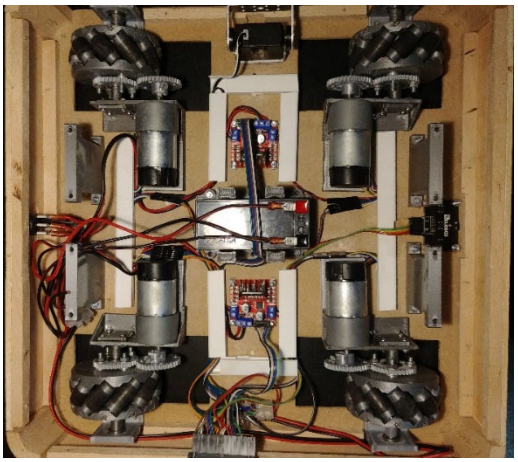


***Figura 61**. Imatge del primer pis del robot.* Elaboració pròpia

Al segon pis hi trobem l'element més important del robot, la Raspberry Pi. Sense ella, el robot no es podria moure ja que no es poden connectar els components enlloc. Juntament amb la Raspberry hi ha un conversor que transforma els 12 Volts de la bateria a 5 Volts per tal de poder-li donar potència per engegar-la. A més, aquest pis també conté el mecanisme que permet unir el robot amb els carros. Com ja he mencionat anteriorment, no serà el robot qui porti a sobre l'estanteria sinó que mourà un carro amb rodes reduint així el pes i augmentant el nombre de possibilitats de carros. Per aquest mecanisme tant el carro com el robot tenen dos carrils que encaixen entre ells. Els del robot consten d'un embut a l'inici que augmenta el marge d'error i d'un sensor de final de carrera al final. A la **Figura 62** es pot veure el resultat del segon pis.

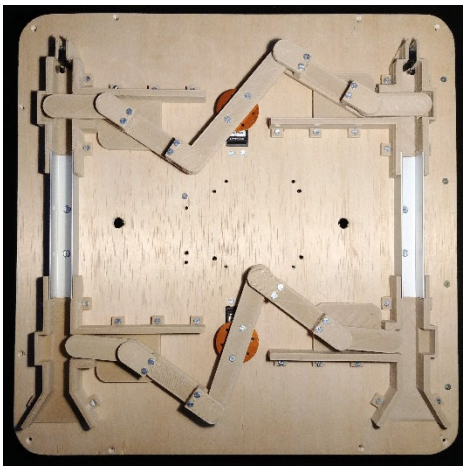


***Figura 62.*** *Imatge del segon pis del robot.* Elaboració pròpia

## 3.2. Disseny del programari

Una vegada el robot ja estava muntat i llest per fer-lo moure, calia pensar en les necessitats de programació. Per començar, s'havia de decidir què faria el robot i com ho faria. Hi havia un total de dos necessitats a cobrir: el mode autònom i el mode de seguiments d'usuaris. El primer objectiu es basa en seguir una línia fins que el robot detecta el codi QR demanat. El mode de seguiment d'usuaris és seguir els docents. Aquests portaran un objecte que el robot haurà d'identificar i seguir-lo.

Quan les necessitats havien quedat clares, calia trobar la manera de connectar-se amb la Raspberry Pi i donar-li les ordres. La idea inicial era crear una APP pròpia que es connectés remotament amb la Raspberry Pi. Finalment però, degut a la falta de temps es va prendre la decisió d'utilitzar una APP ja existent creada per Ettore Gallina i anomenada RaspController que permet controlar els diferents elements de la Raspberry Pi a partir d'una connexió remota. Tot i no utilitzar una APP pròpia, s'utilitza una que compleix els requisits inicials.

A continuació s'expliquen tots els passos seguits per arribar a programar en Rooby.

### 3.2.1. Especificacions de casos d'ús

Abans de començar a programar, l'Enginyeria del Software ens indica que cal pensar bé totes les necessitats i fer un estudi dels casos d'ús que es volen implementar. Per aquest motiu, a continuació es poden veure aquests requisits en forma de taula de casos d'ús.

**Taula 12 –** Document de casos d'ús - Inicialització

| **Document de casos d'ús - Inicialització** | |
|---|---|
| **Nom** | Inicialització |
| **Actors** | Usuari final |
| **Descripció** | L'usuari engega el robot per poder iniciar el programa. |
| **Precondicions** | El robot:<br>• Té bateria.<br>• Té la càmera accessible i connectada.<br>• Té tots els sensors i actuadors connectats. |
| **Postcondicions** | El robot s'ha engegat i configurat correctament. |
| **Curs normal** | 1. L'usuari engega el robot.<br>2. S'engega el programa.<br>3. Es configuren les entrades i sortides.<br>4. Comprovar si està connectat a la xarxa WiFi.<br>4.1. Es connecta a la xarxa.<br>5. Comprovar si la càmera funciona.<br>5.1. S'engega la càmera. |
| **Excepcions** | 4.1. (E1) No es pot connectar a la xarxa. Error.<br>5.1. (E2) No es pot engegar la càmera. Error. |
| **Prioritat** | Màxima. És la base per engegar-ho tot. |
| **Freqüència d'ús** | Un cop cada vegada que s'engegui el robot. |
| **Requisits especials** | Cap. |
| **Suposicions de partida** | El robot comença en una superfície plana. |

Font: Elaboració pròpia

**Taula 13 –** Document de casos d'ús – Mode autònom

| Document de casos d'ús - Mode autònom | |
|---|---|
| **Nom** | Mode autònom |
| **Actors** | Usuari final |
| **Descripció** | L'usuari li indica al robot quin és el lloc al que ha d'arribar. El robot segueix una línia i llegeix codis QR fins que troba el que coincideix amb el destí desitjat. Finalment para motors. |
| **Precondicions** | • El robot té bateria.<br>• Disposa d'accés a la xarxa WiFi.<br>• El sensor de llum ha d'estar connectat a la Raspberry Pi.<br>• La càmera ha de funcionar i ha d'estar connectada.<br>• Els motors DC han d'estar connectats a la controladora L298n.<br>• El robot ha d'estar sobre una línia.<br>• L'usuari ha de donar una petició vàlida. |
| **Postcondicions** | El robot ha arribat al destí proposat. |
| **Curs normal** | 1. L'usuari a través de la aplicació indica el destí del robot.<br>2. El robot rep el missatge.<br>3. S'engeguen els motors.<br>4. El robot comença a seguir la línia mentre busca QR.<br>5. El robot detecta el codi QR coincident amb el missatge.<br>6. El robot arriba al seu destí i para motors. |
| **Excepcions** | 3.1 (E1) Hi ha errors amb els motors DC.<br>4.1 (E2) El robot perd la línia i para els motors.<br>4.2 (E3) Hi ha errors amb el sensor de llum.<br>4.3 (E4) Hi ha errors amb la càmera. |
| **Prioritat** | Juntament amb el seguidor d'usuaris, és el programa primordial que permet el moviment del robot. |
| **Freqüència d'ús** | Sovint. |
| **Requisits especials** | • Hi ha d'haver una línia negra que marqui el recorregut.<br>• La línia ha de ser de mínim 47 mm de gruix.<br>• Hi ha d'haver els codis QR al costat de les línies. |
| **Suposicions de partida** | El robot comença en una superfície plana. |

Font: Elaboració pròpia

**Taula 14 –** Document de casos d'ús – Mode de seguiment d'usuaris

| **Document de casos d'ús - Mode de seguiment d'usuaris** | |
|---|---|
| **Nom** | Mode de seguiment d'usuaris |
| **Actors** | Usuari final |
| **Descripció** | L'usuari li indica al robot que l'ha de seguir. Per fer-ho, durà un objecte de color groc. S'acabarà el programa quan l'usuari li indiqui a través de la aplicació o quan el robot no trobi l'objecte durant més de 5s. |
| **Precondicions** | El robot:<br>• Té bateria.<br>• Disposa d'accés a la xarxa WiFi.<br>• Té la càmera accessible i connectada.<br>• Té tots els sensors connectats.<br>• Els motors DC han d'estar connectats a la controladora L298n.<br>• El robot és capaç de veure l'objecte. |
| **Postcondicions** | El robot ha seguit l'usuari fins a finalitzar el programa. |
| **Curs normal** | 1. L'usuari a través de la aplicació indica el destí del robot.<br>2. Fins a rebre missatge d'aturar-se o no trobar l'objecte durant més de 5s:<br>2.1. El robot busca l'objecte.<br>2.2. Calcula les coordenades on està.<br>2.3. Calcula el moviment a realitzar. |
| **Excepcions** | 3.1 (E1) Es pararà el robot esperant a tornar a detectar l'objecte. |
| **Prioritat** | Juntament amb el mode autònom, és el programa primordial que permet el moviment del robot. |
| **Freqüència d'ús** | Sovint. |
| **Requisits especials** | • L'objecte ha de ser de color groc i esfèric d'un diàmetre aproximat de 5cm. |
| **Suposicions de partida** | El robot comença en una superfície plana. |

Font: Elaboració pròpia

### 3.2.2. El diagrama de flux del programa principal

En la **Figura 63** es pot interpretar el diagrama de flux que es va seguir per al programa principal del robot. En ella s'hi poden observar moltes de les decisions que es van prendre i que es trobaran explicades en els apartats següents.

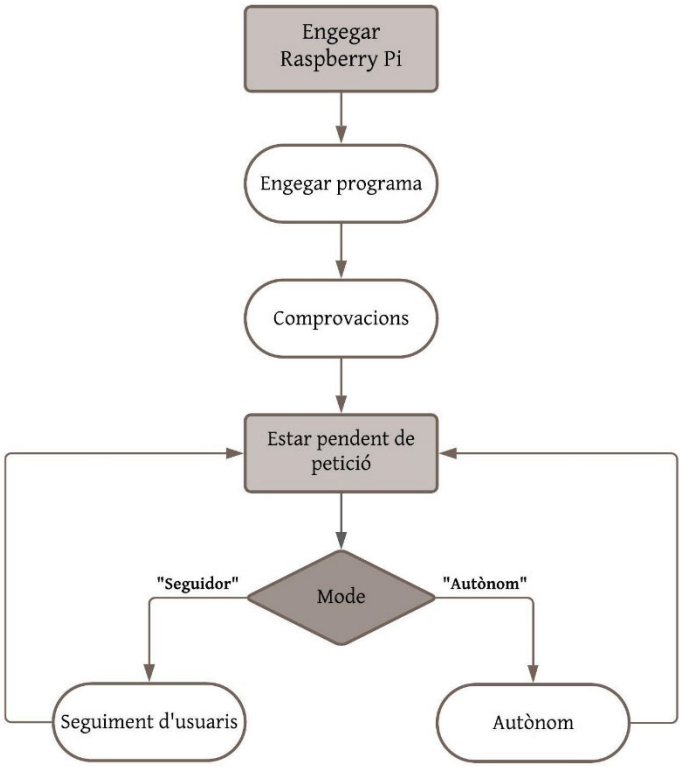



***Figura 63**. Diagrama de flux del programa principal. Elaboració pròpia*

### 3.2.3. Mode autònom

El mode autònom té com a objectiu fer arribar el robot a un lloc desitjat. Per tal d'aconseguir-ho, es fa ús de la càmera i del sensor de llum. Inicialment, l'usuari li envia un missatge al robot que li indica quin és el seu destí. Una vegada ha interpretat aquest missatge, comença a seguir la línia i va llegint tots els codis QR que es troba pel camí. A la que el text que hi ha escrit en el codi QR coincideix amb el missatge que li ha enviat l'usuari, el robot para els motors en senyal de que ja ha arribat al lloc desitjat. A la **Figura 64** es pot seguir el procediment explicat.

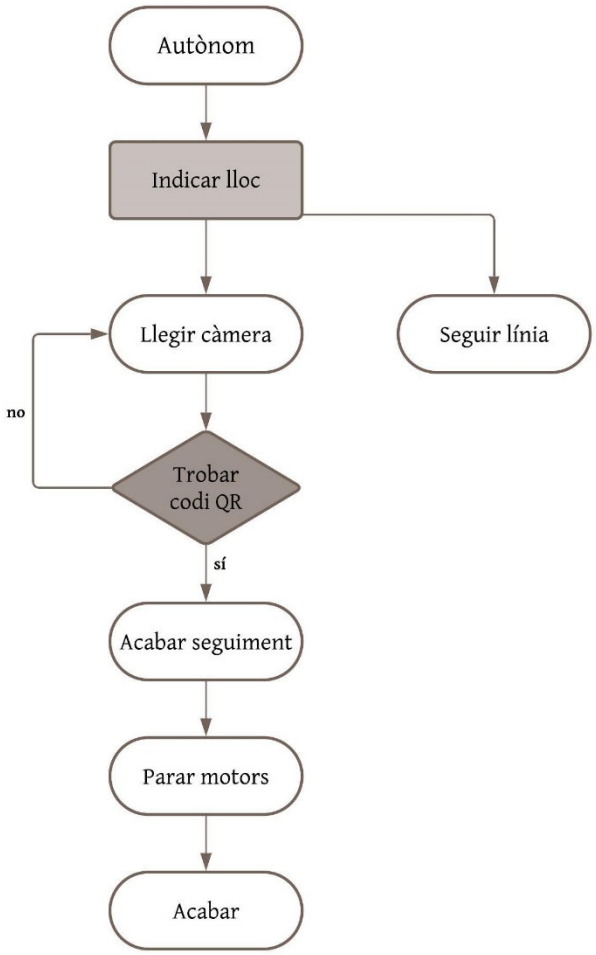



***Figura 64**. Diagrama de flux del mode autònom. Elaboració pròpia*

Aquest mode consta de la unió de dues funcions essencials, el detector de codis QR i el seguidor de línies. A continuació s'expliquen els processos de disseny de cadascun.

### 3.2.3.1. Detector de codis QR

La primera funció és el detector de codis QR que permet saber al robot quan ha arribat al seu destí. Per poder dur a terme aquest programa, es va haver de fer recerca sobre localització i lectures de codis QR i es va veure que, per tal de realitzar una lectura adequada, primer calia saber interpretar-los. Realitzar una funció pròpia que descodifiqués un per un tots els codis que s'anessin detectant era una feina feixuga. Així doncs, es va decidir fer ús de la llibreria *pyzbar* que desxifra tots els codis QR i en permet la seva lectura.

Però no es podia fer la prova de funcionament si encara no s'havia programat la detecció dels codis QR. Per dur a terme aquest procés, només calia encendre la càmera i llegir totes les imatges cercant-los. Després d'haver fet recerca es va trobar la llibreria VideoCapture que facilita totes les necessitats. Després d'entendre el seu funcionament, ja es tenia el primer programa creat.

#### 3.2.3.1.1. Les primeres proves massa lentes

Els veritables problemes van començar al llarg de la realització de les primeres proves. El programa funcionava bé i es descodificaven exitosament els codis QR però el problema era que moltes vegades el robot passava de llarg i no els detectava. Es van fer diverses proves com mantenir la càmera davant del codi QR o baixant la velocitat del robot però, tot i així, seguia fallant massa vegades.

Es va haver de buscar molta informació de la llibreria VideoCapture i es va intentar millorar la qualitat del programa intentant agilitzar el procés però, tot i així, seguia fallant. Finalment, es va descobrir que aquesta llibreria era molt lenta a l'hora de realitzar la lectura d'imatges perquè les llegeix a mida que les va detectant. Això provoca que no detecti moltes imatges perquè en aquell moment s'està realitzant una altra tasca. És per aquest motiu que es va decidir calcular quina era la quantitat de FPS (Fotogrames Per Segon) que es detectaven.

Per poder fer aquesta comprovació calia crear una funció de càlcul de FPS. El funcionament de la funció es basava inicialment en mirar quina era l'hora local i es contaven quantes imatges anava detectant la càmera. Finalment, quan el programa finalitzava es tornava a mirar l'hora local per poder calcular quin havia sigut el temps transcorregut. Per tal de trobar els FPS, es dividien la quantitat d'imatges trobades entre el temps. Després d'algunes proves, es va observar que es detectaven massa pocs fotogrames per segon. Normalment, la mitjana es troba entre els 24 i els 30 fotogrames per segon, dada que calia ampliar si es volia assegurar la identificació dels codis QR. Amb l'ús de la llibreria VideoCapture es van detectar una mitjana d'entre 15 i 20 FPS. Era

un resultat preocupant perquè la intenció era superar la mitjana i el resultat obtingut era molt més baix. Aquí es va veure la necessitat de buscar una alternativa.

#### 3.2.3.1.2. La llibreria VideoStream

La llibreria VideoStream és semblant a la VideoCapture amb la diferència de que internament va creant processos en paral·lel que van llegint la imatge i guardant-la en memòria descodificades. Així doncs, es va modificar el programa per poder fer ús de la llibreria VideoStream i es van realitzar les mateixes proves que l'anterior vegada. Aquest cop però, es va començar amb la comprovació dels FPS. En aquest cas, la diferència es notava molt ja que es va obtenir un resultat d'entre 35 i 45 fotogrames per segon doblant l'anterior resultat. Veient que el resultat era el desitjat, es va decidir fer les proves amb el robot i es va poder observar que, efectivament, aquest codi sí que funcionava correctament.

### 3.2.3.2. El seguidor de línies

Una vegada es va aconseguir detectar i llegir els codis QR es va haver de realitzar la segona funció essencial: el seguidor de línies. La idea inicial era realitzar un PID (Proporcional Integral Derivatiu) per dur a terme la realització d'aquesta part. Amb aquest codi, el sensor de llum era capaç de detectar a quina part de la línia es trobava ja que feia una aproximació dels valors màxims i mínims. D'aquesta manera, quan detectava que s'estava allunyant del color negre, modificava lleugerament la seva posició realitzant un moviment suau i aconseguint que semblés un moviment recte.

Malauradament, els sensor del que es disposava era digital i, per tant, només detectava el blanc i el negre. Com que no era capaç de trobar valors intermedis es van buscar vàries opcions. Una d'elles era transformar la sortida digital a analògica fent ús d'un ADS-15. El problema que va sorgir era que no hi havia prou pins lliures a la Raspberry Pi per poder-hi afegir més elements. Així doncs, es va haver de buscar una solució fent ús dels sensors digitals.

#### 3.2.3.2.1. Mètode de seguiment de la línia amb sensors digitals

Com que la placa constava d'un total de tres sensors, tot i que no es podia aconseguir un moviment amb PID, sí que es podia simular. Els sensors laterals havien de servir per detectar quan el robot s'estava allunyant del negre i, el del mig, detectaria la desviació més pronunciada. D'aquesta manera, es podia modificar la direcció amb prou marge de temps perquè el moviment del robot semblés que fos recte. Per fer-ho, es va dibuixar un esquema com el de la **Figura 65** per tal de veure quins eren els possibles casos amb els que es podia trobar el robot.

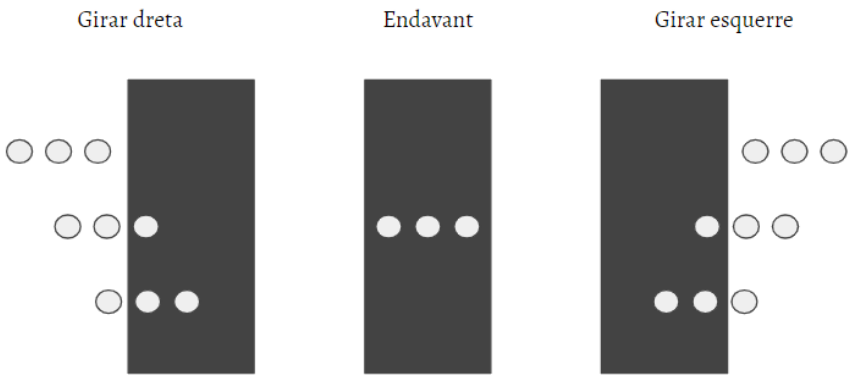



***Figura 65**. Esquema dels possibles casos amb els que es pot trobar el robot.* Elaboració pròpia

Excepte en el cas que els tres sensor detectin el color negre indicant que el robot es mourà recte, la resta dels casos el robot ha de corregir la posició cap al costat contrari al que s'està desviant. Si pel motiu que sigui el robot perd la línia es calcula quin ha sigut l'últim moviment abans de perdre-la i es modifica la direcció del robot tenint en compte una d'aquestes tres possibilitats:

- Moviment recte: Indica que la línia s'ha acabat i, per tant, s'han de parar els motors i donar el programa per acabat.
- Moviment lateral cap a la dreta: Indica que la línia es troba al costat esquerre. Per aquest motiu, el robot es mou en direcció contraria fins que la torna a trobar. En el cas que no es torni a trobar la línia, el robot pararà els motors i es mostrarà un missatge d'error.

- Moviment lateral cap a l'esquerra: Igual que l'anterior moviment, el robot es mourà en direcció contrària a la que anava ja que la línia queda a l'altre costat. Si tampoc es troba la línia, també es paren els motors i s'envia el missatge d'error.

Així doncs, després de tenir tots els moviments del robot controlats, només calia triar bé la mida de la cinta negra per tal d'ajudar a obtenir un moviment més suau i rectilini. Tenint en compte que la mida de la placa és de 50 mm i que els sensors de llum es troben a una distància de 5 mm entre ells, es va decidir agafar una cinta de 47 mm. A la **Figura 66** es pot veure el robot seguint la línia.

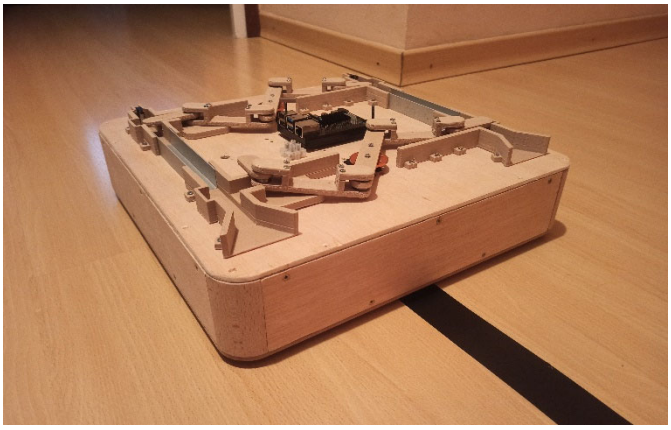


***Figura 66**. Rooby seguint la línia. Elaboració pròpia*

### 3.2.3.3. El mode autònom al complet: QR i seguiment de línies

L'últim pas per acabar la realització d'aquest programa era unificar les dues funcions. El problema que hi havia era que no s'havien d'executar de manera seqüencial sinó que ho havien de fer en paral·lel fins que es trobés un codi QR. Això va suposar un repte ja que s'havia de fer ús d'un fil d'execució (*thread* a partir d'ara) i mai abans s'havia treballat amb aquest concepte. Un *thread* permet executar una tasca en segon pla o el que és el mateix, sense bloquejar el programa mentre s'està executant. Per poder-lo utilitzar van fer falta unes pràctiques anteriors a tot el procés final de programació. Calia entendre les diferents funcions de la llibreria per poder fer funcionar el programa.

#### 3.2.3.3.1. Els threads

Primer de tot, calia entendre que cada fil d'execució necessitava executar una funció. D'aquesta manera, quan es crea un *thread* cal dir-li què estarà executant. Un cop creat, un fil es pot engegar (*start* a partir d'ara), tornar a la branca principal (*join* a partir d'ara) i avortar-lo (*terminate* a partir d'ara).

```
1. t1.start()
2. t2.start()
3. t1.join()
4. t2.terminate()
```

En aquest exemple, s'engeguen dos fils anomenats t1 i t2. Amb la funció t1.join() el que fa és esperar que s'acabi el procés t1 per continuar. Un cop acabat, amb t2.terminate() s'avorta el procés t2.

Aquest era l'esquema a seguir en el cas del mode autònom. Per una banda, es creaven dos fils: un per a la detecció de codis QR (t1) i l'altra per al seguiment (t2). Tant un com l'altre s'engegaven i, per tal de saber quan calia acabar el mode autònom es feia un `join` de la funció de detecció de codis QR. Un cop trobar el codi, el programa continuaria la seva execució i faria un `terminate` del fil del seguidor de línia per aturar els motors donant per finalitzat el programa.

### 3.2.3.4. Proves del mode autònom

Amb el programa del mode autònom acabat es van procedir a realitzar les diferents proves. El fet de canviar la llibreria VideoCapture per VideoStream va permetre que el robot no passés de llarg mentre buscava codis QR. El funcionament del seguidor de línies també era correcte i el robot va ser capaç de recórrer tot un passadís fins arribar al lloc sol·licitat en les diferents proves que es van fer.

### 3.2.4. Mode de seguiment d'usuaris

L'altre mode que calia programar era el seguiment d'usuaris mitjançant les tècniques de visió per computador estudiades al Capítol 3.4 del marc teòric.

Per fer-ho, es fa ús de la càmera i un objecte en forma de clauer que els usuaris es situaran a la mà. Com a objecte es va decidir utilitzar una pilota de tenis de color groc perquè l'esfera és un volum geomètric que no canvia en cap de les seves perspectives i el color groc és un color poc comú i, per tant, ressalta per sobre la resta. Tot i que la idea principal era seguir als propis treballadors es va veure que en un passadís hi circula molta gent i que, avui en dia anem tots vestits tant iguals que el robot es podria confondre fàcilment. Així doncs, es va prendre la decisió que el robot detectés la pilota i la seguís fins que l'usuari li enviés un missatge conforme ja s'havia acabat el programa.

A l'inici d'aquesta part del projecte es va fer molta recerca per poder determinar quines eren les necessitats del programa. Després d'haver trobat diferents tutorials de seguiment d'objectes, es va veure que per tenir un programa de qualitat, calia dividir-lo en tres parts: buscar l'objecte, seguir-lo en el vídeo i transformar les coordenades 2D al moviment del robot. A la **Figura 67** es pot seguir el procediment explicat.

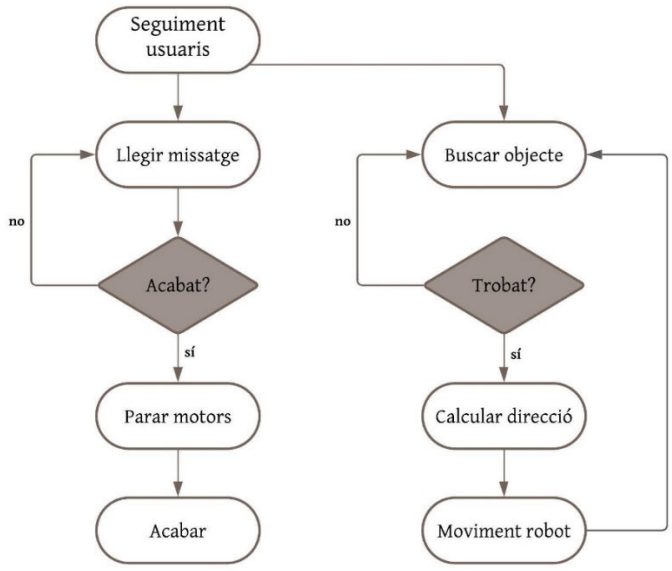



***Figura 67**. Diagrama de flux del mode de seguiment d'usuari. Elaboració pròpia*

### 3.2.4.1. Buscar l'objecte

El primer repte constava en detectar l'objecte proposat. Per aconseguir-ho, es van utilitzar tots els conceptes apresos al llarg del curs de OpenCV. Tot i així, encara que per aquesta primera part de programació ja es tinguessin alguns coneixements, calia realitzar moltes proves i aplicar un procés de visió artificial adequat. Per tal de detectar l'objecte es va decidir centrar-se en dos de les seves característiques per, si en fallava una, poder seguir fent el seguiment amb l'altre. Així doncs, els trets amb els que es va treballar van ser la forma i el color.

Com que es poden trobar moltes circumferències a l'entorn però, en canvi, no és gaire habitual el color groc, es va decidir que inicialment el robot buscaria el color i, una vegada trobat, es detectaria l'objecte segons la seva forma. D'aquesta manera, hi hauria moltes més possibilitats de localitzar l'objecte desitjat.

Per tal de detectar el color, es va fer ús de diferents funcions de la llibreria OpenCV (veure **Figura 68**). L'objectiu final d'aquest procés era crear una màscara on només hi aparegués la pilota a seguir. A partir de l'espai de color HSV, es van poder realitzar tots els passos necessaris per aconseguir l'objectiu.

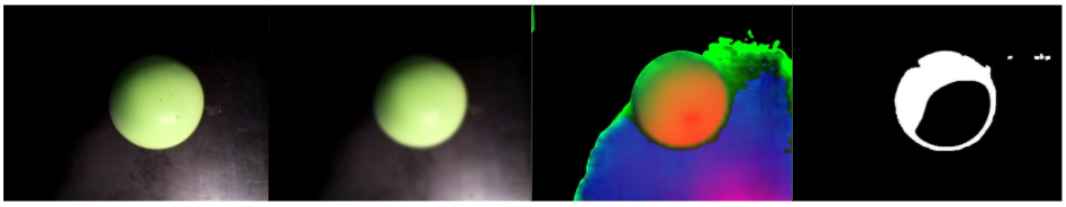


***Figura 68.*** *Procés de detecció de l'objecte. D'esquerre a dreta: BGR, Gaussian Blur, HSV, Màscara. Elaboració pròpia*

Una vegada obtinguda la màscara calia conèixer la forma de l'objecte. Per fer-ho, a través de diferents funcions que també formaven part de la llibreria OpenCV, es dibuixava un cercle al voltant de l'objecte, es trobava la seva àrea i es contaven els vèrtexs que tenia. Si la figura tenia més de 8 vèrtexs significava que era de forma cilíndrica i que, per tant, es podia passar a la següent condició per assegurar-se que l'objecte detectat era el desitjat. Perquè es complís la següent condició, el radi de la figura havia d'estar entre 10 i 35 mm.

Si era així, s'havia detectat la pilota de tenis però, en el cas de que alguna d'aquestes condicions fallés, es descartava l'objecte ja que era impossible que fos la pilota.

#### 3.2.4.2. Seguiment de l'objecte

Després d'haver aconseguit detectar l'objecte i tenir la primera part del programa resolta, s'havia de realitzar el seguiment de l'objecte amb la càmera. Els objectius d'aquesta part del programa eren:

- Detectar una sola vegada l'objecte i ser capaç de traçar-ne el seu moviment
- Ser capaç de gestionar quan l'objecte desapareix o surt dels marges del vídeo
- No perdre l'objecte quan és tapat
- En el cas de que es perdi l'objecte, tornar-lo a trobar

Una vegada clars els objectius calia indagar una mica en el tema. Un element clau pel seguiment d'objectes és el càlcul del centroide entès com el centre del conjunt de píxels de la figura detectada. A la **Figura 69** es pot apreciar el concepte.

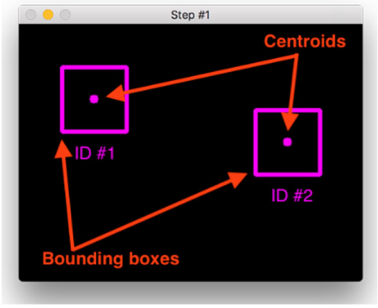



***Figura 69.*** *Detecció dels centroides de dues figures. (Rosebrock, 2018)*

Quan s'havia calculat, no era necessari detectar contínuament l'objecte perquè ja es tenia guardat un punt que el determinava i en simulava el seu moviment. Així doncs, guardant les posicions del centroide en una taula es podia realitzar el seguiment de l'objecte amb la càmera d'una manera bàsica però eficaç.

#### 3.2.4.3. Moviment del robot

Amb els centroides trobats, teníem la posició de l'objecte d'un món 3D en una representació en el pla. Això implicava haver de trobar en quina direcció calia moure el

robot per anar fent el seguiment. El mètode utilitzat per fer la conversió de coordenades 2D a la direcció que s'havia de moure el robot era bàsic però efectiu.

Per tal de seguir a l'usuari es va decidir que el robot utilitzaria tres tipus de moviments: el gir (esquerre i dret), el moviment diagonal (esquerre i dret) i el moviment recte. Per saber quin era el tipus de moviment que havia de realitzar es va dividir la imatge en 5 parts diferents, una per cada moviment. El càlcul era tan senzill com fer una divisió però, per obtenir un programa universal, es va crear fent ús de variables i de tant per cents. D'aquesta manera, es podien obtenir les 5 franges utilitzant un requadre de mides qualssevol.

El següent pas era relacionar l'objecte amb les franges. Per fer-ho es va fer ús de les coordenades del centroide, exactament les de l'eix de les abscisses. Es va decidir no utilitzar l'eix de les coordenades perquè no era important conèixer a quina alçada estava col·locat l'objecte. Així doncs, el robot realitzava un moviment diferent per cadascuna de les franges on es trobava el centroide. Per exemple, si l'objecte es detectava a la franja del mig, el robot es movia endavant. A la **Figura 70** es pot observar la divisió de la imatge en les 5 franges diferents i, a la **Figura 71**, la detecció de la pilota i del moviment a partir d'aquestes franges.

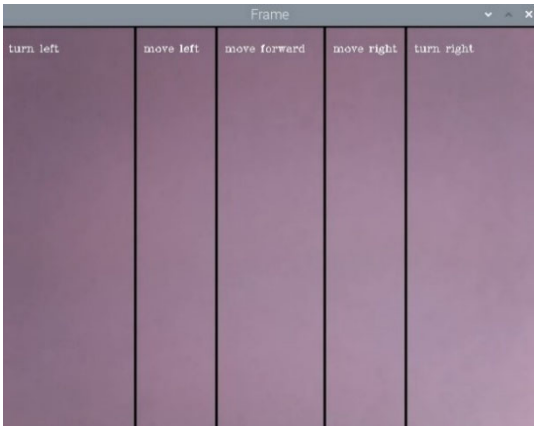



***Figura 70**. Divisió de la imatge en 5 franges. Elaboració pròpia*

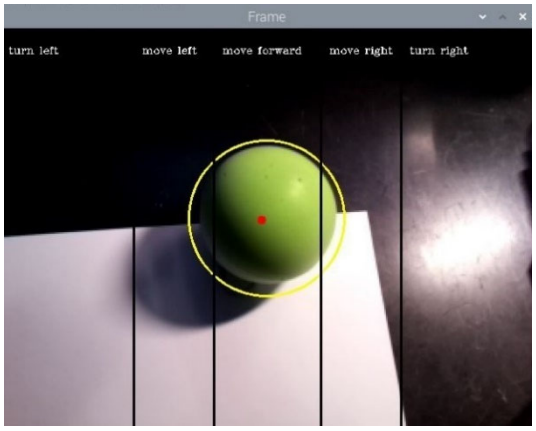



***Figura 71**. Detecció de la pilota i del moviment del robot segons la posició del centroide. Elaboració pròpia*

D'aquesta manera, gràcies a la llibreria de OpenCV per la detecció de l'objecte, al càlcul del centroide i de la distància Euclidiana i a l'ús de les franges pel moviment del robot, es va aconseguir realitzar un seguiment d'usuaris fiable i útil.

#### 3.2.4.4. Proves de seguiment

El primer que es va detectar amb les proves va ser que les condicions de llum afectaven al funcionament del programa. Aquest problema s'ajuntava amb la qualitat amb què la càmera gravava. Al ser una marca menys coneguda i de baix cost, la resolució no era tan bona i captava les imatges fosques. Per tant, si ho combinàvem amb un escenari amb llum artificial, el pla quedava encara més fosc i era gairebé impossible detectar l'objecte

Després de fer recerca es va veure que un dels problemes podia ser el ISO (International Organization for Standardization) que venia a ser la sensibilitat del sensor a l'hora de captar la llum. La càmera constava d'un ISO molt baix i, per tant, només calia ampliar-lo per tal d'obtenir imatges més clares. Es va trobar una llibreria que permetia canviar la configuració de la càmera però, només era funcional en les fotografies. En el meu cas, necessitava gravar un vídeo i analitzar les imatges una a una i, per tant, aquella llibreria no em servia. Així doncs, encara que vaig obtenir els resultats desitjats (veure **Figura 72**), vaig haver de buscar una altra solució per tal de poder millorar el programa.

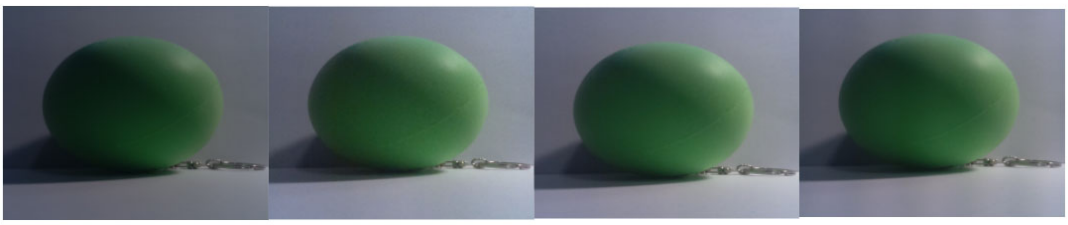


***Figura 72**. Comparació dels diferents resultats de la modificació del ISO. D'esquerra a dreta: 100, 200, 400 i 800. Elaboració pròpia*

Finalment, em vaig decantar per la compra d'una nova càmera capaç de captar imatges clares. Vaig decidir buscar les característiques de totes les càmeres de la marca Raspberry Pi (veure **Figura 73**) perquè, encara que fossin una mica més cares, valia la pena si m'ajudava a fer el programa funcional.


***Figura 73**. Comparació de les imatges capturades per totes les possibles càmeres a comprar. (Cholewiak, 2017)*

En la imatge es pot veure que la Raspberry Pi Camera i la Arducam 5MP RPi Camera eren les que captaven imatges més clares. Tot i així, la primera ja no estava a la venta perquè és molt vella i la segona només serveix per la Raspberry Pi Zero, és a dir, no la podia utilitzar. Per tant, només em quedava una possibilitat, la Raspberry Pi v2 Camera, que tot i ser més fosca que les altres, era de molt bona qualitat.

Una vegada solucionat aquest problema, es va detectar que tot i així costava detectar la pilota. Es va arribar a la conclusió que, com que la pilota era una mica verdosa, depenent del contrast de llum que rebia, el seu color variava. Això era impossible d'arreglar a través de la programació ja que no es podien anar modificant els llindars de colors cada vegada que es canviava d'escenari.

La solució plantejada va ser canviar la pilota per una que fos d'un color cridaner i que no pogués variar. Així doncs, es va decantar per una pilota de color verd croma assegurant-se que el color no variaria. D'aquesta manera el problema es va solucionar i, com que el llindar de l'espai de color HSV es podia ampliar, hi havia moltes més possibilitats de detectar la pilota sempre. A la **Figura 74** es pot veure el procés de seguiment d'usuaris.

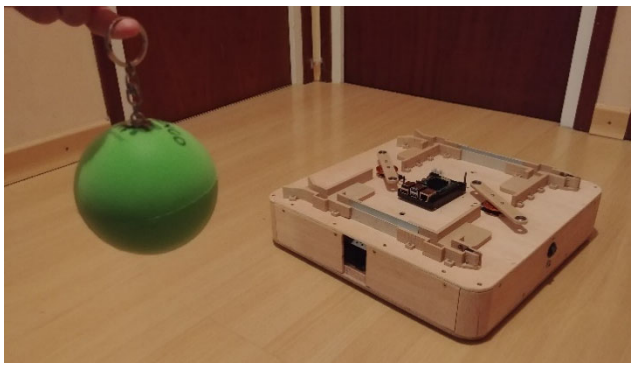


***Figura 74**. Procés de seguiment d'usuaris. Elaboració pròpia*

## 3.3. Proves del robot Rooby

Una vegada realitzades totes les proves necessàries i obtinguda la versió definitiva de cada programa, es va posar a prova el robot. Sotmetent-lo a diferents escenaris es va voler comprovar tant la seva fiabilitat com la capacitat d'autonomia.

Primer de tot es va comprovar si el robot era capaç de col·locar-se sota el carro i activar els mecanismes per tal de subjectar-lo (veure **Figura 75**). L'escenari constava d'una línia al terra amb el robot a sobre. Tot va funcionar de seguida ja que l'embut permetia reconduir el robot per tal que entrés adequadament. Així doncs, el robot ja havia superat la primera prova.

***Figura 75**. Comprovació del mecanisme de subjecció pel carro. Elaboració pròpia*

Tot seguit es va decidir comprovar la part autònoma. Cada vegada que s'engegava el programa, el robot estava a un punt d'inici diferent. Tot i així, enviant-li com a objectiu diferents aules, va aconseguir arribar-hi totes les vegades. Per tal de comprovar el marge d'error però, es va construir un escenari erroni conscientment. Finalment, el robot va aconseguir passar la prova complint cada vegada el seu objectiu o, si més no, informant que hi havia un error. A la **Figura 76** es pot veure un dels escenaris pel que va haver de passar el robot.

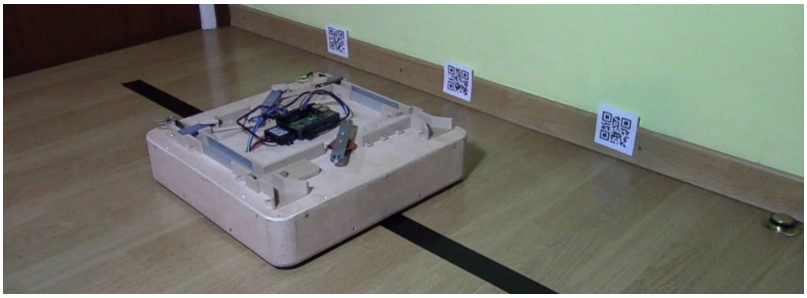


***Figura 76**. Comprovació de la programació autònoma. Elaboració pròpia*

Finalment es va dur a terme la comprovació del seguiment d'usuaris. En aquest cas l'objectiu era aconseguir que el robot seguí la pilota durant més d'un minut. Igual que les anteriors comprovacions, aquesta també va funcionar. Gràcies al mètode de divisió d'imatge el robot era capaç de seguir usuaris sense perdre'ls en cap moment. A la **Figura 77** es pot veure com el robot va complir el seu objectiu.

***Figura 77**. Comprovació de la programació del seguiment d'usuaris. Elaboració pròpia*

## 3.4. Estudi de viabilitat

Un dels objectius del treball era crear un robot de baix cost per aquelles petites empreses, escoles o hospitals que no es poguessin permetre un robot de logística convencional. Per aquest motiu, una vegada obtingut el robot definitiu, cal comprovar que realment el pressupost és molt més baix amb comparació als productes del mercat. Així doncs, a la **Taula 15** es pot veure la relació de preus amb cadascun dels elements que s'han necessitat per fer la construcció del robot.

**Taula 15** – Estudi de la viabilitat del projecte

| Material | Unitat | Preu Unitari | Total |
|---|---|---|---|
| Raspberry Pi 4 | 1 | 61,99€ | 61,99€ |
| Caixa Raspberry Pi | 1 | 20,99€ | 20,99€ |
| Micro SD 32 Gb 98 MB/s | 1 | 6,90€ | 6,90€ |
| Raspberry Pi Càmera v2.0 | 1 | 28,99€ | 28,99€ |
| Cinta flexible 100 cm Raspberry Pi Càmera | 1 | 4,49€ | 4,49€ |
| Motor CQRobot | 4 | 32,99€ | 131,96€ |
| Controladora L298N Dual H Bridge | 2 | 8,99€ | 17,98€ |
| 10 rodaments 6 x 13 x 5 mm | 3 | 9,69€ | 29,07€ |

| | | | |
|---|---|---|---|
| Servomotor LewanSoul 20 kg | 2 | 19,49€ | 38,98€ |
| Servomotor Longruner 17 kg | 1 | 18,99€ | 18,99€ |
| Bateria DSK 12 V 1.3 Ah | 1 | 9,91€ | 9,91€ |
| Convertidor Haofy de 12 V a 5 V | 1 | 12,98€ | 12,98€ |
| Carregador de bateria de plom 12 V | 1 | 9,99€ | 9,99€ |
| Final de carrera | 2 | 2,08€ | 4,16€ |
| Interruptor de tres posicions | 1 | 2,00€ | 2,00€ |
| Connector alimentació | 1 | 2,03€ | 2,03€ |
| Cinta plana IDC de cable de 10 pins 5 m | 1 | 8,79€ | 8,79€ |
| Volandera inox A2 | 1 | 2,29€ | 2,29€ |
| 30 T.Mètrica 3 x 12 | 1 | 1,69€ | 1,69€ |
| 30 T.Mètrica 3 x 16 | 3 | 1,79€ | 5,37€ |
| 30 T.Mètrica 3 x 20 | 1 | 1,69€ | 1,69€ |
| Tub alumini quadrat | 1 | 6,99€ | 6,99€ |
| Canaleta 16 x 10 mm | 1 | 2,89€ | 2,89€ |
| Fustes carro | 1 | 30,88€ | 30,88€ |
| Llistó potes carro | 1 | 10,00€ | 10,00€ |
| Rodes carro | 4 | 2,50€ | 10,00€ |
| PLA filament impressora gris | 2 | 15,72€ | 31,44€ |
| PLA filament impressora fusta | 1 | 34,99€ | 34,99€ |
| Total | | | 548,43€ |

Font: Elaboració pròpia.

Obtinguts els resultats de la taula anterior, es pot veure que el preu definitiu és de 548,43€. Tot i que a primera vista sembli que és un robot molt car, cal tenir present que els robots de logística ronden entre els 10.000 i els 80.000€. A la **Figura 78** es poden observar quatre tipus de robots de logística.

***Figura 78**. Recopilatori de robots de logística. (Plus, 2019)*

Tot i que estèticament estiguin més ben acabats els robots de les grans empreses, s'ha de tenir en compte que en Rooby està fet de fusta i que s'han intentat utilitzar els materials més barats possibles. Per aquest motiu, difícilment seria possible aconseguir un robot més barat perquè el preu dels motors puja molt el resultat final i s'hauria d'optar per utilitzar elements menys coneguts i, per tant, de menys qualitat. Així doncs, es pot contemplar com l'objectiu inicial del treball s'ha complert. Finalment s'ha aconseguit dissenyar i programar un robot de baix cost que serà útil per totes aquelles empreses, escoles o hospitals que no poden permetre's grans pressupostos.

# 4. CONCLUSIONS

Per concloure el treball és important saber que, malgrat totes les adversitats trobades al llarg de la realització del projecte, s'han pogut complir totes les expectatives inicials.

L'objectiu principal d'aquest projecte era construir un robot controlat per odometria capaç de moure's autònomament a partir d'uns punts de guia i de seguir persones mitjançant la visió per computador. És per aquest motiu que, per tal d'aconseguir-ho, s'ha dissenyat i creat un robot amb els mínims costos possibles. Per la part mecànica, amb l'ús de la impressora Creality CR-10S Pro s'han pogut realitzar molts dissenys que han permès no haver de comprar peces. Respecte l'electrònica, l'aprenentatge de l'ús de diferents sensors com el de llum o el polsador ha permès ampliar la programació i complir amb part dels objectius. Finalment, pel que fa a la programació, l'ús de llibreries com la OpenCV, la pyzbar i la VideoStream han sigut de gran utilitat per tal d'assolir l'objectiu.

Així doncs, s'ha aconseguit demostrar la hipòtesi inicial que partia de si una estudiant de batxillerat, amb els coneixements que tenia, era capaç de crear un robot de fàcil utilitat que simplifica la feina a les petites empreses, escoles o hospitals transportant material d'un costat a l'altre amb una baixa probabilitat d'error.

Pel que fa als requisits plantejats es conclou que s'han assolit els següents:

- Dissenyar un robot des de zero i construir-lo
- Fer funcionar cadascun dels elements que formen el robot
- Programar un seguidor d'usuaris a través de la visió per computador i un mode autònom fent ús de sensors

Les limitacions que s'han trobat en la realització del treball son varies. En primer lloc, el fet de no disposar d'una talladora làser ha fet que el treball amb les fustes fos complicat.

Calia una gran precisió i amb les eines i habilitats de què es disposava ha sigut complicat. En segon lloc, el fet de només disposar del taller de l'escola o el garatge d'un veí, ha limitat el temps de treball disponible per la construcció del robot. Per últim, cal afegir que el fet d'haver de reduir els costos i la manca d'estoc degut a la pandèmia també han dificultat molt poder mantenir la planificació prevista. Malgrat tots aquests entrebancs, els resultats són molt satisfactoris.

Hi han hagut un seguit de funcionalitats que degut a la manca de temps no s'han pogut implementar. En qualsevol cas, m'agradaria com a repte personal mirar d'implementar les següents aportacions:

- Creació d'una aplicació mòbil per a fer funcionar el robot
- Control dels motors amb l'ús del codificador
- Aprofundir més en l'apartat de seguiment d'usuaris

Per acabar, cal remarcar que **el robot és funcional i, a falta de més proves per adaptar-se a més escenaris, està llest per posar a la venta.**

***Figura 79.*** *Resultat final del carro i el robot. Elaboració pròpia*

# 5. RELACIÓ DE FONS


Arduino. (2020). *Arduino*. Recollit de Arduino: https://store.arduino.cc/arduino-uno-rev3

BananaPi. (2018). *Banana Pi Open Source Project*. Recollit de Banana Pi Open Source Project: http://www.banana-pi.org/w2.html

Barrientos Sotelo, V. R., & García Sánchez, S. O. (2007). Robots Móviles: Evolución y Estado del Arte. *Polibits*, 17.

BeagleBoard. (06 / 09 / 2019). *beagleboard*. Recollit de beagleboard: https://beagleboard.org/black

Bits, D. O. (17 / Juny / 2019). *Ochobitshacenunbyte*. Recollit de Ochobitshacenunbyte: https://www.ochobitshacenunbyte.com/2019/06/17/permisos-especiales-en-linux-sticky-bit-suid-y-sgid/

BSI. (2005). *ISO/IEC 19501:2005*.

Cerruzi, P. (2008). *Historia de la Informática. A fronteras del conocimiento.* Madrid: BBVA.

Cholewiak, S. A. (17 / Gener / 2017). *Semifluid*. Recollit de Semifluid: http://www.semifluid.com/2017/01/23/raspberry-pi-camera-comparison/

Corporativo. (27 / Juny / 2020). *gnu*. Recollit de gnu: https://www.gnu.org/philosophy/categories.es.html

Corporativo. (04 / Abril / 2020). *OkHosting*. Recollit de OkHosting: https://okhosting.com/blog/tipos-de-licencia-de-software/

Dejan. (Maig / 2019). *How to Mechatronics*. Recollit de https://howtomechatronics.com/projects/arduino-mecanum-wheels-robot/

Florentina, A., & Doroftei, I. (2011). Practical Applications for Mobile Robots based on Mecanum Wheels -. *MECAHITECH'11, 3*.

González, A. G. (Juny / 2014). *Panama Hitek*. Recollit de Panama Hitek: http://panamahitek.com/el-puente-h-invirtiendo-el-sentido-de-giro-de-un-motor-con-arduino/

Gonzalez, A., Martínez de Pison, F., Pernía, A., Alba Elías, F., Castejón, M., Ordieros, J., & Vergara, E. (2006). *Técnicas y algoritmos básicos de visión artificial.* La Rioja: Universidad de la Rioja.

Hassan, M. (25 / Març / 2020). Learn OpenCV in 3 hours with Python. Recollit de https://www.youtube.com/watch?v=WQeoO7MI0Bs&t=1307s

Jimenez, C. J. (2017). *Desrrollo de un sistema de control de un robot móvil.* Valladolid: Universidad de Valladolid - Escuela de Ingenierías Industriales.

Linux, N. (05 / Gener / 2013). *Nazion Linux*. Recollit de Nazion Linux: https://www.nazionlinux.com/come-creare-un-avviatore-desktop/

Maza, G. V. (2017). *Procesamiento de imágenes usando opencv aplicado en raspberry pi para la clasificación del cacao.* Pirhua: Universidad de Pirhua.

Navas, M. A. (11 / Setembre / 2016). *Profesional review*. Recollit de Profesional review: https://www.profesionalreview.com/2016/09/11/gestor-de-paquetes-en-linux/

OpenCV. (10 / Agost / 2014). *OpenCV - Python Tutorials*. Recollit de OpenCV - Python Tutorials: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

Oriol. (05 / Juliol / 2015). *ComputerNewAge*. Recollit de ComputerNewAge: https://computernewage.com/gnu-linux/gestion-software/#gestores-paquetes-repositorios

Pérez Martín, J. (2016). *Desarrollo de un sistema controlador para una red domótica inalámbrica*. San Cristóbal: Universidad de La Laguna.

Perseo. (2011). *DesdeLinux*. Recollit de DesdeLinux: https://blog.desdelinux.net/permisos-y-derechos-en-linux/

Plus, G. (23 / Setembre / 2019). *Geek Plus*. Recollit de Geek Plus: https://www.geekplus.com/product-2/moving

Raffino, M. E. (30 / Maig / 2020). *Concepto.de*. Recollit de Concepto.de.: https://concepto.de/cpu/

Raffino, M. E. (28 / Juliol / 2020). *Concepto.de*. Recollit de Concepto.de.: https://concepto.de/codigo-fuente/

RaspberryPi. (2020). *Raspberry Pi*. Recollit de Raspberry Pi: https://www.raspberrypi.org/

Rosebrock, A. (23 / 07 / 2018). *PyImageSearch*. Recollit de PyImageSearch: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

Sánchez, C. M. (2014). *Robots móviles de ruedas: Generalidades*. Mèxic: Upiita.

Solución, P. (10 / Maig / 2018). *Diferencias entre CLI y GUI*. Recollit de Diferencias entre CLI y GUI: https://pc-solucion.es/2018/05/10/diferencias-entre-cli-y-gui/

Studiomaven. (13 / 11 / 2014). *Studiomaven*. Recollit de http://studiomaven.org/Course__200c_f14_steinfeld_session_740180.html

Sulbaran, H. (31 / Desembre / 2012). *Efémides de Tecnología*. Recollit de Efémides de Tecnología: https://helisulbaran.blogspot.com/2013/05/07-de-mayo-1954-ibm-lanza-el-ibm-704-su.html

Team, O. (2020). *Opencv*. Recollit de Opencv: https://opencv.org/

Valcare, J. (05 / Març / 2014). *Automaticación Industrial*. Recollit de Automatización Industrial: http://automatizacion.eu/images/vision/consejos/cv_bas_guide_vol2.pdf

Valcare, J. (05 / Març / 2014). *Automatización Industrial*. Recollit de Automatización Industrial: http://automatizacion.eu/images/vision/consejos/cv_bas_guide_vol2.pdf

# 6. GLOSSARI

En aquest apartat hi ha un conjunt de paraules que es fan servir de manera reiterada al llarg del treball i que són necessàries per al correcte seguiment.

## 6.1. Linux

**Codi font**: Conjunt de línies de text que expressen els passos que ha de seguir la computadora per la correcta execució d'un programa específic. És a dir, és aquell codi que es fa en llenguatges no binaris.

**Consola o terminal:** Eina capaç de realitzar les tasques de manera més ràpida que mitjançant una interfície gràfica del sistema operatiu. Consumeix menys recursos i ens permet treballar remotament amb l'ordinador. L'he fet servir per connectar-me a la Raspberry des del meu ordinador i sense cables.

**Debian:** És una distribució de Linux. En concret, és la que s'ha fet servir en el curs de formació i a la Raspberry per controlar el robot.

***Kernel***: També anomenat nucli, és la part central d'un sistema operatiu i el que s'encarrega de realitzar tota la comunicació segura entre el software i el hardware de l'ordinador.

## 6.2. Robòtica

***Drivetrain*:** Base on hi ha el sistema motriu d'un robot mòbil.

**GPIO:** Cadascun dels ports d'entrada i/o sortida de la Raspberry. Es poden configurar fàcilment amb Python amb les llibreries RPi.GPIO i GPIO Zero.

**Enginyeria de requeriments:** Documentació d'un seguit de necessitats.

**Moviment holonòmic**: A més dels moviments naturals d'anar endavant, enrere o girar sobre l'eix, un robot amb aquesta característica és capaç de fer moviments laterals i diagonals sense haver de girar prèviament.

**Moviment *pan &tilt*:** Son dos tipus de moviments. Un rota sobre l'eix vertical (tilt) i l'altre sobre l'horitzontal (pan).

**Rodaments:** Peça que serveix per evitar la fricció en moviments rotatoris com un eix motriu.

***Shaft collar***: Peça que subjecte un eix, per exemple hexagonal, a una segona peça mitjançant cargols.

## 6.3. Programació

**Diagrama de casos:** Resum en forma de taula de les necessitats i estructura d'un algorisme.

**Diagrama de flux**: Representació gràfica d'un algorisme o procés.

**Fil d'execució o *Thread*:** Cada fil pot executar una operació a la vegada. En el món de la informàtica sovint s'utilitzen els *multithreads* per poder fer vàries operacions en paral·lel.

**Funció:** Conjunt d'instruccions parametritzades que es poden reutilitzar.

**Llibreria:** Conjunt de funcions implementades per algú per complir un objectiu.

## 6.4. Visió per computador

**Fotograma:** Cadascuna de les imatges que formen un vídeo.

**FPS**: Fotogrames per segon. Serveix per calcular la velocitat de fotogrames d'un vídeo.

**Píxel**: És la menor unitat de representació del color en una imatge.

**Centroide:** Coordenades en el pla del centre d'un conjunt de píxels.

# 7. ANNEX 1 - FUNCIONS BÀSIQUES DE OPENCV

## 7.1. Preprocessament

**Taula 16** - OpenCV, la funció Dilate

| **Funció: Dilate** | |
|---|---|
| **Per què serveix:** Incrementa el color blanc d'una imatge en blanc i negre fent que els contorns augmentin de tamany | **Exemple**:<br>  <br>***Figura 80**. Aplicació del filtre Dilate a una imatge. Elaboració pròpia* |
| **Quan fer-la servir:** S'utilitza per eliminar els espais buits que queden a la imatge després de passar-la en blanc i negre | |
| **Com es fa servir:** cv2.dilate(imatge, kernel, iteracions = x) | |

Font: Elaboració pròpia.

**Taula 17** - OpenCV, la funció Erode

| **Funció: Erode** | |
|---|---|
| **Per què serveix:** Incrementa el color negre d'una imatge en blanc i negre fent que els contorns disminueixin de mida | **Exemple**:<br>  <br>***Figura 81**. Aplicació del filtre Erode a una imatge. Elaboració pròpia* |
| **Quan fer-la servir:** S'utilitza per eliminar els soroll que apareix a la imatge o bé també per separar dos objectes que han quedat units | |
| **Com es fa servir:** cv2.erode(imatge, kernel, iteracions = x) | |

Font: Elaboració pròpia.

**Taula 18** - OpenCV, la funció Canny

| **Funció: Canny** | |
|---|---|
| **Per què serveix:** Transforma la imatge a blanc i negre per poder-ne determinar les bores i les cantonades dels diferents objectes | **Exemple**:<br>  <br>**Figura 82**. Aplicació del filtre Canny a una imatge. Elaboració pròpia |
| **Quan fer-la servir:** S'utilitza sempre per poder dur a terme la fase de preprocessat d'una manera més encertada | |
| **Com es fa servir:** cv2.Canny(imatge, llindar mínim, llindar màxim) | |

Font: Elaboració pròpia.

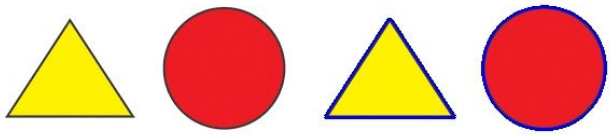
# 7.2. Segmentació

**Taula 19** - OpenCV, la funció Find Contours

| **Funció: Find Contours** |
|---|
| **Per què serveix:** Detecta el contorn de la figura que estàs buscant |
| **Quan fer-la servir:** S'utilitza quan t'interessa conèixer on es troba la figura i quina forma té |
| **Com es fa servir:** cv2.findContours(imatge, mode, mètode) |

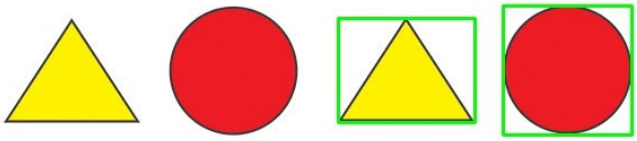
Font: Elaboració pròpia

**Taula 20** - OpenCV, la funció Draw Contours

<table>
<tr><th colspan="2">Funció: Draw Contours</th></tr>
<tr><td>Per què serveix: Dibuixa el contorn de la figura que has detectat anteriorment</td><td rowspan="2">Exemple:<br>  <br>Figura 83. Aplicació de la funció Draw Contours a una imatge. Elaboració pròpia</td></tr>
<tr><td>Quan fer-la servir: S'utilitza per treballar la figura que has detectant a la mateixa pestanya o bé, en una altra</td></tr>
<tr><td colspan="2">Com es fa servir: cv2.drawContours(imatge, contorn, -1, color)</td></tr>
</table>

Font: Elaboració pròpia.

**Taula 21 -** OpenCV, la funció Bounding Rectangle

<table>
<tr><th colspan="2">Funció: Bounding Rectangle</th></tr>
<tr><td>Per què serveix: Dibuixa un rectangle al voltant de la figura que estàs detectant</td><td rowspan="2">Exemple:<br>  <br>Figura 84. Aplicació del Bounding Rectangle a una imatge. Elaboració pròpia</td></tr>
<tr><td>Quan fer-la servir: S'utilitza sempre que es vol saber si estàs detectant bé la figura o no</td></tr>
<tr><td colspan="2">Com es fa servir: cv2.boundingRect(vèrtexs)</td></tr>
</table>

Font: Elaboració pròpia.

# 7.3. Parametrització

**Taula 22** - OpenCV, la funció Àrea

| **Funció: Àrea** |
|---|
| **Per què serveix:** Calcula l'àrea de la figura detectada |
| **Quan fer-la servir:** S'utilitza quan busques una figura igual a les altres però de diferent mida |
| **Com es fa servir:** cv2.contourArea(contorn, orientació) |

Font: Elaboració pròpia.

**Taula 23** - OpenCV, la funció Polígon

| **Funció: Polígon** |
| --- |
| **Per què serveix:** Calcula quants vèrtexs té un polígon a partir de la seva curvatura |
| **Quan fer-la servir:** S'utilitza per diferenciar un polígon d'un altre quan els dos són de la mateixa mida |
| **Com es fa servir:** cv2.approxPolyDP(curvatura, precisió, tancat) |

Font: Elaboració pròpia.

**Taula 24** - OpenCV, la funció Llargada

| **Funció: Llargada** |
| --- |
| **Per què serveix:** Calcula el perímetre d'un polígon o bé la llargada de la seva curvatura |
| **Quan fer-la servir:** S'utilitza per diferenciar un polígon d'un altre quan la única diferencia de la que disposen és el perímetre |
| **Com es fa servir:** cv2.arcLength(curvatura, tancat) |

Font: Elaboració pròpia.

# 8. ANNEX 2 – AUTÒNOM

En aquest annex s'explica detalladament el procés a seguir per tal de realitzar el programa del mode autònom del robot. L'objectiu d'aquest mode és aconseguir que el robot es pugui moure per l'espai sense necessitar cap control de l'usuari.

L'algoritme ideal per aquest mode ha de complir:

- Una interpretació ràpida de codis QR
- Un seguiment de línies amb un PID

A falta d'un sensor analògic, es realitza el mateix treball amb un de digital. Tot i que el codi es modifica, queda molt simplificat i fa la mateixa funció que un moviment amb un PID. A continuació s'expliquen els passos a seguir per tal d'obtenir el mode autònom.

1. Detecció i descodificació dels codis QR

   En aquesta primer part el robot enregistrarà tot el que succeeix amb l'objectiu de buscar el codi QR que, descodificat, té el mateix missatge que ha rebut. D'aquesta manera podrà arribar fins al lloc desitjat.

2. Moviment del robot a partir del seguidor de línies

   Per tal de poder trobar els codis QR, el robot s'haurà de moure. Així doncs, aquesta funció consta d'un seguidor de línies preparat especialment per un sensor digital.

3. Unificació de les dues funcions perquè es duguin a terme alhora

   Finalment, cal que el robot realitzi les dues funcions anteriors. Com que no interessa que sigui un procés seguit d'un altre, sinó que es necessita que es realitzi a la vegada, cal programar-ho d'una manera diferent a l'habitual.

# 8.1. Estructura del mode autònom

A continuació s'explica detalladament l'estructura del codi del mode autònom per tal d'entendre com s'han assolit els objectius inicials del programa.

## 8.1.1. Detecció i descodificació dels codis QR

Com ja s'ha mencionat anteriorment, el primer pas és programar el detector i descodificador de codis QR. La llibreria més important per aquest programa és la `pyzbar` perquè permet descodificar els codis QR per poder interpretar-los. Tot seguit es mostren els passos realitzats.

```
1. # import dels paquets necessaris
2. import time, imutils
3. from imutils.video import VideoStream
4. from pyzbar import pyzbar
5.
6. def qrCode_fn(self, qrStop):
7.   # engegar la càmera i llegir totes les imatges descodificant
8.   # els codis QR trobats per conèixer el punt d'arribada
9.   vs = VideoStream(usePiCamera=True,
                      resolution=(320,240)).start()
10.  time.sleep(0.5)
11.  qr = False
```

A les **Línies 2-4** s'importen tots els paquets que faran falta al llarg del programa: `time`, `imutils`, `VideoStream` i `pyzbar`.

A partir de la **Línia 6** es crea la funció `qrCode_fn()` on es realitzarà tota la resta del programa. A les següents línies es duu a terme la inicialització de la funció engegant la càmera (**Línies 9 i 10**) i creant la variable que permetrà sortir del bucle de lector de codis QR (**Línia 11**).

```
1.   while not qr:
2.     frame = vs.read()
3.     decode = pyzbar.decode(frame)
4.     for qrInformation in decode:
5.       if qrInformation.data.decode('utf-8') == qrStop:
6.         qr = True
7.   vs.stop()
```

A la **Línia 1** s'inicia el bucle que es dedicarà a descodificar tots els codis QR que vagi trobant fins que trobi el que coincideix amb l'ordre enviada per l'usuari.

Primer es realitza la lectura de la imatge enregistrada per la càmera (**Línia 2**). Tot seguit, amb la llibreria `pyzbar` es detecta si hi ha algun codi QR i en el cas de que hi sigui, es descodifica i es guarda dins de la variable `decode` (**Línia 3**). D'aquesta manera, en les pròximes línies es podrà determinar la finalització de la funció.

Finalment, a la **Línia 4** es crea un bucle que llegirà cada missatge dels codis QR guardats a la variable `decode`. A les **Línies 5 i 6** es duu a terme la comparació que determina si el codi QR detectat coincideix o no amb l'ordre rebuda i, en el cas que coincideixi, es para la càmera donant per acabada la funció (**Línia 7**).

### 8.1.2. Moviment del robot a partir del seguidor de línies

A la segona part es duu a terme tota la part de programació del moviment del robot perquè pugui cercar el codi QR sol·licitat per l'usuari. En aquest cas és essencial la llibreria `LineSensor` per tal de poder utilitzar el sensor de llum i detectar la línia. A més, s'utilitzarà una altra funció dins de la mateixa classe Rooby per moure les rodes.

```
1. # import dels paquets necessaris
2. import RPi.GPIO as GPIO
3. from gpiozero import LineSensor
4.
5. def lineFollower_fn(self):
6.   # amb el sensor de llum detectar la línia negra i seguir-la in
       definidament
7.   # CREACIÓ DE LES VARIABLES
8.   # creació de les llistes per, més endavant, poder comparar i
       saber cap a on s'haurà de moure el robot
9.   forward   = [[1, 0, 1], [0, 0, 0]]
10.  turnRight = [[0, 0, 1], [0, 1, 1]]
11.  turnLeft  = [[1, 0, 0], [1, 1, 0]]
12.  out       = [1, 1, 1]
13.  # creació de la resta de variables que serviran per tancar el
       programa
14.  lastMove = 'none'
15.  counter = 0
16.  sensorL = LineSensor(self._pins['sensors']['light']['lft'])
17.  sensorM = LineSensor(self._pins['sensors']['light']['mdl'])
18.  sensorR = LineSensor(self._pins['sensors']['light']['rgt'])
```

A les **Línies 2 i 3** s'importen tots els paquets que faran falta al llarg del programa: `RPi.GPIO` i `LineSensor`.

A partir de la **Línia 5** es crea la funció `lineFollower_fn()` on es realitzarà tota la resta del programa. Entre les **Línies 9 i 18** es creen totes les variables necessàries pel programa. Les que es troben entre les **Línies 9-12** determinen el moviment del robot i, la resta de línies (**Línies 14-18**) serveixen per tancar el programa. Més endavant, al llarg de les següents línies s'entendrà millor la funcionalitat de totes aquestes variables.

```
1.   # MAIN
2.   #  PRIMER: es guarda en una llista el valor de cada sensor per
        poder detectar la direcció del robot
3.   while True:
4.     if lastMove == 'stop':
5.       break
6.     move = [sensor.value, sensorM.value, sensor.value]
```

A la **Línia 3** s'inicia el bucle que permetrà el moviment indefinit del robot seguint la línia.

Primer es comprova que la variable `lastMove` és igual al missatge de parar per determinar la finalització del programa (**Línies 4 i 5**). En el cas que no siguin iguals, a la **Línia 6** es llegeix el valor de cada sensor i es guarda com a llista dins de la variable `move`.

```
1.   #  SEGON: es compara la llista creada amb cadascuna de les
        opcions dels possibles valors existents
2.     if move == forward[0] or move == forward[1]:
3.       self.move(0,0)
4.       lastMove = 'forward'
5.     elif move == turnRight[0] or move == turnRight[1]:
6.       self.move(45,0)
7.       lastMove = 'turnRight'
8.     elif move == turnLeft[0] or move == turnLeft[1]:
9.       self.move(315,0)
10.      lastMove = 'turnLeft'
11.    else:
12.      self.stop()
13.      lastMove = 'stop'
```

Al llarg de les **Línies 2-13** es duu a terme tota la decisió del moviment. Depenent del valor detectat per cadascun dels sensors de llum el robot s'haurà de moure cap a un costat o cap a l'altre. És a dir, si per exemple el robot es troba en una situació on anteriorment

s'ha especificat que haurà de girar a la dreta, realitzarà aquest moviment. L'esquema de la decisió del moviment es troba explicat a la **Figura 65**

### 8.1.3. Unificació de les dues funcions

Finalment, una vegada les dues funcions ja estan acabades i funcionen correctament cal unificar-les per tal d'aconseguir el mode autònom definitiu. En aquesta última fase és essencial l'ús dels threads ja que permeten realitzar dos processos alhora. Així doncs, es pot veure com en la funció hi destaca la llibreria `multiprocessing`.

```
1. # import dels paquets necessaris
2. import time, multiprocessing
3.
4. def autonomousMode(self):
5.   # execució de les funcions qrCode_fn() i lineFollower_fn()
6.   # primer es guarden les funcions anteriors com un multiprocés
7.   qrCode = multiprocessing.Process(name= 'qrCode', target=
              self.qrCode_fn, arg= qrStop)
8.   lineFollower = multiprocessing.Process(name= 'lineFollower',
                    target= self.lineFollower_fn)
9. # finalment s'engeguen els dos processos i s'espera a que es de-
   tecti el codiQR demanat per parar el procés del seguidor de lí-
   nies
10.  qrCode.start()
11.  lineFollower.start()
12.  time.sleep(0.5)
13.  qrCode.join()
14.  lineFollower.terminate()
```

A la Línia 2 s'importen tots els paquets que faran falta al llarg del programa: `time` i `multiprocessing`.

A partir de la **Línia 4** es crea la funció `autonomousMode()` on es realitzarà tota la resta del programa. A les **Línies 7 i 8** es guarden les funcions que es volen utilitzar com a processos per, a les següents línies poder engegar-les alhora.

Finalment, a les **Línies 10-14** s'engeguen els dos processos i s'espera a que el codi QR sol·licitat per l'usuari sigui detectat per parar la funció del seguidor de línies amb un `terminate()`.

# 9. ANNEX 3 – SEGUIDOR D'USUARIS

En aquest annex s'explica detalladament el procés a seguir per tal de realitzar el programa del mode de seguiment d'usuaris. L'objectiu d'aquest mode és aconseguir que el robot pugui seguir a un usuari sempre que se li demani i, per fer-ho, segueix una pilota d'espuma de color verd croma.

L'algoritme ideal per aquest mode ha de complir:

- Una detecció segura de l'objecte
- Haver de detectar l'objecte només una vegada
- Saber gestionar la pèrdua de l'objecte en la imatge

A falta de temps, s'ha acabat creant un programa senzill però molt efectiu. Tot i que hi ha millores pendents, el robot no s'equivoca i sap mantenir el seguiment fins que se l'ordena parar. A continuació s'expliquen els passos a seguir per tal d'obtenir el mode de seguiment d'usuaris.

1. Detecció de l'objecte a seguir

   En aquesta primer part, a través de mètodes de preprocessament d'imatge, el robot detectarà l'objecte que es vol seguir.

2. Seguiment de l'objecte

   Quan s'hagi trobar la pilota, calcularà el seu centroide i dibuixarà una circumferència al voltant per tal de determinar-ne el radi. Una vegada es coneix l'objecte i el seu centroide, el robot seguirà aquest objecte sabent cada vegada on està situat el centroide.

3. Moviment del robot pel seguiment

   Finalment, cal que el robot es mogui per poder seguir a l'usuari. En aquesta última part, es calcularà quins són els moviments que ha de realitzar el robot per tal de no perdre de vista la pilota.

# 9.1. Estructura del mode de seguiment d'usuaris

A continuació s'explica detalladament l'estructura del codi del mode de seguiment d'usuaris per tal d'entendre com s'han assolit els objectius inicials del programa.

## 9.1.1. Detecció de l'objecte a seguir

Així doncs, el primer pas per obtenir el seguiment d'usuaris és detectar l'objecte a seguir. En aquest cas, la pilota. Per fer-ho, s'utilitzen funcions de la llibreria OpenCV que permeten ressaltar el color verd de la imatge i, per tant, simplificar aquesta detecció. Tot seguit, es mostren els passos realitzats.

```
1.   # import dels paquets necessaris
2.   import imutils, time, cv2, RPi.GPIO as GPIO
3.   from imutils.video import VideoStream
4.   from collections import deque
5.
6.   def trackingMode(self):
7.     # detectar l'objecte, seguir-lo i moure el robot per tal de
         dur a terme el seguiment
8.     # CREACIÓ DE LES VARIABLES
9.     # definir el llindar del color verd a detectar en l'espai de
         color HSV i crear la llista dels centroides
10.    greenLower    = (16, 30, 0)
11.    greenUpper    = (75, 225, 200)
12.    pts           = deque(maxlen=64)
13.    width, height = 640, 480
14.    # definir les franges per determinar el moviment del robot
15.    strip1 = int((width*25)/100)
16.    strip2 = int((width*15)/100)
17.    strip3 = int((width*20)/100)
18.    # càlcul dels punts en l'eix de les abscisses per determinar
         el moviment que haurà de realitzar el robot
19.    turnLeft  = strip1
20.    left      = int(strip1  + strip2)
21.    forward   = int(left    + strip3)
22.    right     = int(forward + strip2)
23.    turnRight = int(right   + strip1)
```

A les **Línies 2-4** s'importen tots els paquets que faran falta al llarg del programa: `imutils`, `time`, `cv2`, `RPi.GPIO`, `VideoStream` i `deque`.

A partir de la **Línia 6** es crea la funció `trackingMode()` on es realitzarà tota la resta del programa. A les **Línies 9 i 10** es creen els llindars del color verd en l'espai de color HSV per poder detectar la pilota amb més facilitat. Tot seguit, a la **Línia 12** es crea una llista de màxim 64 elements on s'hi guardaran tots els centroides detectats. Finalment, a la **Línia 13** es determina l'alçada i l'amplada que tindrà la imatge.

Entre les **Línies 14-22** es realitzen els càlculs de les diferents franges de la imatge per, més endavant, determinar el moviment del robot. De la **Línia 14** a la **Línia 16** es calcula quin tant per cent de la imatge pertoca a cada franja i, de la **Línia 19** a la **Línia 22** s'assignen els resultats anteriors a un tipus de moviment determinat. Per entendre aquesta distribució de franges amb moviments veure **Figura 70**.

```
1.   vs = VideoStream(usePiCamera=True, width=width).start()
2.   time.sleep(0.5)
3.   while True:
4.     frame = vs.read()
5.     # aplicar els filtres Gaussian Blur i HSV a la imatge
6.     blurred = cv2.GaussianBlur(frame, (25, 25), 0)
7.     hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
8.     # a partir de la transformació de la imatge a l'espai de
         color HSV crear la màscara de la pilota
9.     mask = cv2.inRange(hsv, yellowLower, yellowUpper)
10.    mask = cv2.erode(mask, None, iterations=2)
11.    mask = cv2.dilate(mask, None, iterations=2)
```

A les Línies 1 i 2 s'engega la càmera per, a la Línia 3 poder iniciar el bucle que es dedicarà a calcular els centroides de cada imatge rebuda i a seguir l'objecte.

Primer es realitza la lectura de la imatge enregistrada per la càmera (**Línia 4**). Com ja s'ha comentat anteriorment, per tal de facilitar la detecció de la imatge s'aplica el filtre Gaussian Blur i es transforma a l'espai de Color HSV (**Línies 6 i 7**). Una vegada s'ha ressaltat la pilota respecte la resta d'objectes, es crea la màscara per guardar només l'objecte desitjat de la imatge (**Línies 9-11**).

### 9.1.2. Seguiment de l'objecte

Una vegada s'ha aplicat la màscara, ja es pot inicial el procés de seguiment de l'objecte. L'objectiu és trobar el centroide i el radi de la pilota per poder seguir el seu moviment. En aquesta part en destaca el càlcul del centroide ja que permet conèixer sempre les coordenades a les que es troba l'objecte.

```
1.    # càlcul del contorn de la figura per poder-ne determinar
        tant el centroide com el radi
2.    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
3.    cv2.CHAIN_APPROX_SIMPLE)
4.    cnts = imutils.grab_contours(cnts)
5.    centroid = None
6.    # lectura de tots els valors dels contorns per acabar-ne
        trobant el centroide i el radi
7.    if len(cnts) > 0:
8.      c = max(cnts, key=cv2.contourArea)
9.      ((x, y), radius) = cv2.minEnclosingCircle(c)
10.     M = cv2.moments(c)
11.     centroid = (int(M['m10'] / M['m00']), int(M['m01'] /
                    M['m00']))
```

Entre les **Línies 2-5** es preparen totes les variables i s'apliquen les funcions necessàries per facilitar el càlcul del centroide. En aquest cas es busca el contorn de l'objecte detectat a través del `grab_contours()`. Tot seguit, a la **Línia 5** es crea la variable del centroide tot i que, de moment, es deixa com a `None`.

És a partir de la **Línia 7** que a través d'un recorregut de tots els contorns trobats es va calculant cada centroide. Abans dels centroides però, es busca el radi i les coordenades d'aquest (**Línies 9 i 10**). Finalment, a través dels moments es troba el centroide de cada imatge capturada per la càmera (**Línia 11**).

### 9.1.3. Moviment del robot pel seguiment

Finalment, una vegada s'obté la imatge dividida en diferents franges, s'ha detectat la pilota i es coneixen les coordenades del centroide, ja es pot dur a terme l'última part del mode de seguiment d'usuaris: el moviment del robot. Per programar-lo s'han de tenir en compte els aspectes que s'explicaran a continuació.

```
1.      # radi entre els 10 i 40 mm
2.      if 10 < radius < 40:
3.        if centroid is None:
4.          self.stop()
5.        elif (centroid[0] > 0) and (centroid[0] < turnLeft):
6.          self.turn('left', 0)
7.        elif (centroid[0] > turnLeft) and (centroid[0] < left):
8.          self.move(225, 0)
9.        elif (centroid[0] > left) and (centroid[0] < forward):
10.         self.move(270, 0)
11.       elif (centroid[0] > forward) and (centroid[0] < right):
12.         self.move(315, 0)
13.       elif (centroid[0] > right) and (centroid[0] < turnRight):
14.         self.turn('right', 0)
```

En aquesta part del codi ens trobem que el radi medeix entre 10 i 40 mm. Aquest fet implica que la pilota està situada prou lluny del robot i que, per tant, la persona s'està allunyant d'ell. Per aquest motiu, en aquest cas el robot es mourà sempre en direcció positiva acostant-se a l'usuari. Entre les **Línies 2-14** es poden veure els casos de cadascuna de les franges determinades anteriorment. Gràcies a la distribució de les franges és més senzill saber el moviment que s'ha de realitzar.

```
1. # radi entre els 40 i 50 mm
2.       elif 40 < radius < 50:
3.         self.stop()
```

En aquest cas ens trobem que el radi medeix entre 40 i 50 mm. Aquesta és la distància de seguretat entre el robot i la persona. És a dir, en el cas de que la persona no es mogui, el robot no es pot moure indefinidament, sinó que hi ha d'haver un punt on pari. En aquest cas, aquesta és a la distància a la que s'atura (**Línies 2 i 3**)

```
1. Radi superior als 50 mm
2. elif 50 < radius:
3. if centroid is None:
4. self.stop()
5. elif (centroid[0] > 0) and (centroid[0] < turnLeft):
6. self.turn('left', 0)
7. elif (centroid[0] > turnLeft) and (centroid[0] < left):
8.  self.move(135, 0)
9. elif (centroid[0] > left) and (centroid[0] < forward):
10.  self.move(90, 0)
11.  elif (centroid[0] > forward) and (centroid[0] < right):
12.  self.move(45, 0)
13.  elif (centroid[0] > right) and (centroid[0] < turnRight):
14.  self.turn('right', 0)
```

Finalment existeix la possibilitat de que el radi sigui superior als 50 mm. En aquest cas la persona s'està acostant massa al robot i, per tant, aquest s'ha de moure enrere. L'estructura és la mateixa que el primer cas però en aquest el robot es mou enrere (**Línies 2-14**)

Finalment, després de programar el moviment del robot, s'obté la versió definitiva del mode de seguiment d'usuaris. Tot i que es pot complementar amb altres punts, aquest programa és funcional i consta d'un gran marge d'error.