TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
Universitat Pompeu Fabra Barcelona

Enginyeria Electrònica Industrial i Automàtica

PROGRAMACIÓ COL·LABORATIVA ENTRE TURTLEBOT I TM5-900

Memòria final

Pol Caballero Estebanell

PONENT: Josep López Xarbau

PRIMAVERA 2022

TecnoCampus
Mataró-Maresme

Agraïments

A en Josep López Xarbau per haver-me ajudat a fer realitat aquest projecte.

A en Mark Rodríguez per ajudar-me a resoldre problemes en la sortida del node *listen.*

A l'Enric Sitjà per donar-me suport emocional durant tota la realització del projecte i ajudar-me a mantenir la motivació.

A la comunitat de ROS i Stack Overflow per a ajudar-me a resoldre tots els dubtes que he anat tenint amb tota la programació que s'ha hagut de fer.

Finalment, a la meva mare, la meva germana i el meu pare, que tot i haver-nos deixat aquest estiu passat, m'ha inspirat a acabar el projecte.

Resum

Aquest projecte consisteix en la creació d'un entorn de treball on el TurtleBot i el braç robòtic TM5 interactuïn entre ells. El TurtleBot, mitjançant un sensor LIDAR i tècniques de SLAM, entregarà al TM5 una peça. El braç robòtic, mitjançant visió artificial, recollirà la peça i la dipositarà en la cèl·lula de fabricació. S'emprarà ROS basat sobre Linux per a programar el TurtleBot i el TM5.

Resumen

Este proyecto consiste en la creación de un entorno de trabajo donde TurtleBot i el brazo robótico TM5 interactúan entre ellos. TurtleBot, mediante un sensor LIDAR y técnicas de SLAM, entregara al TM5 una pieza. El brazo robótico, mediante visión artificial, recogerá la pieza y la depositará en la célula de fabricación. Se utilizará ROS basado sobre Linux para programar TurtleBot y TM5.

Abstract

This project pretends to create a workspace where TurtleBot and the robotic arm TM5 interact with each other. TurtleBot, using a LIDAR sensor and SLAM techniques, will supply TM5 with a piece. The robotic arm, using artificial vision, will collect the supplied piece and deposit it into the industrial workspace. ROS running on Linux will be used to program TurtleBot and the robotic arm.

Índex.

Índex de figures

Glossari de termes

LIDAR	*Light Detection And Ranging/Laser Imaging, Detection And Ranging*
PCB	*Printed Circuit Board*
QR	*Quick Response Code*
ROS	*Robot Operating System*
SCARA	*Selective Compliance Assembly Robot Arm*
SLAM	*Simultaneous Localization And Mapping*
TCP	*Tool Center Point*
GUI	*Grafical User Interface*

Normatives

Directives

2006/42/CE – Directiva de les maquines per garantir la seguretat del usuari

Normatives UNE-EN ISO

UNE-EN ISO 10218-1:2011 i UNE-EN ISO 10218-2:2011: Requisits de seguretat per a robots industrials. Part 1 i 2

Normatives ISO/TS/TR

ISO/TS 15066:2016 - Requisits de seguretat per a robots col·laboratius

ISO/TR 20218-1:2018 – Disseny segur de robots industrials

1. Objectius

1.1. Propòsit

El propòsit d'aquest projecte és desenvolupar un entorn industrial on hi interactuen un robot autònom, un robot col·laboratiu i un PLC per així poder aportar un camp de treball on els estudiants i professorat puguin assolir els coneixements necessaris i realitzar les proves convenients.

1.2. Finalitat

Obtenir una eina amb l'objectiu de col·laborar en l'assoliment dels conceptes robòtics tractats en aquest treball tot tenint en compte l'industria 4.0 i totes les tècniques tractades en aquest.

1.3. Objecte

Obtenir tant el TurtleBot com el braç robòtic TM5 i el PLC funcionals per poder fer una demostració complerta i redactar la memòria del projecte.

1.4. Abast

Pel que fa a l'abast d'aquest projecte, aquest incorpora un estudi de la robòtica en àmbit general i específic de cara al TurtleBot i el braç robòtic Techman, els robots utilitzats en aquest projecte.

En quant al TurtleBot s'ha realitzat un estudi del LIDAR i de ROS. Es realitzarà una programació amb Python per generar un algoritme que permeti al TurtleBot, un cop tingui el terreny guardat en memòria, trobar la seva posició i anar fins a una posició assignada.

Respecte al braç robòtic Techman s'ha realitzat un estudi de trajectòries, arquitectures de braços robòtics i visió artificial. Es realitzarà una programació combinant TMFlow, el software de programació proporcionat per Techman, ROS i Python per realitzar tasques específiques com ara detecció d'imatge.

En quant al PLC es programarà mitjançant Python, llegint i escrivint les sortides i entrades del PLC mitjançant la llibreria *pylogix.*

Finalment, aquest projecte inclou una planificació, viabilitat tècnica, mediambiental i econòmica a més a més d'un estudi de perspectiva de gènere.

2. Revisió d'antecedents i necessitats d'informació

2.1. Història dels robots

Un exemple de robot que es pot trobar de la mà d'un dels inventors més importants de la història és el *Robot de Leonardo*, de *Leonardo da Vinci*. Aquest humanoide es creu que va ser construït sobre l'any 1495, i podria moure els braços de forma independent, aixecar la visera de l'armadura. Aquest robot podia realitzar els moviments mitjançant politges.[i]

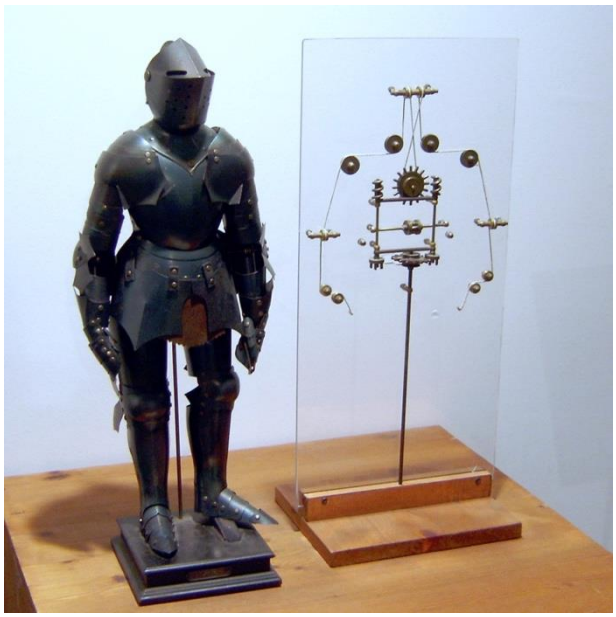


Figura 1: Robot de Leonardo

Una de les primeres aparicions de robots automatitzats és el *Teler de Jacquard*, un teler creat per *Joseph Marie Jacquard* l'any 1801. Aquest dispositiu utilitzava unes plaques perforades de cartró per crear patrons complexes amb tela de manera autònoma, simplement feia falta carregar el un seguit de plaques en un ordre concret per obtenir el patró desitjat, encendre la màquina i el patró es començava a fer.

Figura 2: Teler de Jacquard

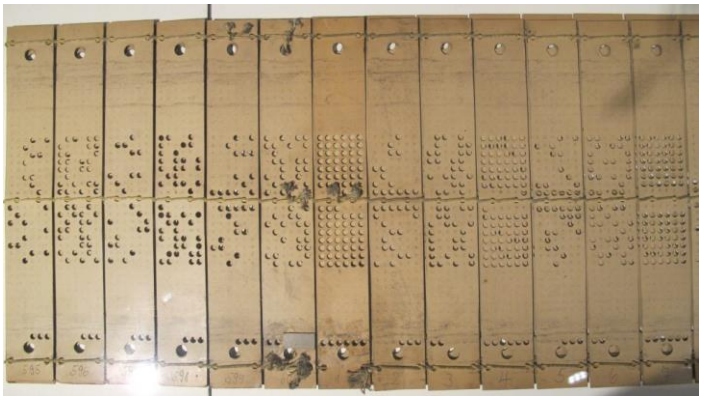


Figura 3: Plaques perforades

2.2. Què és un robot?

Un robot és una mecanisme articulat capaç de realitzar accions complexes de manera automàtica, ja sigui mitjançant instruccions integrades en un computador en el propi dispositiu o bé rebudes a distància. [ii]

2.2.1. Arquitectura

A grans trets es pot distingir dues categories de robots principals, els robots mòbils i els fixes. Es cert que n'hi ha de més tipologies, però només aquestes són les que afecten al treball present i per tant les que s'ha tractat.

- Robots Mòbils

  Aquest tipus de robot, com el seu nom indica, es poden moure pel seu entorn ja que no estan fixats a un punt físic. Degut a la classificació generalista, aquesta classificació engloba una gran varietat de robots, ja que en aquesta categoria hi ha robots de tots els terrenys. Tant *Roomba* com el TurtleBot, els drons quadrocòpters, drons militars, Spot[iii] i inclús Tesla Bot[iv].

  En aquest treball el més destacable dels mencionats és el TurtleBot, ja que és un dels dos robots amb el que s'ha tractat en aquest treball.

- Robots Fixes

Els robots fixes d'àmbit industrial es poden classificar en 4 principals categories que es mostren a continuació.

- Cartesià

Aquesta tipologia de robot es basa en les coordenades cartesianes, que consten de tres eixos X, Y i Z. D'aquesta manera tots els punts on pot arribar el robot, concretament la punta de l'eina muntada al robot (anomenada TCP), s'especifica amb 3 valors, un per a cada eix.

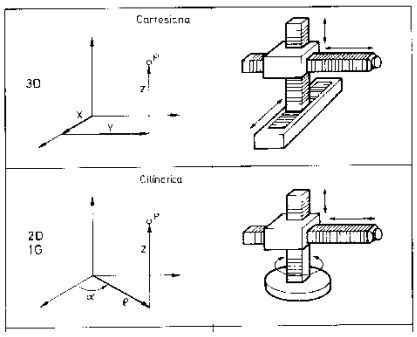


Figura 4: Eixos cartesians

Aquesta mena de robots s'utilitzen a petita escala per a la impressió 3D i a gran escala en aplicacions com ara el paletitzat, degut a la seva gran estabilitat estructural. Un desavantatge que presenten és que la relació de volum ocupat i volum de treball cobert molt baixa.

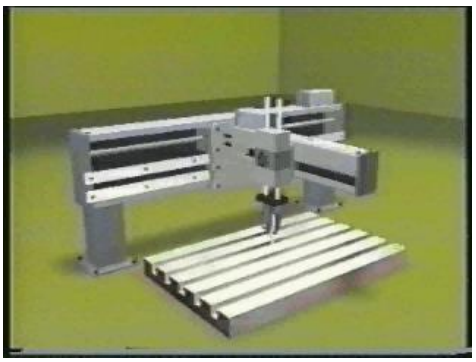


Figura 5: Robot cartesià

- Robot angular o antropomòrfic

Els robots angulars, també anomenats antropomòrfics degut a la seva semblança amb un braç humà, estan formats exclusivament per articulacions de rotació, d'aquí prové el nom angular. El seu posicionament doncs, es tracta amb coordenades angulars, una per a cada articulació que té el robot.

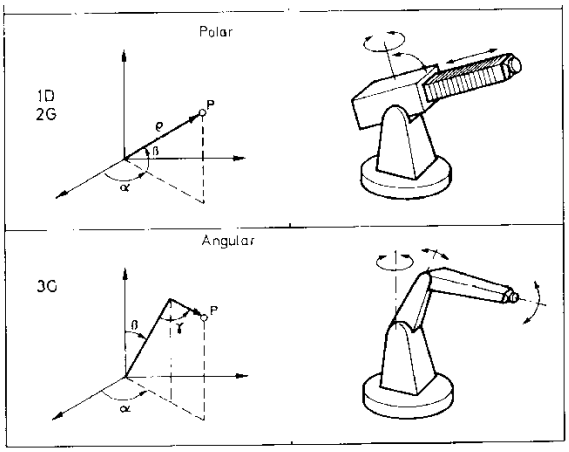


Figura 6: Eixos angulars

Degut a la seva alta maniobrabilitat és comú veure'l utilitzat per tasques com soldar, pintar, cargolar o transportar materials. Gràcies a les seves articulacions, pot realitzar trajectòries complexes amb facilitat, per això és apte per a les tasques esmentades prèviament. D'altre banda, aquestes mateixes articulacions impedeixen realitzar fàcilment trajectòries rectilínies.

Figura 7: Robot angular

- Robot SCARA

El robot SCARA esta conformat normalment per dues articulacions de rotació i una de translació. Les seves coordenades doncs consten de dues coordenades angulars i una longitudinal.

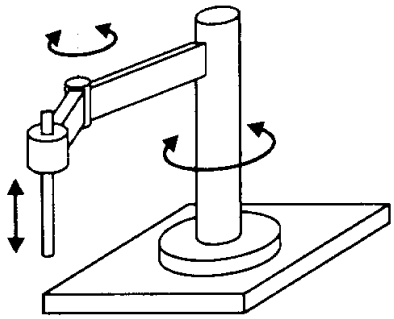


Figura 8: Robot SCARA

Normalment s'utilitzen per aplicacions com ara assemblatge o tasques d'inserció, degut a la seva velocitat en tots els eixos. A més, degut a la seva estructura presenten una gran estabilitat, tot i que el seu moviment és molt limitat.

- Robot Delta

Finalment tenim el robot Delta, el qual esta format per 3 o 4 cilindres que actuen conjuntament per situar el TCP en el lloc desitjat. La seva geometria i els seus actuadors permeten realitzar els moviments ràpidament, és per això que s'utilitzen majoritàriament per aplicacions de pick-and-place.

Figura 9: Robot Delta

2.2.2. Sensors

Per a la correcte realització de les tasques a desenvolupar d'un robot, és necessari rebre informació del entorn i dels processos que es desenvolupen. Els elements encarregats de proporcionar aquestes dades s'anomenen sensors.

Un sensor es un dispositiu capaç de mesurar magnituds físiques i transformar-les a una senyal elèctrica. Usualment fa falta transformar la senyal elèctrica per a que sigui útil, i per això cal un transductor. Un transductor és un dispositiu elèctric que condiciona la senyal d'entrada donada per un sensor i la converteix en un rang de voltatge o corrent pre-establert, com ara una senyal de 4-20mA[v], un valor analògic de 0-5V o bé una comunicació digital per protocol Modbus o I2C, per exemple.

De sensors hi ha molts tipus, però els rellevants per aquest treball són sobretot els de distància (LIDAR), les càmeres i sensors de força i desplaçament angular.

2.2.3. Actuadors

Els robots, en funció de les entrades que reben mitjançant els sensors actuen, es a dir, interactuen amb el seu entorn, i per això cap un element que mitjançant una senyal pugui activar-se, un actuador. Els actuadors no són res més que elements que permeten al robot realitzar accions o fer moviments. N'hi ha de diversos tipologies, en funció de la font de energia i e funció del moviment que realitzen.

Un motor rotatiu, per exemple, pot ser elèctric, hidràulic o pneumàtic, en funció de cada aplicació s'utilitza un o un altre. L'elèctric té un control molt més fàcil que no pas la resta, però no es poden utilitzar en entorns perillosos degut a les espurnes. El pneumàtic es pot utilitzar en entorns perillosos, però requereix una instal·lació pel subministrament d'aire i no pot fer tanta força. L'hidràulic també requereix una instal·lació, però pot fer molta més força que el pneumàtic, tot i que és força brut.

De actuadors podem trobar motors i cilindres principalment, cada un pot ser elèctric, pneumàtic o hidràulic, tal com s'ha esmentat anteriorment. Principalment es pot distingir entre actuador rotatius o lineals. Els actuador lineals pneumàtics o hidràulics s'anomenen cilindres mentre que els elèctrics s'anomenen motors lineals. S'utilitzen per accionar pinces d'un robot, les pales d'una excavadora o controlar la suspensió d'un vehicle.

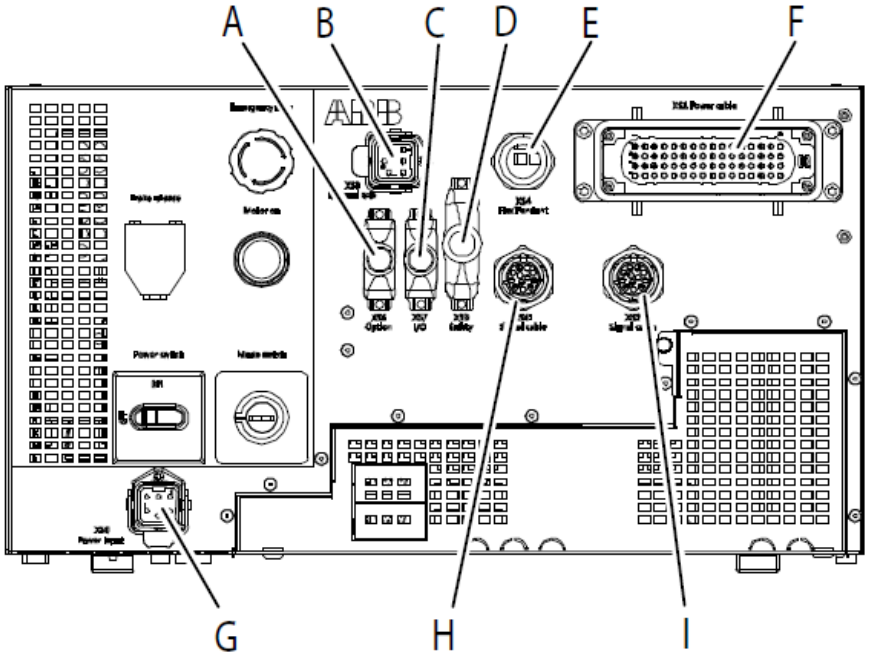
Hi ha diversos tipus de motors elèctrics, que s'utilitzen cada un en les aplicacions que més s'adeqüin. Un d'ells és el servomotor, també anomenat servo. Els servos són motors de corrent continua que a diferència d'aquests es poden mantenir en una posició fixa.[vi]
Tenen un sensor de posició que permet detectar la posició en cada instant, pel que és ideal per a fer sistemes precisos en el camp de la robòtica.

Figura 10: Servomotor

Els motors de corrent continua, tot i que generalment són molt mès precisos que els d'alterna, no són comparables en termes de precisió amb els servos. També hi ha motors pas a pas, motors també de corrent contínua tant precisos com els servos, utilitzats en robots cartesians com impressores 3D.

2.2.4. Controlador

El controlador és un petit computador que executa les ordres rebudes, ja sigui per un programa o manualment per l'operari, i les envia a cada motor del robot per realitzar els moviments. És imprescindible per a qualsevol operació, ja que és qui tradueix les ordres que rep en ordres específiques per a cada servo que conté el robot. En la *Figura 11* es pot veure el controlador del robot d'ABB IRC5, robot del qual es disposa al TecnoCampus.



xx0900000266

A	XS6 Opción
B	XS8 Eje adicional
C	XS7 E/S
D	XS9 Seguridad
E	XS4 FlexPendant
F	XS1 Cable de alimentación
G	XS0 Entrada de alimentación
H	XS41 Cable de señales
I	XS2 Cable de señales

Figura 11: Controlador IRC5

El controlador és l'element que governa tot el sistema i el més important, ja que sense la traducció dels moviments en instruccions individuals per a cada actuador no es poden realitzar tasques complexes.

L'ésser humà compren l'espai en coordenades cartesianes (esquerra o dreta, endavant o endarrere, amunt o avall), però per a certs robots no hi ha problema per a funcionar amb aquest tipus de coordenades, com per exemple una impressora 3d convencional.

Els braços robòtics, en canvi, tenen una arquitectura no cartesiana, per tant cada punt del espai amb orientació estarà definit per a coordenades articulars, les quals contenen la posició de cada articulació individual. Per tant, una trajectòria que un enginyer defineix, s'ha de traduir en un seguit de punts i trajectòries entre aquests punts, les quals es defineixen en coordenades articulars. Aquesta conversió de coordenades cartesianes a articulars la realitza el controlador mitjançant un seguit d'operacions matricials molt complexes.

2.2.5. Robots col·laboratius

Els robots col·laboratius, també anomenats cobots, son robots que estan dissenyats per a treballar conjuntament amb humans, ja sigui en un espai compartit o en una zona propera en la que hi podria haver contacte i, per tant, cal tenir en consideració la presencia humana. Degut a això, les aplicacions d'aquests robots varien de les dels robots tradicionals[vii].

Els cobots generalment estan construïts amb materials més lleugers i tous, amb cantonades rodones i control de velocitat i força[viii]. Per a operar amb seguretat, entre d'altres, s'utilitzen sensors de contacte i força, els quals en cas de contacte o excés de força aplicada prevenen que el robot continuï desenvolupant la seva tasca[ix].

Figura 12: Robot col·laboratiu

2.3. TurtleBot

El TurtleBot és un robot de baix cost de programari obert dissenyat per el mapatge de terrenys i per a la conducció autònoma. Hi ha dos models de robot, el Waffle i el Burger, en aquest treball s'utilitzarà el Burger, que es pot veure en la *Figura 13.*

Aquest model esta format per diferents nivells on es situen els diversos components electrònics. Aquests nivells estan apilats un sobre l'altre de forma semblant a una hamburguesa, d'aquí prové el nom.

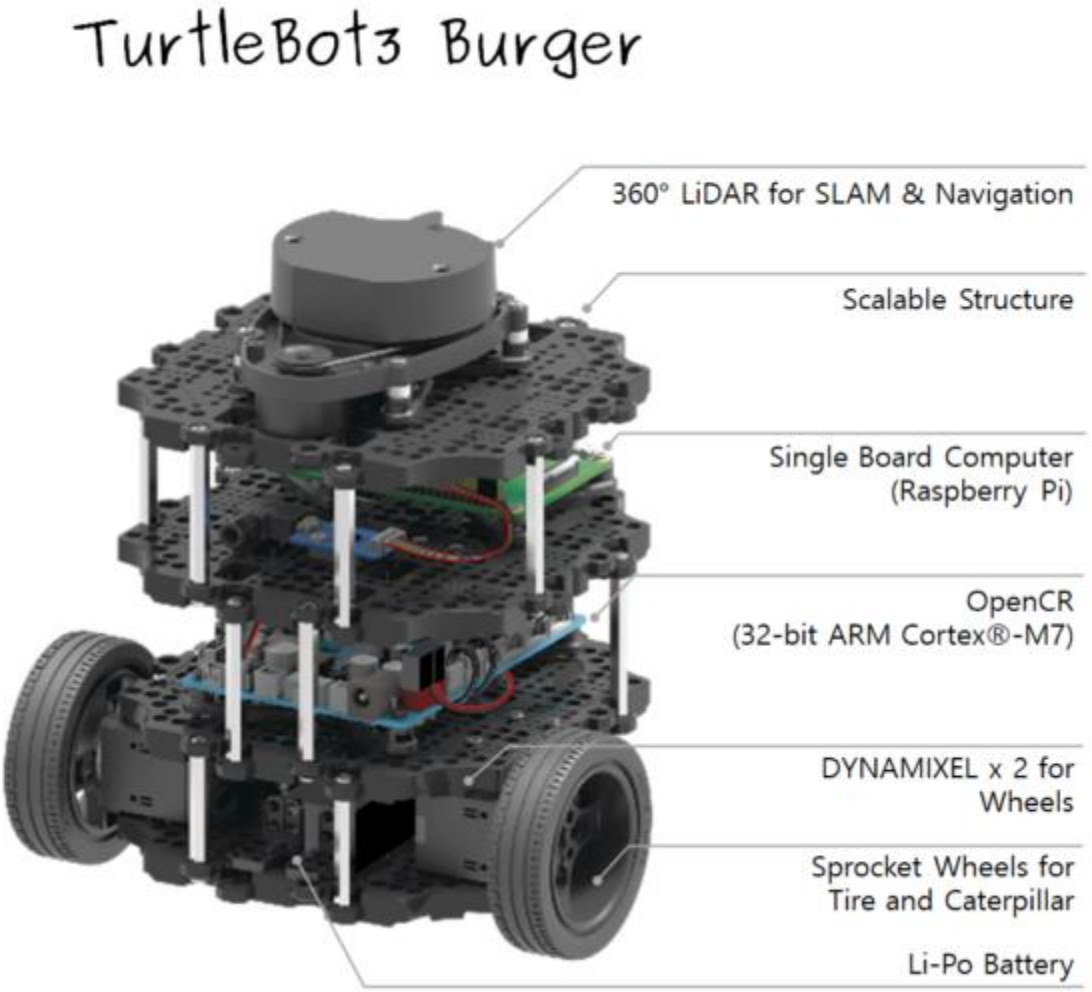


Figura 13: Esquema TurtleBot Burger

La Raspberry Pi és un petit computador d'una sola PCB, que en aquest cas utilitza una distribució de Linux coneguda com a Ubuntu. En aquesta aplicació la Raspberry s'utilitza com a controlador del robot, el qual rep les ordres mitjançant WiFi i trasllada la informació als sensors i actuadors corresponents, tracta les dades rebudes pels sensors i actua en base a elles.

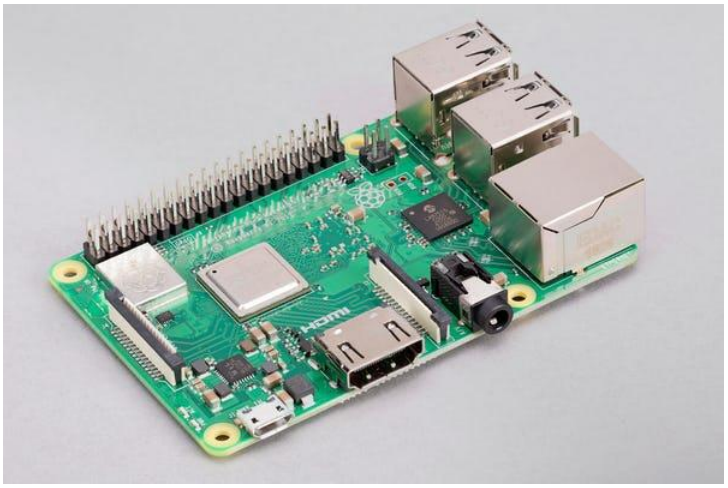


Figura 14: Raspberry Pi 3b+

Pel que respecta al moviment del robot, la Raspberry pi envia les ordres a la placa OpenCR, *Figura 15*. Aquesta, amb el software correcte, és responsable de traduir les ordres rebudes en senyals elèctriques que els motors entenen.

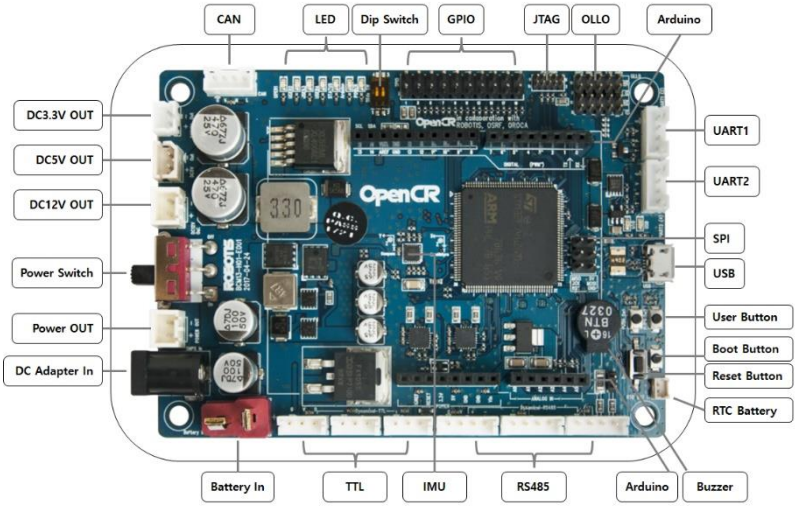


Figura 15: OpenCR

El TurtleBot utilitza un sensor LIDAR per a poder mapar un entorn i posteriorment navegar per ell. LIDAR és una tècnica per detectar distància mitjançant un làser i un receptor, el qual mesura la distància en funció del temps que tarda a rebre el làser rebotat. El sensor LIDAR compte amb un emissor i un receptor làser dins una carcassa que, tal i com es veu a la *Figura 14*, esta lligada a un politja per fer que giri a una velocitat controlada. Al fer això, es pot obtenir una gran quantitat de punts i distàncies que, mitjançant diverses tècniques informàtiques de tractament de dades, es pot generar un mapa del terreny per navegar evitant els obstacles.

Figura 16: Sensor LIDAR

La càmera és un mòdul que es connecta directament a la Raspberry i permet capturar imatges i vídeo. És un element essencial per poder emprar tècniques de visió artificial per localitzar objectes particulars i poder acostar-se.

Figura 17: Camera Raspberry Pi

2.4. TM5-900

Aquest és un robot col·laboratiu, això significa que esta dissenyat per treballar conjuntament amb humans, ja que té mecanismes de regular força.

El robot TM5-900 és un robot automòrfic amb 6 articulacions rotatives que permeten arribar una gran maniobrabilitat. Pot arribar fins a 900 mm de llargada, i amb tota la maniobrabilitat el fa ideal per moltes aplicacions industrials.

Figura 18: TM5-900

El robot duu 6 actuadors rotatoris els quals poden assolir unes velocitats a les articulacions d'entre 180 i 225 º/s. Tenen potència per aixecar fins a 4 Kg amb una velocitat de 1,4 m/s.

En quant a sensors, el robot consta d'una càmera a color de 5Mpixels molt útil per reconèixer objectes visualment i per exemple llegir codis QR. També consta amb un sensor de força per poder limitar la força que fa o per aplicar una força constant sobre una superfície.

2.5. ROS

ROS, Robot Operating System, és un conjunt de llibreries de software i eines de programari lliure dissenyades per a construir aplicacions robòtiques[x]. Al ser de programari lliure s'ha desenvolupat moltes llibreries per a funcionalitats diverses, des d'algoritmes de tecnologia punta fins a els elements més bàsics. És un element molt útil per a desenvolupar aplicacions robòtiques, perquè entre d'altres coses permet fàcilment interconnectar diverses maquines i programes.

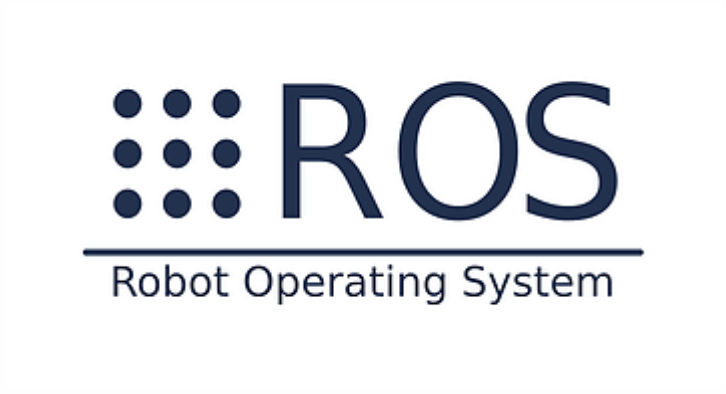


Figura 19: Logo de ROS

La seva estructura consisteix de diversos nodes interconnectats entre ells. El node principal, anomenat master, és l'encarregat de interconnectar tots els nodes, permetent així que intercanviïn informació entre ells.

Els nodes són els elements individuals que realitzen les tasques, per tant un robot controlat per ROS, com per exemple el TurtleBot, està controlat per diversos nodes. Un node controlarà el LIDAR, un altre la càmera i un altre els motors, per exemple.

Els nodes es comuniquen entre ells mitjançant diversos *tòpics* (temes). Aquests son els canals de comunicació que cada node crea perquè la resta de nodes que necessitin la informació puguin accedir-hi. Els *tòpics* funcionen a mode de publicació/subscripció, és a dir, un node publica certa informació en un *tòpic* i un altre node es subscriu a aquest *tòpic* per a rebre aquesta informació.

En la *Figura 20* es pot veure la finestra *rqt_graph*, un entorn gràfic on es pot veure gràficament els nodes, *tòpics*, i les relacions de subscripció entre nodes. Es pot veure, per exemple, el node */teleop_turtle* publicant en el *tòpic /turtle1/command_velocity*.

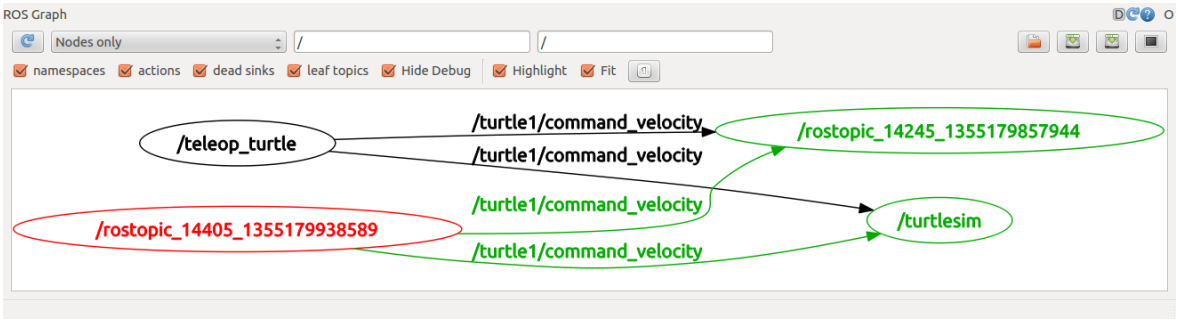


Figura 20: rqt_graph

Ros funciona mitjançant paquets, els quals es compilen mitjançant el sistema de construcció de ROS, Catkin. Catkin s'encarrega d'agafar el codi font dels paquets i compilar-los per al teu sistema. El funcionament és molt senzill, simplement cal tenir dins del entorn de treball una carpeta anomenada *src*, la qual ha de contenir tots els paquets que cal compilar. Seguidament cal aplicar la instrucció *catkin build* per a generar les carpetes que utilitzarà ROS.

3. Anàlisi de viabilitats tècniques

El TecnoCampus disposa dels dos models de TurtleBot, el Burger i el Waffle.

Per realitzar la programació final s'ha decidit que s'emprarà el model Burger, ja que l'altura és més gran i permet al braç robòtic no haver de fer tant de esforç per recollir la peça.

De braços robòtics, el TecnoCampus disposa de tres models, el ABB IRB120, el FANUC LR Mate 200iD/4S i el TM5-900.

Finalment s'ha escollit el braç robòtic TM5-900 degut a diversos factors.

Primerament s'ha tingut en compte l'edat del model, ja que el TM5 és molt més modern que els altres, i per tant és més realista al que es pot trobar actualment en una planta industrial en plena industria 4.0 o en transició.

Segonament, aquest robot té una càmera integrada que permet integrar visió artificial per trobar les coordenades exactes de la peça a recollir.

Finalment, cal mencionar que l'entorn de programació ROS permet més flexibilitat en quant a l'addició de mòduls complexes i interacció amb el TurtleBot.

4. Realització del projecte

En aquest treball s'ha desenvolupat un entorn de col·laboració entre dos robots amb l'objectiu de transportar i mecanitzar una peça emprant diverses tècniques de l'industria 4.0. Per al correcte funcionament de tot el sistema hi ha quatre parts fonamentals; el TurtleBot, el braç robòtic TM5, el PLC i el servidor.

El servidor és l'ordinador que coordina el PLC i ambdós robots mitjançant ROS, emprant de base un sistema operatiu Linux. Aquest, es connecta mitjançant ethernet amb el TM5, mentre que amb el TurtleBot utilitza WiFi. Pel que fa al PLC, degut a que no hi havia disponibles dos ports ethernet en el servidor s'ha connectar el cable ethernet en l'encaminador.

En la *Figura 21* es pot veure il·lustrat l'ecosistema descrit amb les seves connexions i les IPs de cada element. Cal notar com el servidor té dues direccions IP, la que utilitza la interfície ethernet, 192.168.2.103, i la que utilitza la interfície WiFi, 192.168.100.100, per comunicar-se amb el TM5 , el TurtleBot i el PLC.

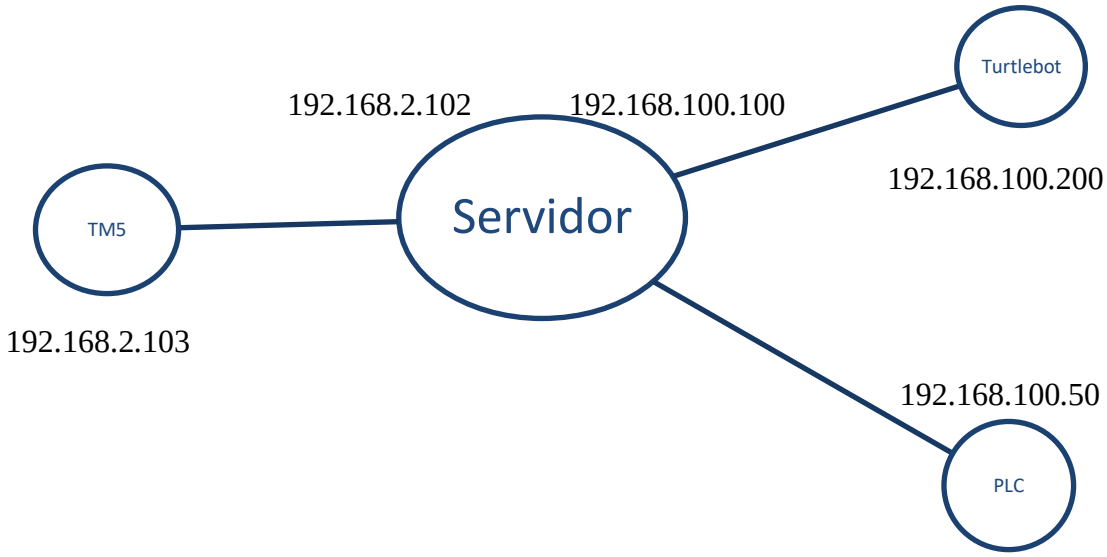


Figura 21: Diagrama ecosistema i IPs

4.1. Explicació de l'operativa general del sistema

La realització pràctica d'aquest treball es realitza en el laboratori 4 del TecnoCampus. El TurtleBot coneix l'entorn i es sap moure dins el laboratori, ja que prèviament s'ha realitzat un mapatge del entorn SLAM emprant el sensor LIDAR que duu el TurtleBot.

El robot, el qual conté la peça a transportar, parteix d'una posició inicial anomenada P0, en la quan roman fins que no s'inicií la seqüència. Quan s'indiqui, el TurtleBot anirà cap a la posició P1 tot evitant obstacles i enviarà una senyal per xarxa, indicant al TM5 que el TurtleBot ja es troba a la posició P1. El TM5, en rebre la senyal, realitzarà una imatge sobre la zona on es troba el TurtleBot i mitjançant tècniques de visió artificial trobarà les coordenades de la peça i procedirà a transportar-la cap a la posició de fresat Pf.

Un cop la peça es trobi a Pf, el TM5 enviarà una senyal al servidor, el qual mitjançant programació TCP amb Python controlarà el PLC de la cèl·lula de fabricació per tal de realitzar el fresat. Un cop el fresat estigui realitzat, el servidor enviarà una senyal al TM5 per tal de tornar la peça fresada a la posició P1 amb el TurtleBot.

Finalment, quan la peça es trobi amb el TurtleBot a la posició P1, el TM5 enviarà una senyal, amb la quan el TurtleBot iniciarà trajectòria evitant obstacles fins a la posició final P2. En la ***Error! No s'ha trobat l'origen de la referència.*** es pot veure un mapa representatiu del laboratori 4 amb les trajectòries que farà la peça.

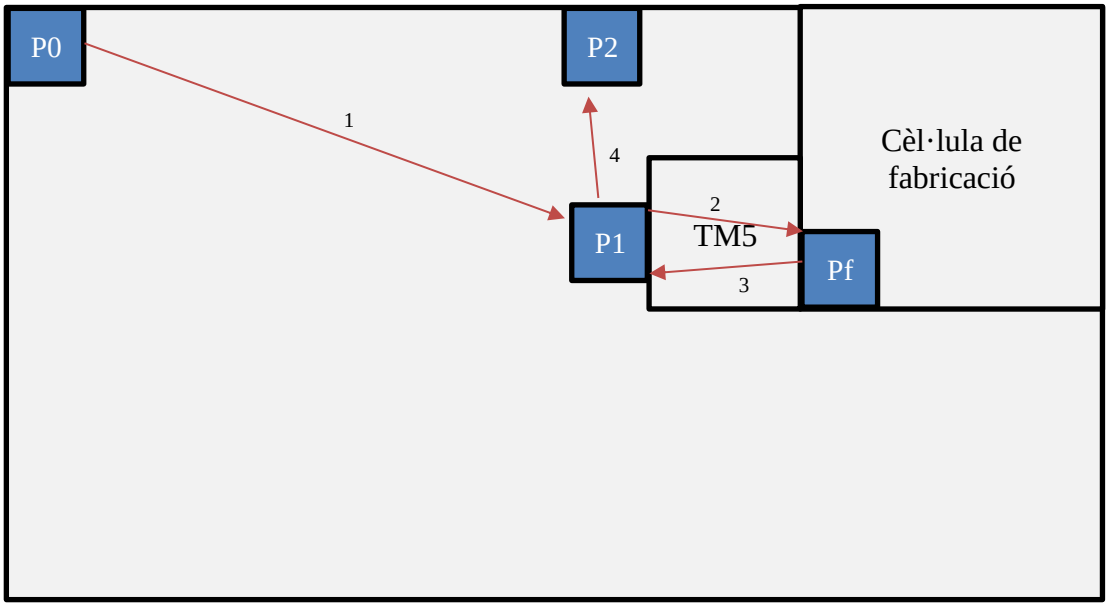


Figura 22: Esquema LAB 4

4.2. Preparació servidor

En aquest projecte s'ha utilitzat com a servidor un portàtil en el qual s'ha instal·lat Ubuntu 18.04, una distribució de Linux molt versàtil. Per a la instal·lació s'ha descarregat la imatge del SO en la web d'Ubuntu i s'ha seguit les instruccions de la web[1].

La versió d'Ubuntu s'ha escollit tenint en compte els factors limitants, que en aquest cas un d'ells és el TurtleBot, ja que conté la Raspberry 3b. Aquesta, tot i ser potent podria ser un factor limitant per a aquest ús, així que s'ha intentat utilitzar el software menys exigent. Cal tenir en compte que per programar tots dos robots mitjançant ROS, las versions de ROS han de ser les mateixes, i degut a que la versió mínima de ROS en la que ambdós robots poden operar és *Melodic*, s'ha utilitzat la versió 18 d'Ubuntu, ja que és la versió que funciona amb ROS Melodic.

Per a la instal·lació de ROS en el portàtil s'ha seguit la guia que proporciona Robotis, l'empresa que distribueix els kits del TurtleBot, a la seva web[2]. Primerament s'instal·la ROS Melodic mitjançant un script automàtic d'instal·lació que es baixa de GitHub.

```
1.  $ sudo apt update
2.  $ sudo apt upgrade
3.  $ wget https://raw.githubusercontent.com/ROBOTIS-
    GIT/robotis_tools/master/install_ros_melodic.sh
4.  $ chmod 755 ./install_ros_melodic.sh
5.  $ bash ./install_ros_melodic.sh
```

Seguidament, s'instal·len tots els paquets que funcionen conjuntament amb ROS.

```
1.  $ sudo apt-get install ros-melodic-joy ros-melodic-teleop-twist-joy \
2.    ros-melodic-teleop-twist-keyboard ros-melodic-laser-proc \
3.    ros-melodic-rgbd-launch ros-melodic-depthimage-to-laserscan \
4.    ros-melodic-rosserial-arduino ros-melodic-rosserial-python \
5.    ros-melodic-rosserial-server ros-melodic-rosserial-client \
6.    ros-melodic-rosserial-msgs ros-melodic-amcl ros-melodic-map-server \
7.    ros-melodic-move-base ros-melodic-urdf ros-melodic-xacro \
8.    ros-melodic-compressed-image-transport ros-melodic-rqt* \
9.    ros-melodic-gmapping ros-melodic-navigation ros-melodic-interactive-markers
```

[1] Index of /releases. (2022). Ubuntu. Recuperat l'11 de febrer de 2022, de https://old-releases.ubuntu.com/releases/

[2] Robotis e-Manual. (2022). Robotis. Recuperat l'11 de febrer de 2022, de https://emanual.robotis.com/

Finalment s'instal·len els paquets propis del TurtleBot per a poder controlar-lo.

```
1.  $ sudo apt-get install ros-melodic-dynamixel-sdk
2.  $ sudo apt-get install ros-melodic-turtlebot3-msgs
3.  $ sudo apt-get install ros-melodic-turtlebot3
```

Per a la configuració de xarxa, s'ha hagut de tenir en compte que la comunicació amb cada robot es realitza a través de dues interfícies de xarxa diferents. En el cas del TM5 es realitza mitjançant ethernet, mentre que en el cas del TurtleBot funciona per WiFi. En l'arxiu de configuració ~/.bashrc s'ha afegit les següent línies, les quals donen informació necessària per al funcionament del sistema.

```
1.  export ROS_MASTER_URI=http://192.168.18.128:11311
2.  export ROS_HOSTNAME=192.168.18.128
3.  export TURTLEBOT3_MODEL=burger
4.  export LDS_MODEL=LD-01
```

Es pot veure com s'indica quina és la IP del master, que en aquest cas és la pròpia, ja que el servidor és el master dels altres robots. Es pot veure com a nivell intern funciona mitjançant un servidor http i utilitza el port 11311 per a totes les comunicacions amb el master. També s'indica la IP pròpia en la segona línia, i es pot veure com és la mateixa que la del master.

Finalment, es pot veure com s'indica tant el model del robot com el model del sensor LIDAR. El model del robot s'utilitza per saber el radi que té el robot, així com altres variables físiques que poden afectar en quant al càlcul de trajectòries i altres aspectes de la navegació. Pel que fa al sensor, el driver que s'ha instal·lat accepta els dos models, però cal indicar quin ha de utilitzar.

Un cop preparat el servidor, per utilitzar ROS cal inicialitzar-lo mitjançant el comandament *roscore.* Aquest comandament inicialitza el ROS master, el node amb el qual tot node s'ha de comunicar per a fer qualsevol cosa.

4.3. Programació TurtleBot 3

4.3.1. Instal·lació i configuració inicial

Per a la preparació del TurtleBot primerament s'ha hagut d'instal·lar el sistema operatiu Ubuntu 18.04 en la Raspberry pi, el mateix SO que en el servidor, tot i que en aquest cas no ha fet falta una interfície gràfica, així que al no instal·lar-la i interactuar-hi únicament via terminal s'ha estalviat recursos, fent així que el TurtleBot funcioni més fluidament.
De la mateixa manera que el servidor, s'ha instal·lat ROS Melodic mitjançant un script.

```
1.  $ sudo apt-get update
2.  $ sudo apt-get upgrade
3.  $ wget https://raw.githubusercontent.com/ROBOTIS-
    GIT/robotis_tools/master/install_ros_melodic_rpi.sh
4.  $ chmod 755 ./install_ros_melodic_rpi.sh
5.  $ bash ./install_ros_melodic_rpi.sh
```

S'ha instal·lat els paquets necessaris per a controlar el hardware i s'ha creat la carpeta de l'entorn de treball catkin, necessari pel funcionament de ROS.

```
1.  $ sudo apt install ros-melodic-rosserial-python ros-melodic-tf
2.  $ mkdir -p ~/catkin_ws/src && cd ~/catkin_ws/src
3.  $ sudo apt install ros-melodic-hls-lfcd-lds-driver
4.  $ sudo apt install ros-melodic-turtlebot3-msgs
5.  $ sudo apt install ros-melodic-dynamixel-sdk
6.  $ git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3.git
7.  $ cd ~/catkin_ws/src/turtlebot3
8.  $ rm -r turtlebot3_description/ turtlebot3_teleop/ turtlebot3_navigation/ turtlebot3_slam/
    turtlebot3_example/
9.  $ cd ~/catkin_ws/
10. $ echo 'source /opt/ros/melodic/setup.bash' >> ~/.bashrc
11. $ source ~/.bashrc
12. $ cd ~/catkin_ws && catkin_make -j1
13. $ echo 'source ~/catkin_ws/devel/setup.bash' >> ~/.bashrc
14. $ source ~/.bashrc
```

S'ha creat normes per a donar autorització a ROS per a emprar els dispositius USB sense haver de fer-ho manualment cada cop.

```
1.  $ rosrun turtlebot3_bringup create_udev_rules
```

Per acabar, s'ha instal·lat el driver del sensor LIDAR i per finalitzar la instal·lació s'ha tornat a construir l'entorn de treball catkin amb els nous paquets.

```
1.  $ sudo apt update
2.  $ sudo apt install libudev-dev
3.  $ cd ~/catkin_ws/src
4.  $ git clone -b develop https://github.com/ROBOTIS-GIT/ld08_driver.git
5.  $ cd ~/catkin_ws && catkin_make
```

Per a la configuració de xarxa, de la mateixa manera que el servidor, s'ha hagut d'editar l'arxiu de configuració ~/.bashrc i afegir aquestes línies.

```
1.  export ROS_MASTER_URI=http://192.168.18.128:11311
2.  export ROS_HOSTNAME=192.168.18.236
3.  export LDS_MODEL=LDS-01
```

Finalment, s'ha de recarregar les variables de l'entorn.

```
1.  source ~/.bashrc
```

De forma similar al servidor, es pot veure com en l'arxiu s'ha afegit la IP del master, la pròpia i el model de robot i sensor LIDAR. La IP del master es pot veure com és exactament la mateixa que en l'arxiu de configuració del servidor, ja que tots els nodes del sistema han de apuntar cap al master, sinó no funcionaria. La IP del *hostname*, és a dir la pròpia, es pot veure doncs com canvia, ja que aquesta és la del TurtleBot via WiFi. De la mateixa forma i pels mateixos motius que en el cas anterior, s'ha afegit la informació del sensor LIDAR.

4.3.2. Mapeig inicial

Un element molt important per a que el projecte funcioni correctament és el mapa del terreny sobre el que es mourà el TurtleBot. Tenint en compte que no es una tasca de exploració el projecte final, fa falta tenir un mapeig del terreny prèviament al desenvolupament del treball. Per aquest motiu s'ha realitzat un mapa del laboratori 4 del TecnoCampus.

Per començar, s'ha iniciat el ROS master en el servidor, el qual permet a altres nodes que s'hi connectin i compartir informació, és el pas inicial per a qualsevol acció amb ROS.

```
1.  roscore
```

Des del servidor, s'ha iniciat una comunicació *ssh* amb el TurtleBot. Aquesta permet iniciar certs programes en el TurtleBot de manera remota des del servidor. Seguidament s'ha iniciat tots els sensors del robot això com les rutines de calibratge inicial d'aquests. Això prepara al robot per a operació, ja que a part de calibrar els sensors, crea i connecta el node del TurtleBot i els seus sensors amb el ROS master.

```
1.  ssh ubuntu@192.168.0.200
```

```
2.  roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

Seguidament, s'inicia també des del TurtleBot el mapeig SLAM, una tècnica de mapeig que els paquets de TurtleBot ROS inclouen. Aquest permet al robot utilitzar el sensor LIDAR per a crear un mapa de l'entorn on es troba. Cal doncs obrir un altra terminal, establir un altre connexió SSH i iniciar el SLAM. En la *Figura 23* es pot veure el la finestra d'rviz l'inici del mapeig.

```
1.  ssh ubuntu@192.168.0.200
2.  roslaunch turtlebot3_slam turtlebot3_slam.launch
```

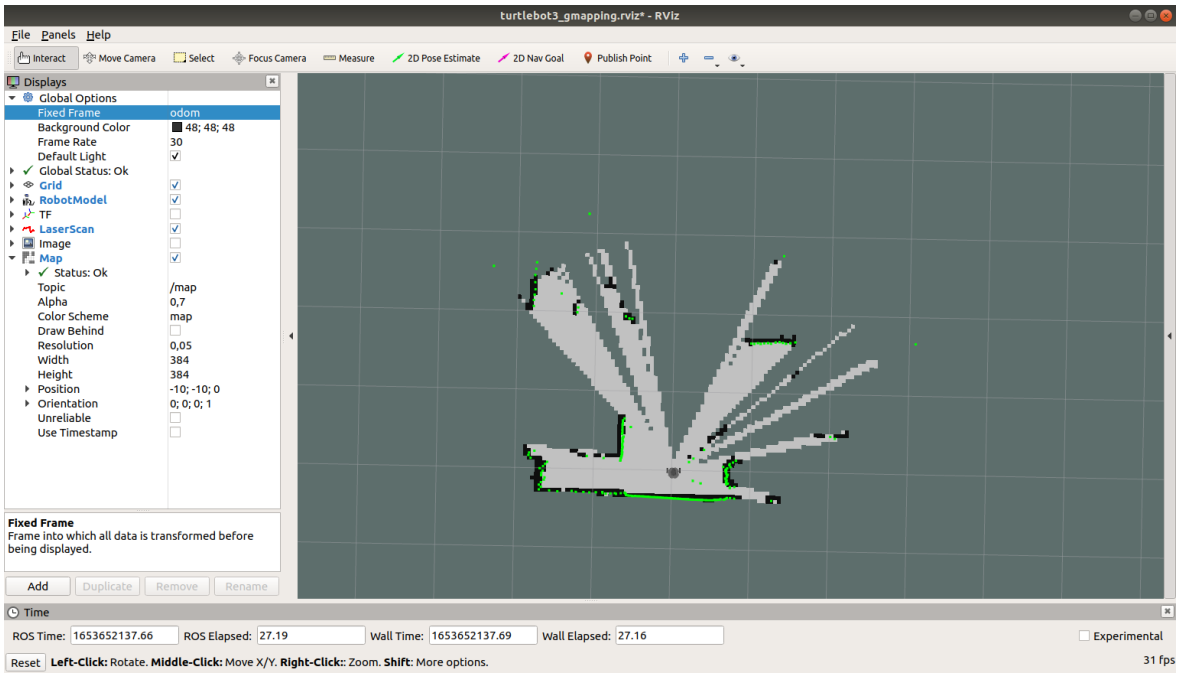


Figura 23: Inici mapejat

Un cop el mapeig s'ha iniciat, per a poder mapar tot l'entorn, cal que el robot observi la sala des de diferents posicions, ja que en cas de quedar-se fix, els objectes que es trobin més propers fan ombra als més llunyans. Per aquest motiu el TurtleBot s'ha de anar movent per a la sala a mapar. Això és pot realitzar fàcilment controlant manualment al robot mitjançant el teclat escrivint aquest comandament en el terminal del servidor.

```
1.  roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

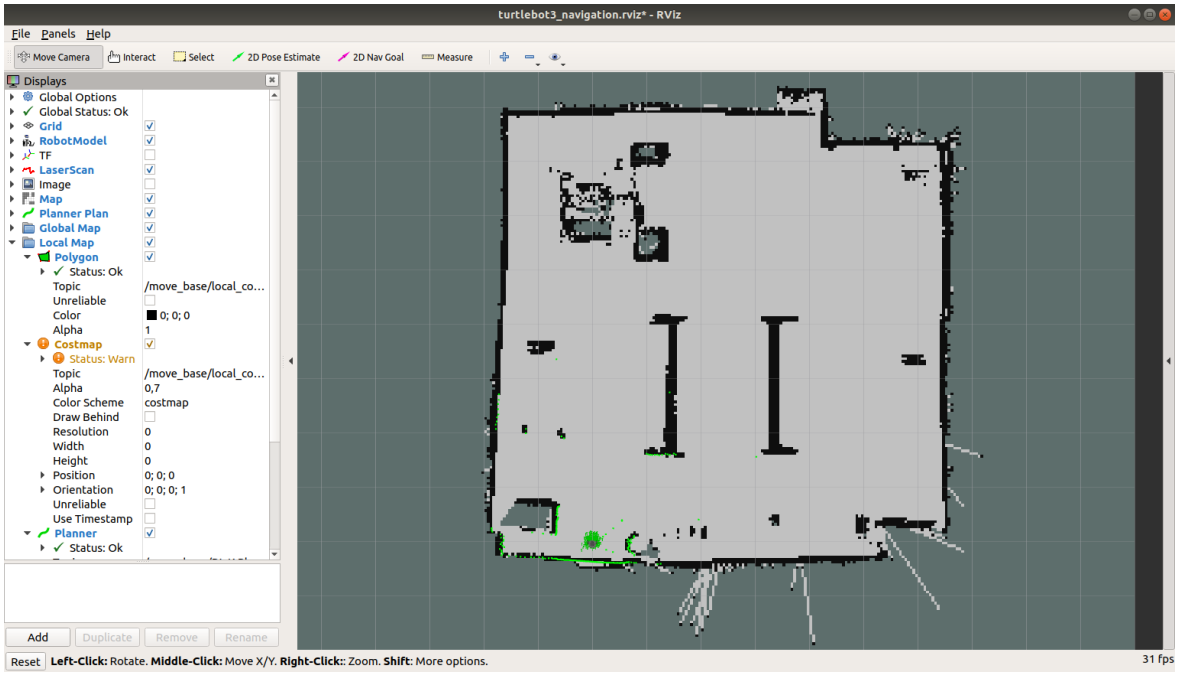


Figura 24: Mapa complert LAB 4

Finalment, un cop el mapa està complert després d'haver-se mogut per a la sala, obtenim la *Figura 24.* Seguidament, des del servidor es guarda el mapa per a la posterior navegació per a la sala.

```
1.  rosrun map_server map_saver -f ~/map
```

Pot passar que durant el mapatge al TurtleBot li rellisquin una mica les rodes, i això causa que el robot es pensi que està avançant quan realment no ho esta fent. En petites quantitats no és res preocupant, ja que el propi motor de navegació té un sistema de correcció d'errors per a petites desviacions. Es pot veure en la *Figura 24* com les parets del laboratori no són del tot rectes degut a l'efecte mencionat prèviament. Tot i així, com la desviació és molt petita no afecte en res i la navegació és exitosa. En grans quantitats però, aquesta desviació pot inutilitzar el mapa, tal i com es pot veure en la *Figura 25*.

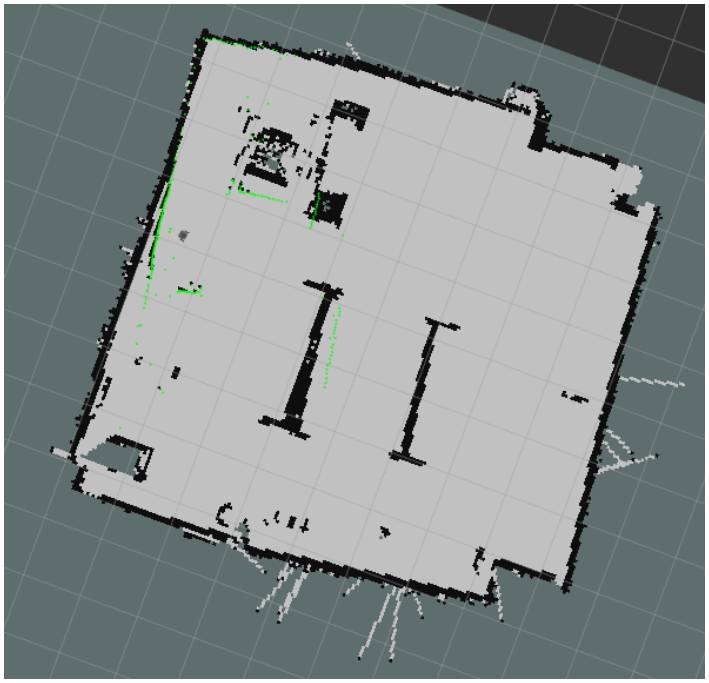


Figura 25: Mapa fallit

4.3.3. Navegació

Un cop s'ha obtingut un mapa, es pot carregar amb la següent instrucció.

```
1.  roslaunch turtlebot3_navigation TurtleBot3_navigation.launch map_file:=$HOME/map.yaml
```

Aquest comandament obre una pestanya de l'entorn RViz on es pot veure el mapa prèviament guardat, així com la detecció actual del sensor LIDAR. Per a poder navegar correctament, cal donar una posició estimada del TurtleBot en el mapa, per així saber a on es troba dins el mapa. Això es pot fer mitjançant el botó indicat en la *Figura 26*.

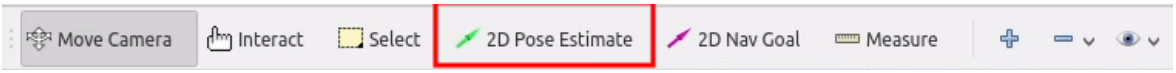


Figura 26: Menú navegació RViz. 2D Pose Estimate

Per a realitzar una tasca de forma automàtica però, no és pràctic haver de realitzar manualment aquest pas. Es per això que en aquest treball s'ha realitzat el posicionament inicial del robot de forma que es pot fer automàticament. De la manera en que opera el mapeig, el TurtleBot parteix de la posició inicial (X:0, Y:0), i a partir d'aquí va afegint terreny. Quan s'inicia la navegació doncs, el programa sempre suposa que es parteix de la posició inicial, i en cas que no sigui així es pot indicar aquesta tal i com s'ha vist prèviament.

Degut a aquest comportament, en aquest treball per a fer el mapeig s'ha posicionat el TurtleBot en una posició coneguda, des de la qual el TurtleBot parteix quan el sistema està operatiu. D'aquesta manera sempre que s'inicii la navegació des d'aquesta posició no fa falta indicar al robot a on es troba relatiu al mapa degut a que ja assumeix que parteix de la posició inicial. En la *Figura 27* es pot veure com el mapa i l'entorn detectat pel LIDAR concorden degut a que parteix de la posició inicial, en canvi, en la *Figura 28* no concorden ja que parteix d'una altre posició.

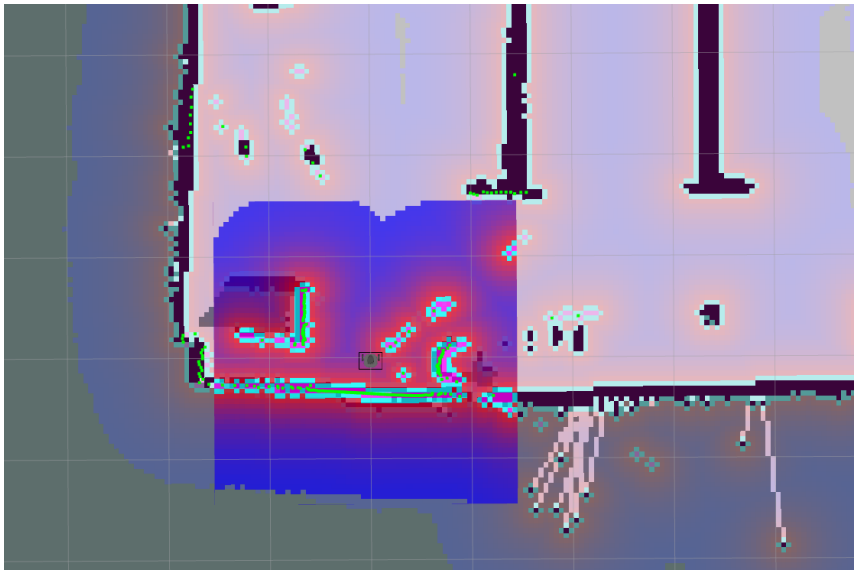


Figura 27: Inici navegació des de la posició inicial

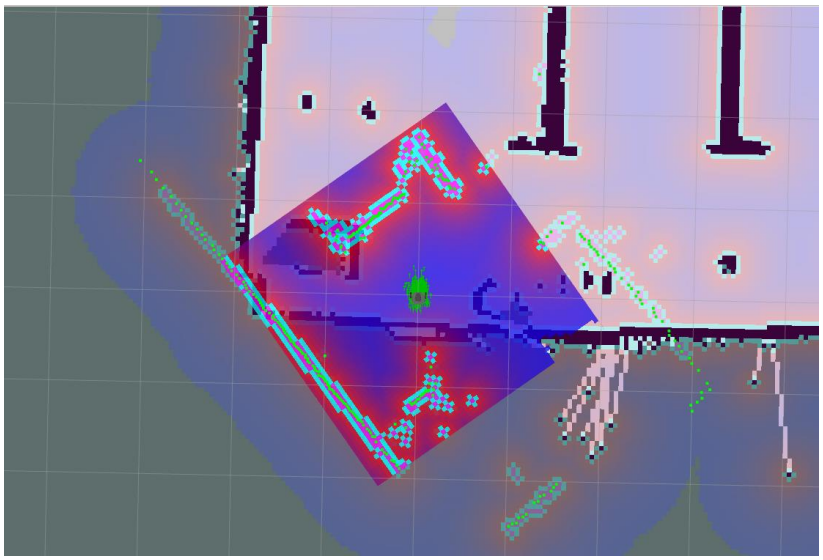


Figura 28: Inici navegació amb posició incorrecte

Cal indicar però que no cal patir per a petites desviacions de la posició inicial del TurtleBot, ja que la navegació del sistema compte amb un sistema de correcció d'errors, així que en el cas que hi hagués un petit error inicial, a mesura que el robot avancés aquest error s'aniria reduint fins a ser negligible.

Un cop resolt el posicionament inicial, el robot pot començar a navegar pel mapa. Mitjançant rviz es pot en el mapa donar un objectiu de posició amb el botó que es mostra en la *Figura 29*.

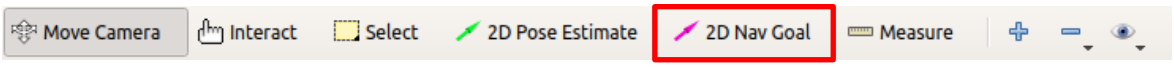


Figura 29: Menú navegació RViz. 2D Nav Goal

Un cop rebut un objectiu, el robot calcularà una trajectòria per arribar-hi per la via més ràpida, tot evitant obstacles que hi puguin aparèixer tot i no estar mapats. La trajectòria es pot veure en el mapa com a una línia negre, tal i com es pot veure en la *Figura 30*.

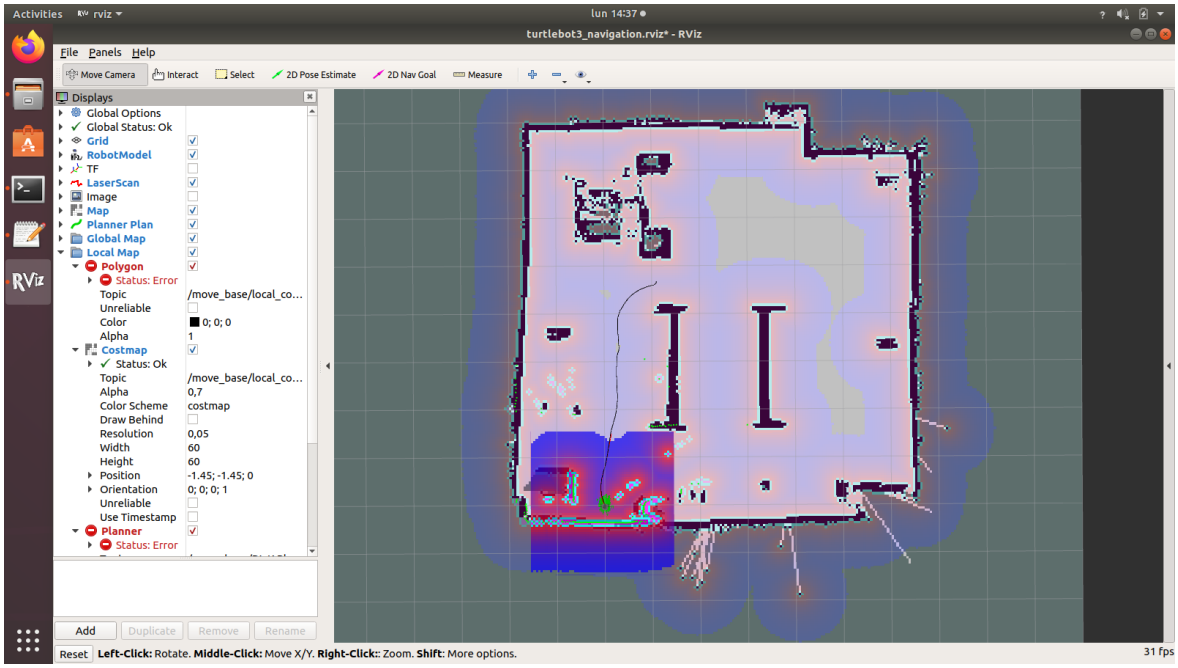


Figura 30: Trajectória TurtleBot

De la mateixa manera que amb el posicionament inicial, per a poder realitzar la tasca automàticament s'ha hagut d'utilitzar una manera alternativa de comunicar-li al robot el seu objectiu. RViz el que fa és únicament oferir una interfície gràfica per a poder visualitzar l'entorn així com interactuar-hi de manera còmode, però per a indicar la posició objectiu, un cop seleccionada en el mapa, publica la posició en els *tòpics* corresponents. Aquest comportament s'ha recreat mitjançant codi, en aquest cas amb Python, indicant la posició i orientació i publicant-la a on correspongui.

4.3.4. Suport peça

Per a que el TurtleBot pogués transportar la peça, s'ha dissenyat un suport mitjançant l'eina Fusion 360, un software molt potent de disseny en 2 i 3D, i s'ha produït mitjançant la talladora làser que disposa el TecnoCampus amb unes planxes de fusta. El suport ha d'encaixar en la part posterior del robot, per lo qual s'ha tingut en compte la forma dels suports centrals així com quatre forats per a una bona subjecció.

El suport esta format per dues lamines de fusta encolades l'una amb l'altre, les dues amb el mateix contorn. La inferior no té cap forat tret dels 4 petits dissenyats pels caragols. La superior però, consta d'un forat circular del diàmetre de la peça a transportar. A més, aquesta consta d'un quadrat gravat sobre la fusta. Aquest serveix per a alinear correctament el *landmark*, una enganxina amb un disseny dissenyat per a ser detectat mitjançant visió artificial. Així doncs, amb les dues làmines i el forat, es genera un encaix per a que la peça reposi en el suport de fusta. Ha calgut llimar el tall del forat per a augmentar la tolerància, ja que sinó no sempre entrava amb facilitat. En la *Figura 31* es pot veure el resultat final del suport muntat ja al TurtleBot.

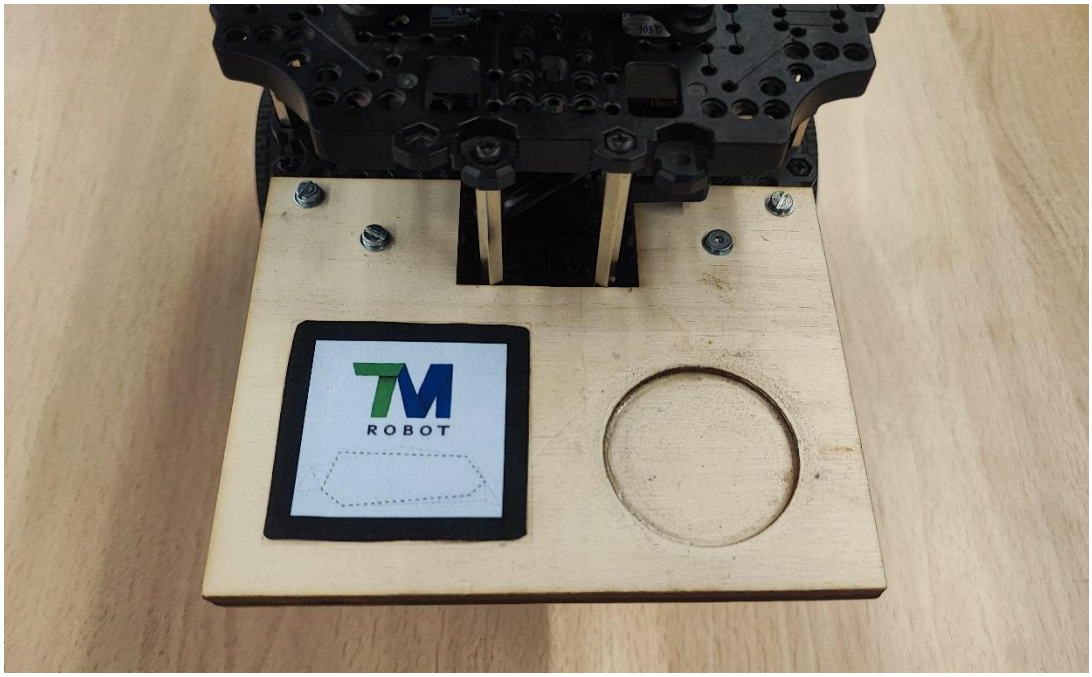


Figura 31: Suport TurtleBot

4.4. Programació TM5-900

4.4.1. Configuració inicial

Per a la programació del TM5-900 mitjançant ROS s'utilitza el driver proporcionat per a la pròpia empresa *Techman Industries*, el qual es pot descarregar mitjançant GitHub[3]. Cal clonar el repositori en la carpeta /*src* del l'entorn de treball i compilar els paquets mitjançant els següents comandaments.

```
1.  cd /catkin_ws/srs
2.  git clone https://github.com/TechmanRobotInc/tmr_ros1.git
3.  cd ..
4.  catkin build
```

Un cop l'entorn de treball s'hagi compilat, cal connectar el servidor amb el robot mitjançant un cable ethernet. Des de la configuració es va a l'apartat de *Network*, on es pot veure que la ha aparegut una nova connexió. A dins es pot veure la IP del robot i la mascara de xarxa amb la que treballa.

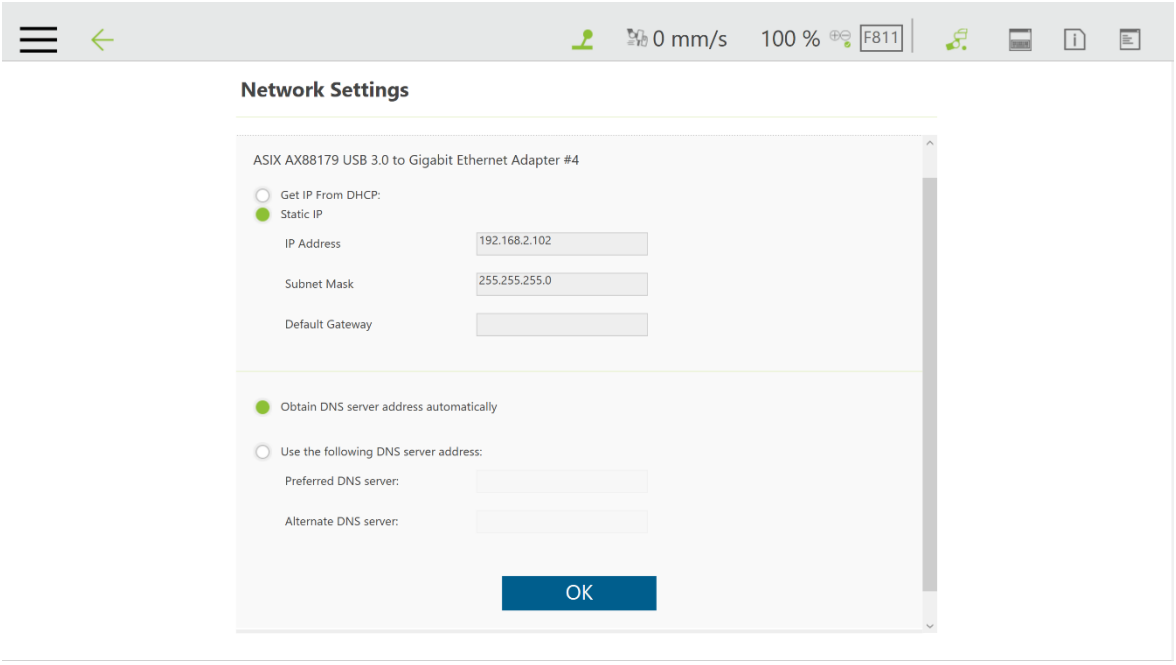


Figura 32: Configfuració de xarxa TM5

[3] GitHub - TechmanRobotInc/tmr_ros1: techman robot ROS1 driver (experiment). (2022). GitHub. Recuperat l'11 de febrer de 2022, de https://github.com/TechmanRobotInc/tmr_ros1

Per a que el robot i el servidor es puguin comunicar cal que estiguin en la mateixa xarxa, per això cal que comparteixin mascara de xarxa i els tres primers valors de la adreça IP, és a dir la IP ha de ser segons el format 192.168.2.X, sent X un valor entre 0 i 255 a excepció de 102, que ja està en ús. Finalment s'ha utilitzat 192.168.2.103 com a la IP del servidor.

A continuació s'ha activat la transmissió de certes variables a través de xarxa. Això dona accés a tot tipus de variables del robot com sortides i entrades tant digitals com analògiques, posició del TCP, força que realitza cada articulació i moltes altres més. Això és necessari ja que sense aquest accés no es disposaria de la informació essencial per a aplicar el control del robot. A continuació es pot veure el llistat complert de les variables.

- Robot_Error
- Project_Run
- Project_Pause
- Safeguard_A
- ESTOP
- Camera_Light
- Error_Code
- Joint_Angle
- Coord_Robot_Flange
- Coord_Robot_Tool
- TCP_Force
- TCP_Force3D
- TCP_Speed
- TCP_Speed3D
- Joint_Speed
- Joint_Torque
- Project_Speed
- MA_Mode
- Robot Light
- Ctrl_DO0~DO7
- Ctrl_DI0~DI7
- Ctrl_AO0
- Ctrl_AI0~AI1
- END_DO0~DO3
- END_DI0~DI2
- END_AI0

4.4.2. TMDriver i MoveIt

L'element que permet la comunicació entre ROS i el robot és el driver. Aquest es l'encarregat d'establir la comunicació entre les dues parts, així com enviar les instruccions pertinents al robot per a que siguin executades, s'executa de la següent manera.

```
rosrun tm_driver tm_driver 192.168.2.102
```

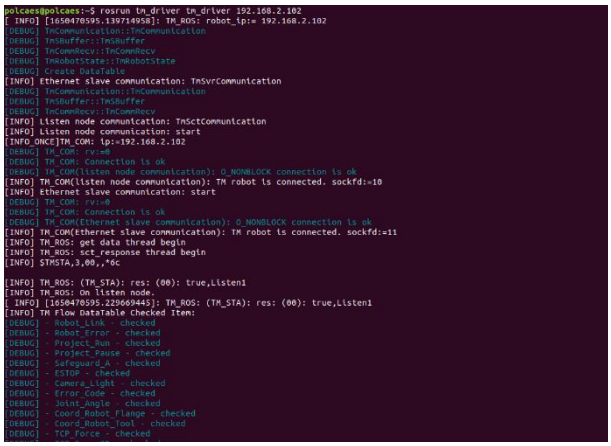


Figura 33: Output tm_driver

Un cop el driver ha establert la comunicació ja es pot interactuar amb el robot. El primer moviment que s'ha realitzat en aquest treball ha estat mitjançant un script de demostració inclòs en el propi driver. Aquest està escrit amb Python i en ell es controla el robot mitjançant les coordenades angulars de cada articulació, és a dir, l'angle que té cada motor.

Tal com s'ha dit en l'apartat *2.2.4*, les coordenades articulars no són ideals per a interactuar amb els braços robòtics, i per tant aquesta solució que planteja el script anterior no és vàlida per aquesta aplicació. És per això que per al desenvolupament d'aquest treball s'ha utilitzat un programari de ROS anomenat MoveIt[4] per a poder fer el càlcul de trajectòries i així poder tractar amb coordenades cartesianes per interactuar fàcilment amb el robot i realitzar moviments complexes.

[4]MoveIt Motion Planning Framework. (2022). MoveIt. Recuperat l'11 de febrer de 2022, de https://moveit.ros.org/

MoveIt:

Amb el següent comandament s'inicia el software MoveIt amb RViz i es carreguen els arxius 3D del robot per a poder evitar col·lisions amb si mateix.

```
roslaunch tm5_900_moveit_config tm5_900_moveit_planning_execution.launch sim:=False
robot_ip:=192.168.2.102
```

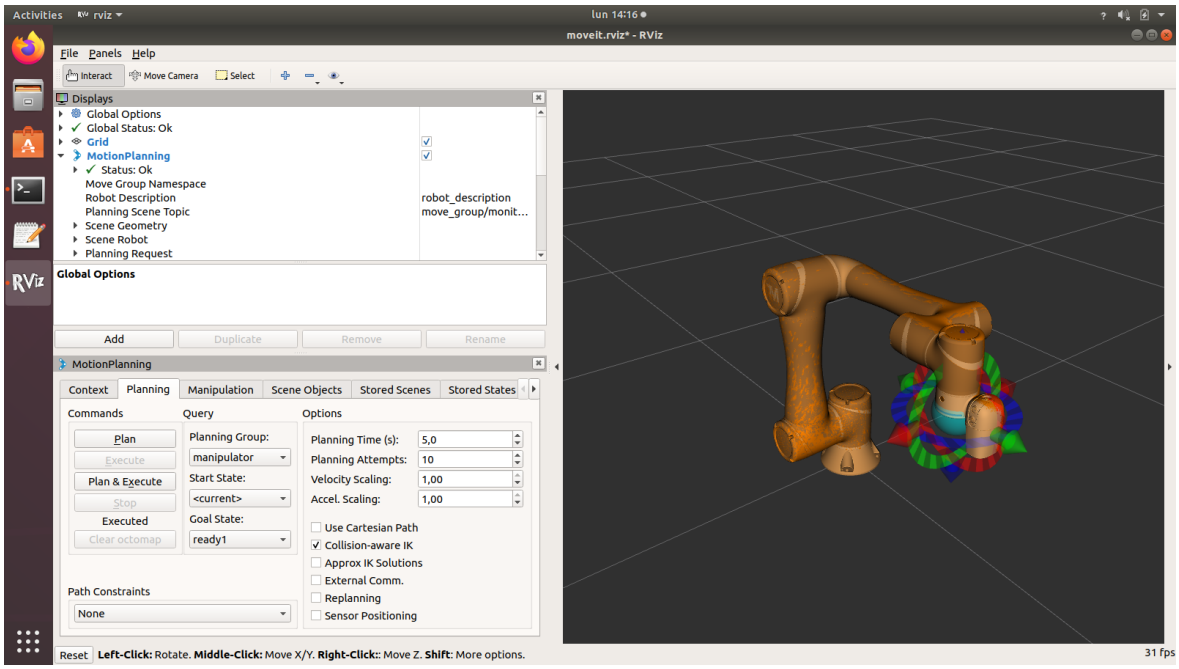


Figura 34: TM5 MoveIt RViz

Com es pot veure en la *Figura 34*, en la pantalla RViz es pot veure la representació del braç robòtic en la seva posició actual. Des d'aquesta interfície es pot escollir d'entre les diverses posicions preprogramades així com moure el TCP cap a la posició que es desitgi mitjançant les fletxes que es veuen. Seguidament es prem el botó *plan* per a planificar la trajectòria, la qual cosa permet veure en pantalla mitjançant un robot gris la trajectòria que seguirà el robot. Finalment es prem el botó *Execute* per a executar la trajectòria prèviament planificada, i així realitzar el moviment en el robot físic.

El node *move_group* és el node principal de MoveIt. S'encarrega de rebre les ordres i realitzar-les, tot llegint les entrades que rep. Com es pot veure en la *Figura 35,* per interactuar-hi mitjançant una interfície gràfica s'utilitza el plugin d'rviz que hem vist anteriorment. Això però, no permet automatitzar el procés, per això cal poder interactuar-hi amb codi. Hi ha dues opcions de llenguatge de programació per utilitzar com a interfície, Python i C++. Degut a la facilitat que presenta Python envers C++ i la experiència prèvia amb aquest llenguatge, s'ha decidit utilitzar la interfície Python anomenada *moveit_commander*.

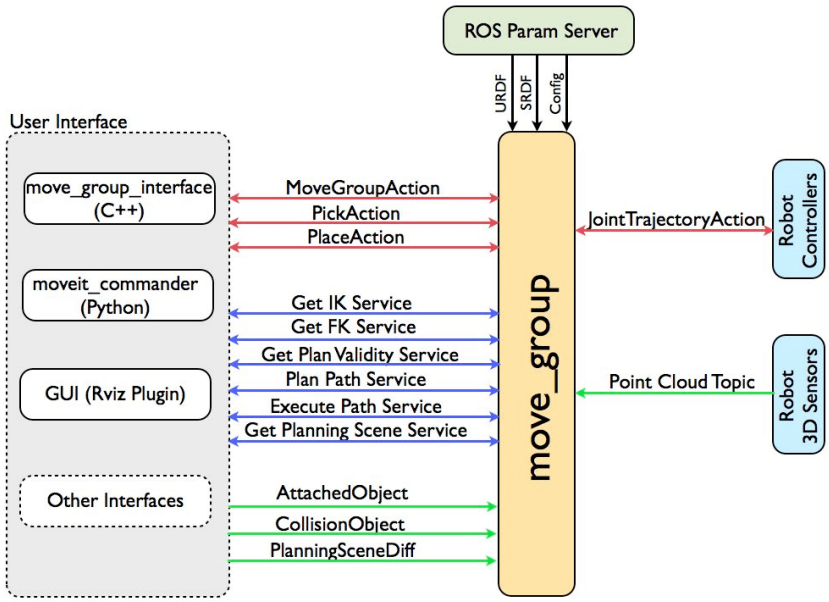


Figura 35: Estructura move_group

Per a realitzar tots els moviments amb el braç robòtic s'ha utilitzat de base el tutorial de *moveit_commander*. S'ha utilitzat moviments lineals, moviments per posició de les articulacions i moviments per coordenades, es pot veure el codi en els annexes.

4.4.3. Limitacions físiques

Degut al espai en qué el robot ha d'operar ha de tenir en compte el seu entorn per a no topar amb cap obstacle. En la *Figura 36* i la *Figura 37* es pot veure com hi ha obstacles que poden entorpir la operació del robot, entre d'ells el tub de cables, la base del robot, els soports de l'estació de mecanitzat i el robot fànuc, entre d'altres.

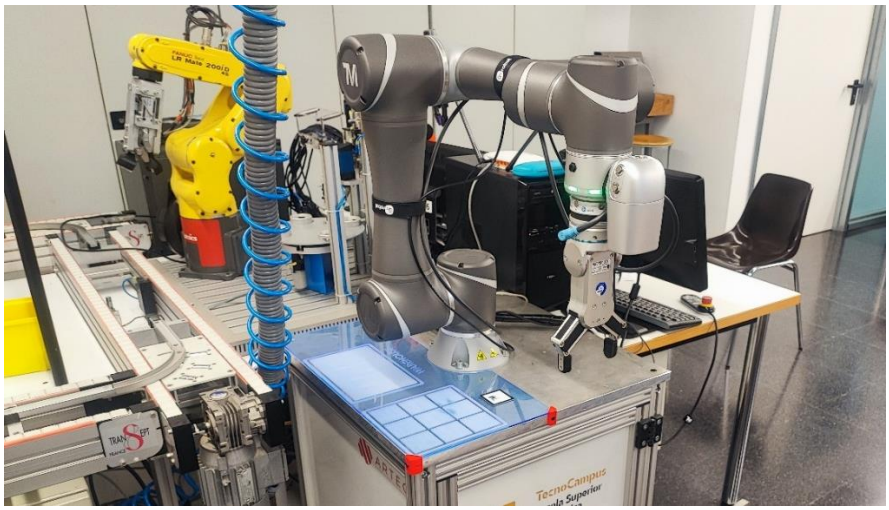


Figura 36: Entorn TM5

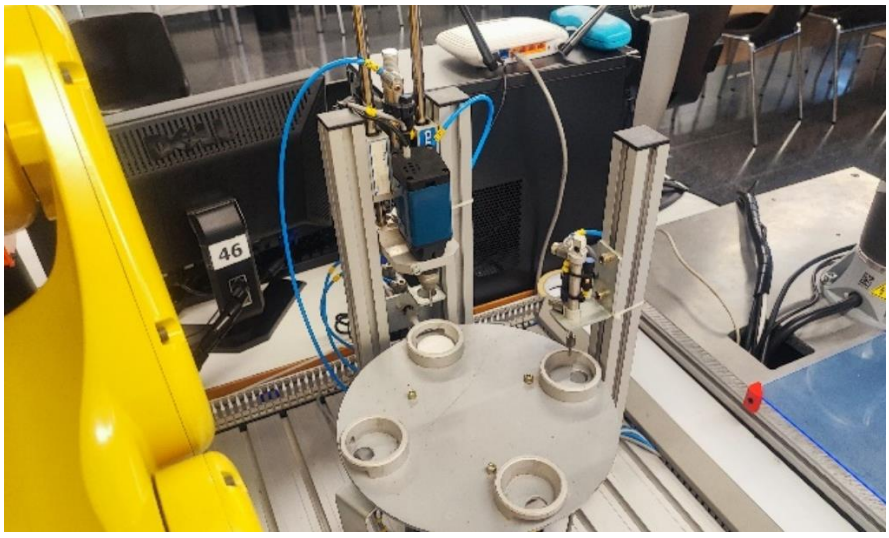


Figura 37: Entorn estació de mecanitzat

Degut al entorn complicat en el que ha d'operar el robot, ha calgut la inserció de barreres virtuals al controlador del robot per a poder tenir en compte els obstacles en la planificació de trajectòries. Això s'ha realitzat mitjançant el node *move_group*, definint una escenografia d'obstacles conformat de diferents figures geomètriques.

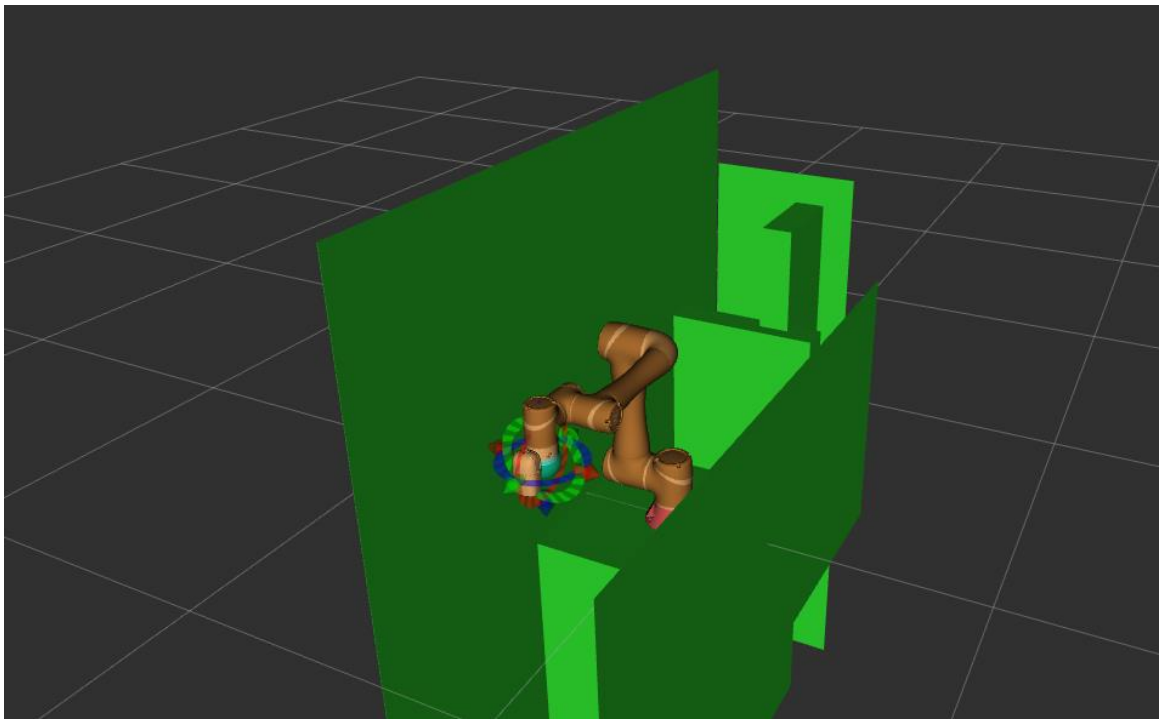


Figura 38: Limitacions físiques TM5

4.4.4. Visió artificial

En aquest treball s'ha emprat visió artificial per a obtenir les coordenades de la peça a transportar, un cilindre, ja que la seva posició no es fixa. Per a realitzar la visió artificial s'ha utilitzat la tasca de visió que proporciona el software TMFlow, ja que amb ROS la visió artificial del robot no és compatible.

Inicialment s'ha intentat determinar la posició de la peça mitjançant el color, però produïa un error considerable. Primerament s'inicia la càmera i es configuren els paràmetres de llum, temps d'exposició i diversos altres, obtenint una imatge nítida del objecte.

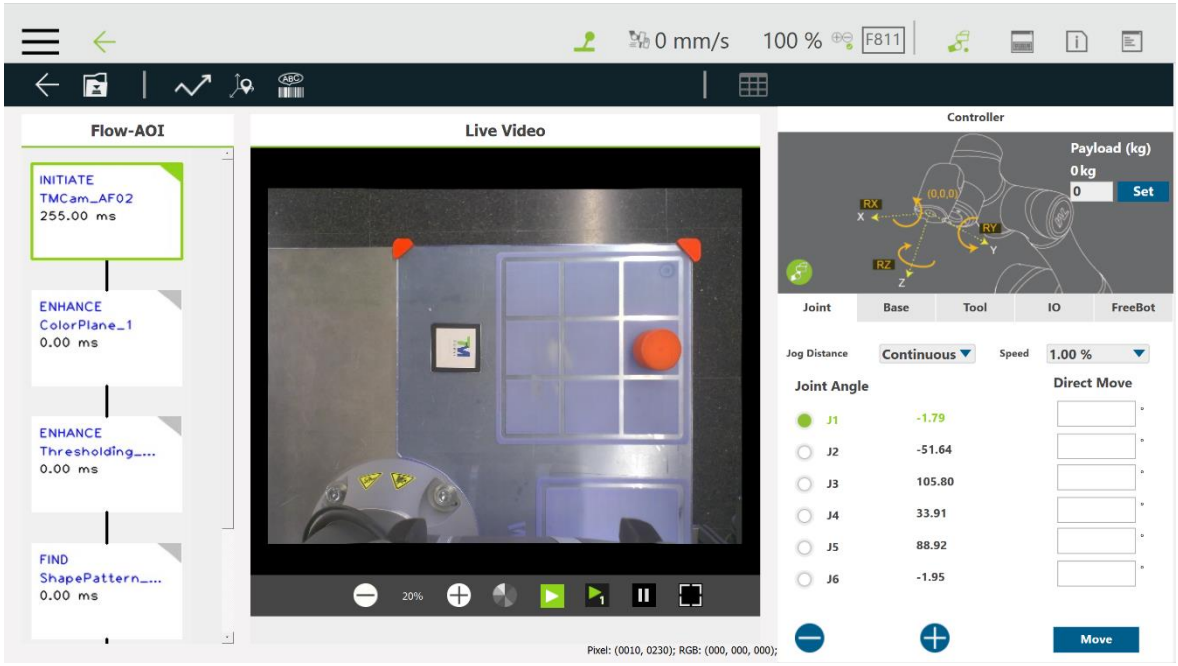


Figura 39: Initiate TMCam

Es pot veure en la *Figura 39* com si la càmera no es troba justament a sobre del objecte es podrà veure la profunditat del objecte, també de color vermell.

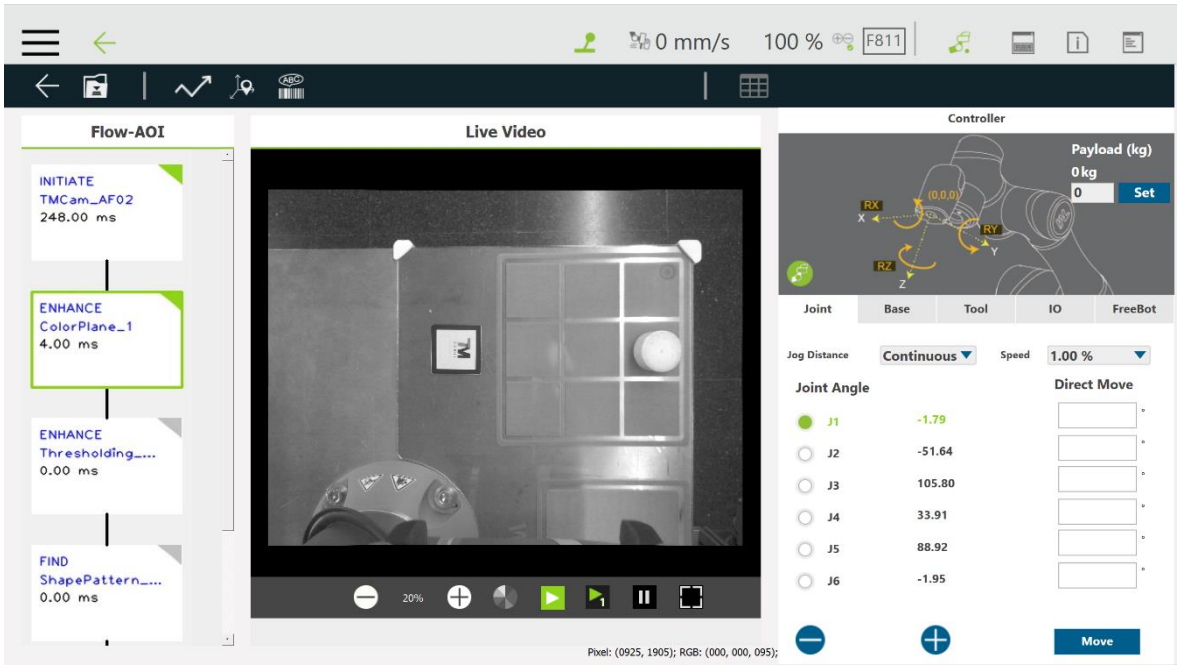


Figura 40: Enhance Color Plane

A continuació apliquem el filtre de Vermell, tal i com es veu en la *Figura 40.* Això converteix la imatge a blanc i negre tenint en compte el nivell de vermell de cada píxel, sent blanc un nivell elevat de vermell. Per a fer l'efecte més visible es pot incrementar el contrast de forma significativa, fent així que només hi hagi blanc o negre, *Figura 41.*

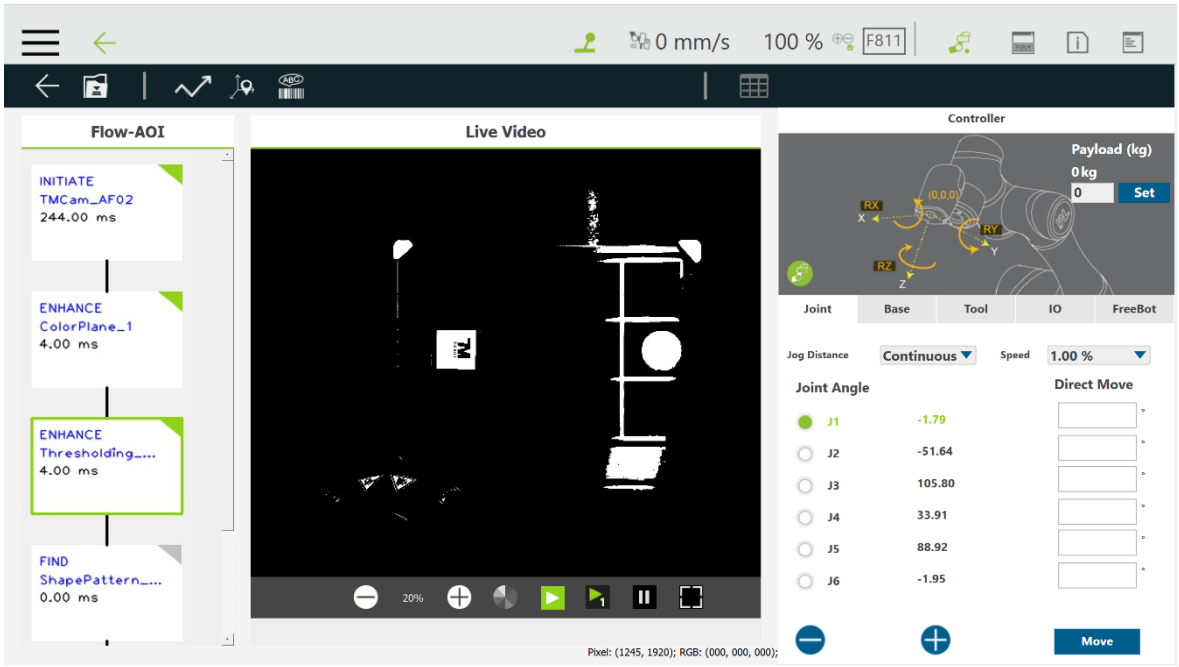


Figura 41: Enhance Thresholding

Finalment s'aplica una reconeixença de formes circulars per determinar les coordenades. En la *Figura 42* però, es pot veure com lla peça, aplicant aquest filtres ha quedat una mica deformat, ja que el gruix de la peça ha afectar a l'hora de definir el contorn de la peça i per tant ha donat unes coordenades una mica esbiaixades.

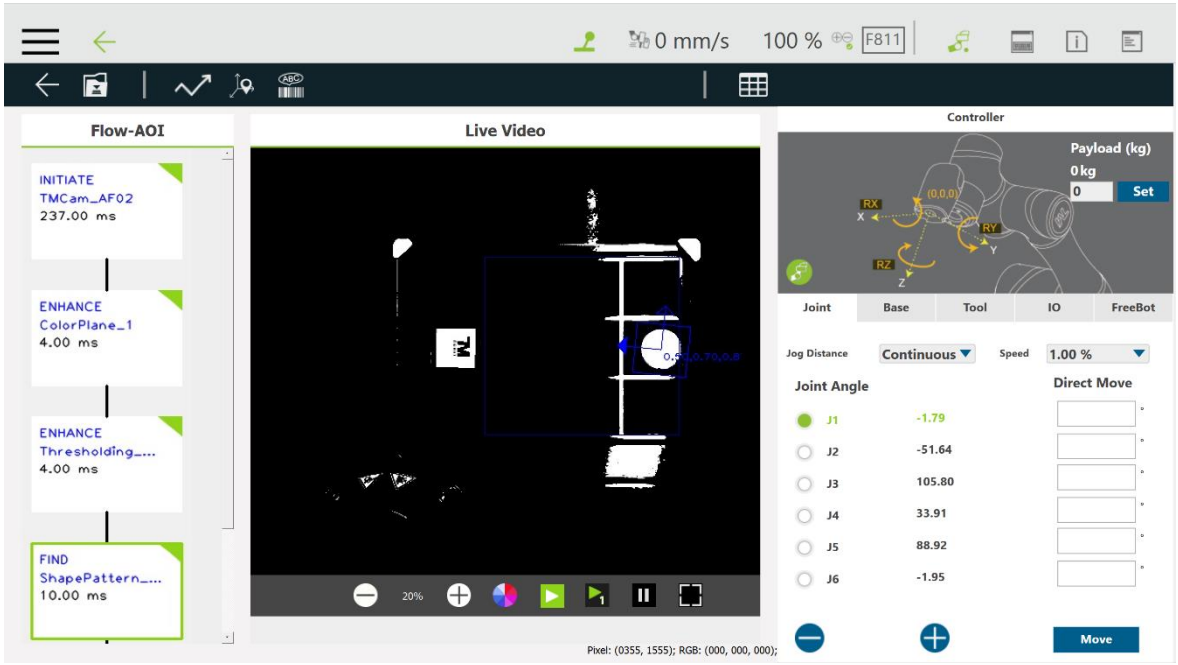


Figura 42: Find Shape Pattern

Degut a aquest problema, a l'hora d'aplicar aquest model de visió artificial en aquest projecte el robot agafava sempre la peça per un costat i no pel centre, motiu pel qual s'ha determinat que no es podia aplicar en aquest cas.

Degut a que no es pot utilitzar la forma de la peça com a base per trobar les coordenades d'aquesta, s'ha recorregut a un sistema de coordenades extern proporcionat per un *landmark*, una etiqueta dissenyada especialment per a ser reconeguda mitjançant visió artificial. A continuació es pot veure com es reconeix el *landmark* en el suport del TurtleBot en la *Figura 43*.

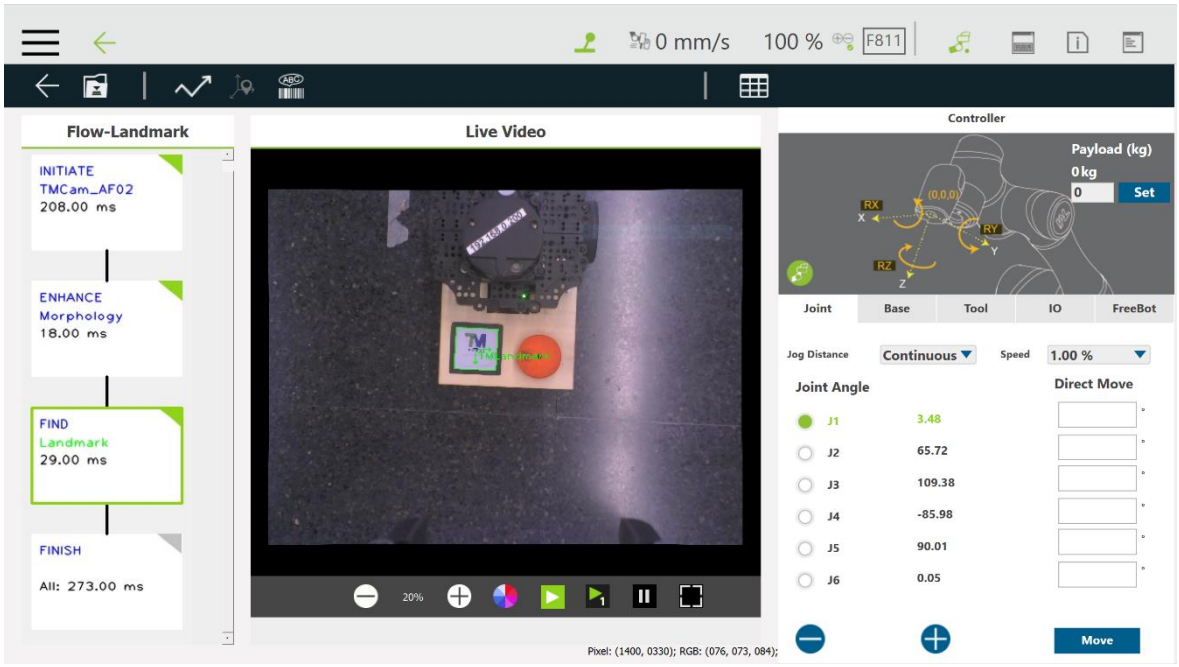


Figura 43: Detecció landmark

Aquesta proporciona un sistema de coordenades propi, el qual permet indicar un punt a l'espai en referència al *landmark.* Adherint el *landmark* al suport del TurtleBot tal i com es veu a la *Figura 43*, s'ha obtingut unes coordenades de la peça, així que s'ha definit un punt anomenat *PickV* amb les coordenades que es poden veure en la *Figura 44.*

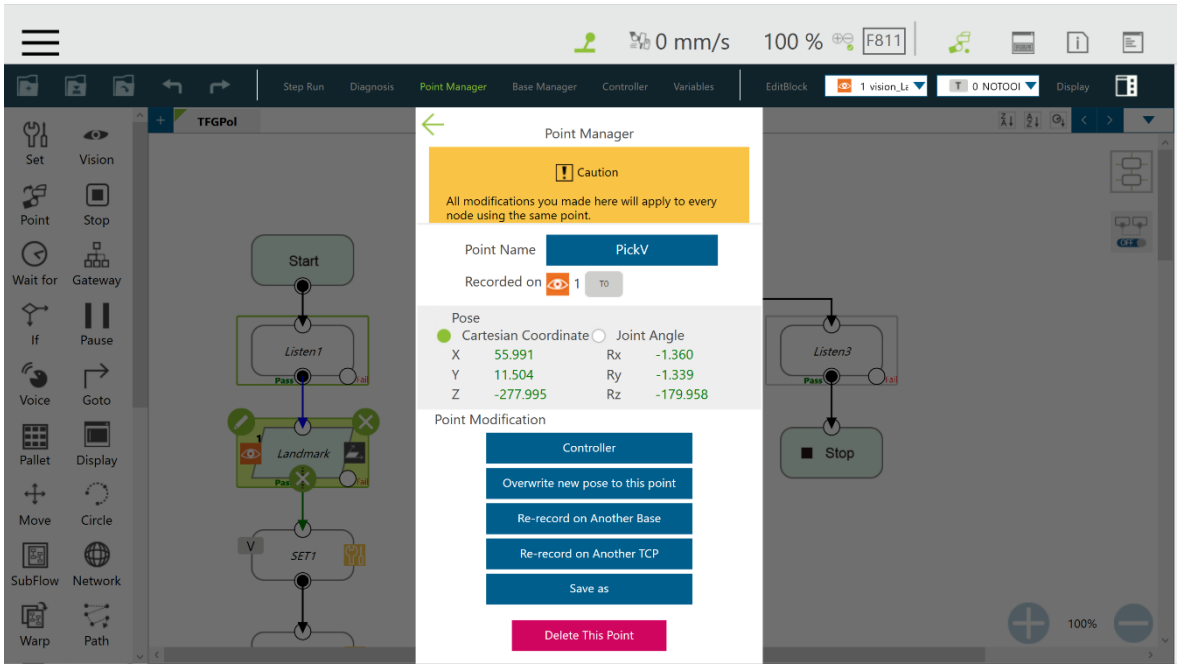


Figura 44: Coordenades PickV

4.4.5. Enviament de coordenades

Un cop obtingudes les coordenades de la peça mitjançant visió artificial, cal enviar-les amb el format adequat al servidor per, mitjançant MoveIt, anar cap a aquestes. En la *Figura 45* es pot veure els nodes emprats per a fer això possible.

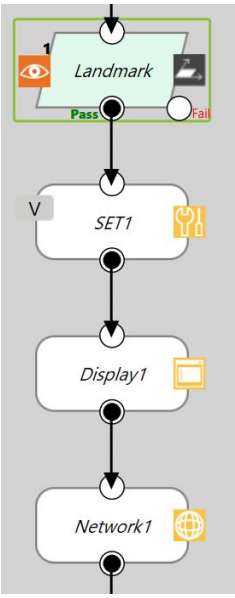


Figura 45: Enviament de coordenades

MoveIt interactua amb el robot mitjançant un sistema de coordenades basat en la base del robot, mentre que les coordenades de la peça estan basades en el *landmark*. Per a que MoveIt pugui interpretar les coordenades cal que estiguin basades en la base del robot, no en el *landmark*. Per a fer el canvi de referència s'utilitza un node en el software TMFlow anomenat *SET*, el qual permet canviar valors de les sortides digitals, fer canvis a variables i d'altres.

Dins el node hi ha la següent expressió, la qual defineix el punt *Pick* com a la transformació del punt *PickV* de la base *vision_Landmark* a la base *RobotBase*.

```
Point["Pick"].Value = changeref(Point["PickV"].Value, Base["vision_Landmark"].Value,
Base["RobotBase"].Value)
```

Un cop obtingudes les coordenades en la base correcte cal enviar-les al servidor mitjançant el node *network*. En ell es defineix una IP i un port on enviar la informació que es vulgui, en aquest cas el servidor, al qual s'envia el valor del punt *Pick* en format text, *Figura 46*.

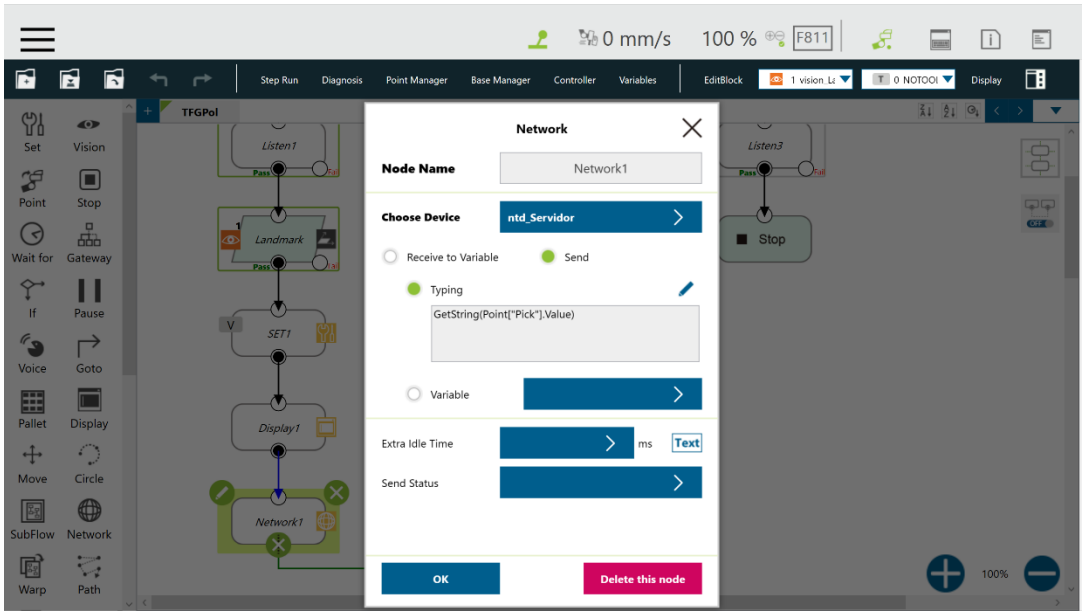


Figura 46: Node network

4.4.6. Pinça RG2

El braç robòtic TM5 està equipat amb una pinça OnRobot model RG2, *Figura 47*. Aquesta consta de una capacitat d'obertura de 100mm i un sensor de força per a aplicar un nivell de força constant.

Figura 47: Pinça RG2

Per a poder controlar la pinça remotament, degut a que el driver per a aquest no està disponible per a ROS[5], s'ha hagut de utilitzar un altre via. La *compute box*, *Figura 48*, és la caixa de control de la pinça RG2 i compte amb entrades i sortides digitals, així com sortides analògiques.

[5]RG2. (2022). ROS Components. Recuperat l'11 de febrer de 2022, de https://www.roscomponents.com/en/grippers/293-rg2.html

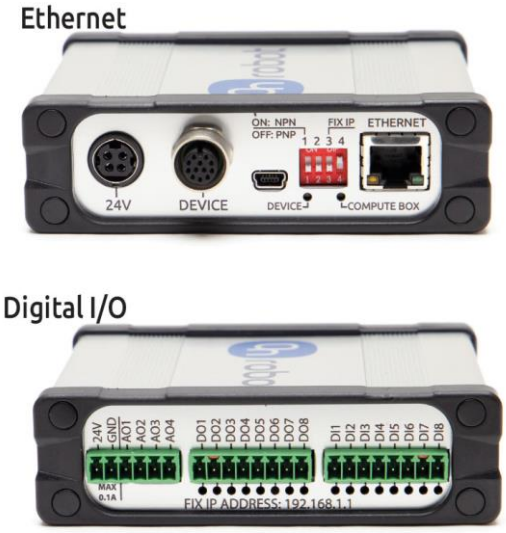


Figura 48: Compute Box

En aquesta aplicació ens interessen les entrades digitals, ja que des del client web que té la caixa es pot configurar una acció en funció de les entrades. En aquest treball s'ha configurat que la pinça s'obri 80 mm quan la entrada digital *DI1* sigui 0 i que es tanqui quan sigui 1. En la *Figura 49* es pot veure un exemple.

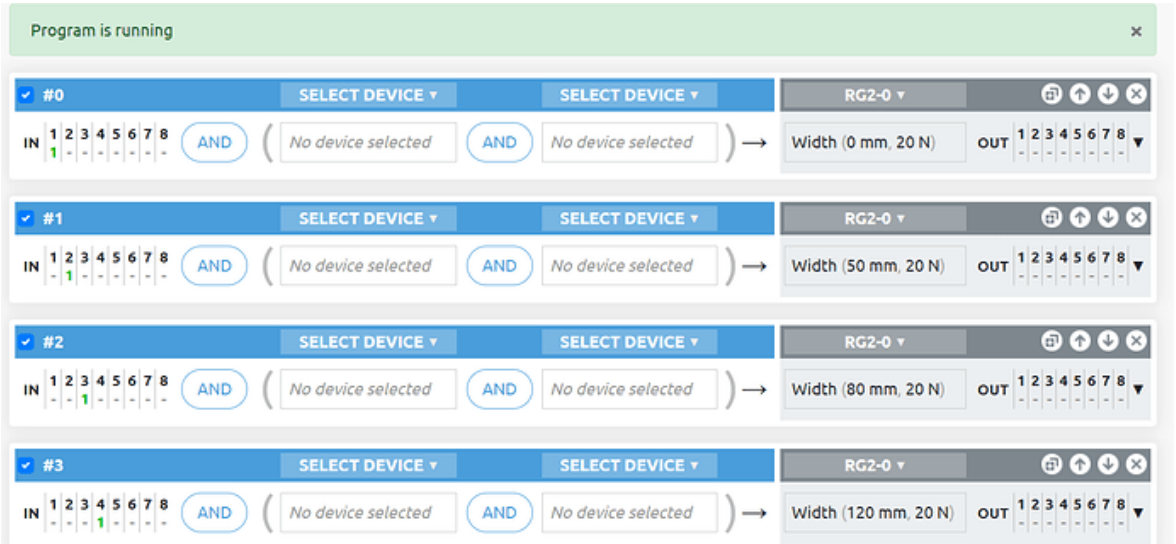
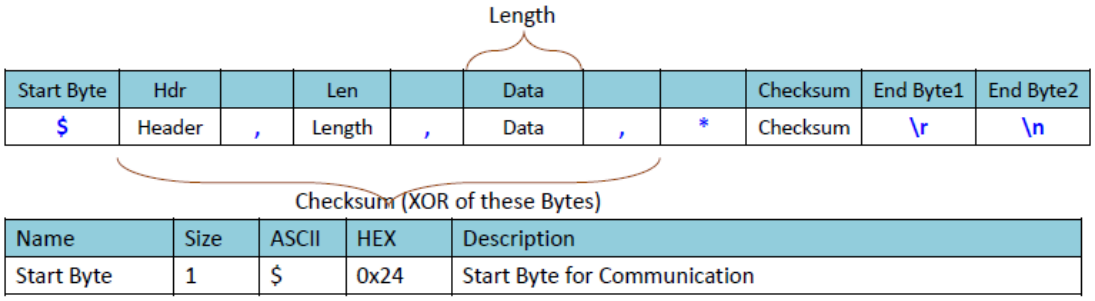


Figura 49: Exemple configuració compute box

Per a controlar la pinça doncs, cal controlar una sortida digital cablejada a la entrada digital *DI1*. La controladora del TM5 compta amb sortides digitals, així que s'ha connectat les masses d'ambdues controladores, així com la entrada DI1 de la *compute box* i la sortida digital *DO0* de la controladora del braç. A conseqüència, la pinça respon a la sortida digital *DO0* de la controladora del TM5, sent un 1 lògic tancada i un 0 lògic oberta.

Per a controlar la sortida digital s'ha recorregut al *ethernet slave*, un mètode de comunicació on es pot controlar certes característiques del robot mitjançant comandaments. Les instruccions que s'envien han de seguir el format que dicta el protocol, *Figura 50*.



Name	Size	ASCII	HEX	Description
Start Byte	1	$	0x24	Start Byte for Communication

Figura 50: Protocol de comunicació

Per a controlar les sortides digitals s'ha d'utilitzar una instrucció TMSVR. Segons el protocol, es comença indicant el tipus d'instrucció que és, seguit per a la longitud de aquesta. Seguidament va les dades i finalment el checksum. Les dades són Ctrl_DO0=1 i Ctrl_DO0=0 en aquest cas. El checksum és una operació matemàtica que s'utilitza en protocols de comunicació per a verificar si la informació rebuda és vàlida. Per a calcular aquest s'ha utilitzat la següent funció amb Python.

```
def getCheckSum(sentence):
    calc_cksum = 0
    for s in sentence:
        calc_cksum ^= ord(s)
    return str(hex(calc_cksum)).lstrip("0").lstrip("x")
```

Ajuntant aquesta informació finalment obtenim dues instruccions, una per obrir i l'altre per tancar la pinça.

```
'$TMSVR,15,T3,2,Ctrl_DO0=1,*72\r\n'
'$TMSVR,15,T3,2,Ctrl_DO0=0,*73\r\n'
```

Aquestes s'han de enviar mitjançant sockets, en la IP del TM5 utilitzant el port del *ethernet slave*, el 5891.

4.4.7. Sortida node listen

Degut a que cal utilitzar el node de visió artificial, cal sortir del node *listen* per a realitzar el projecte. Per a fer-ho, l'única manera de fer-ho és executant la ordre *ScriptExit().* Aquesta s'executa seguint el mateix format que per a la pinça, *Figura 50,* tot i que aquest cop utilitzant una instrucció de tipus TMSCT. Seguint el mateix protocol, la instrucció a enviar és aquesta.

```
'$TMSCT,14,0,ScriptExit(),*66\r\n'
```

Enviant aquesta instrucció quan el robot es troba en el node *listen* el flux del programa surt del node i passa al següent, tal i com s'indica a la documentació. Quan està funcionant el *tm_driver* però, la comunicació és exclusiva entre el TM5 i el procés que ha iniciat la comunicació, per tant al enviar externament aquest comandament el robot no responia, ja que tenia una altre comunicació en procés. Degut a això, la manera que s'ha trobat de resoldre aquest problema és interrompent aquesta comunicació. Per a fer-ho, es mata al driver mitjançant la següent instrucció, forçant a aquest a enviar el ScriptExit() ell mateix.

```
command="rosnode kill /tm_driver"
kill_driver=subprocess.Popen(shlex.split(command))
```

4.4.8. Programació final

Finalment, el programa de TMFlow ha quedat del i com es veu en la *Figura 51*. Compte amb tres nodes *Listen* intercalats amb el enviament de coordenades. El primer node listen, *Listen1*, posa al robot en posició d'espera fins que arriba el TurtleBot, moment en que troba les coordenades i les envia per, un cop entrat al node L*isten2*, agafar la peça i dur-la a mecanitzar. Un cop mecanitzada tornar a reconèixer les coordenades a on deixar la peça. Finalment entre en el node *Listen3*, deixa la peça i s'acaba el programa.

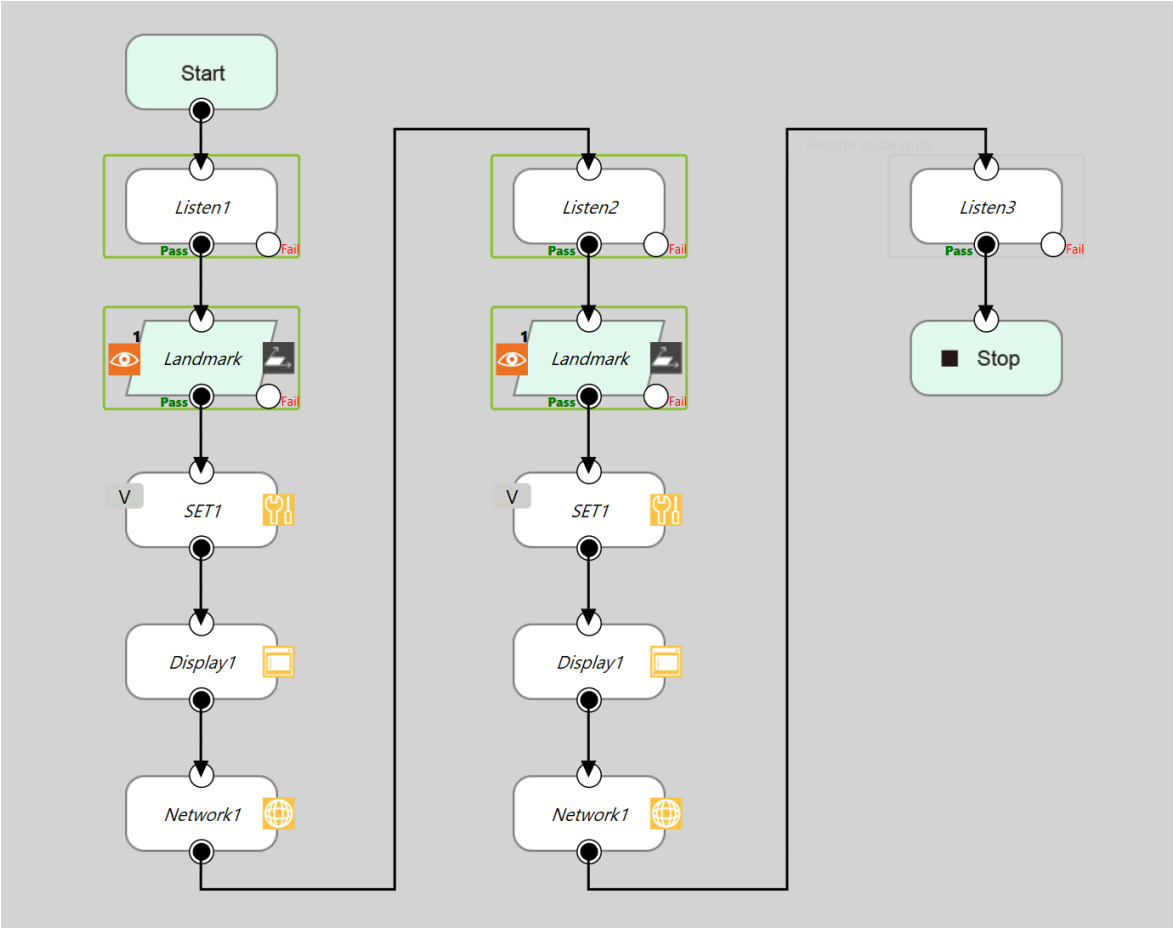


Figura 51: TMFlow

4.5. Programació PLC

Per a realitzar el mecanitzat de la peça s'ha hagut de controlar l'estació de mecanitzat, *Figura 52*, mitjançant el PLC, *Figura 53*.

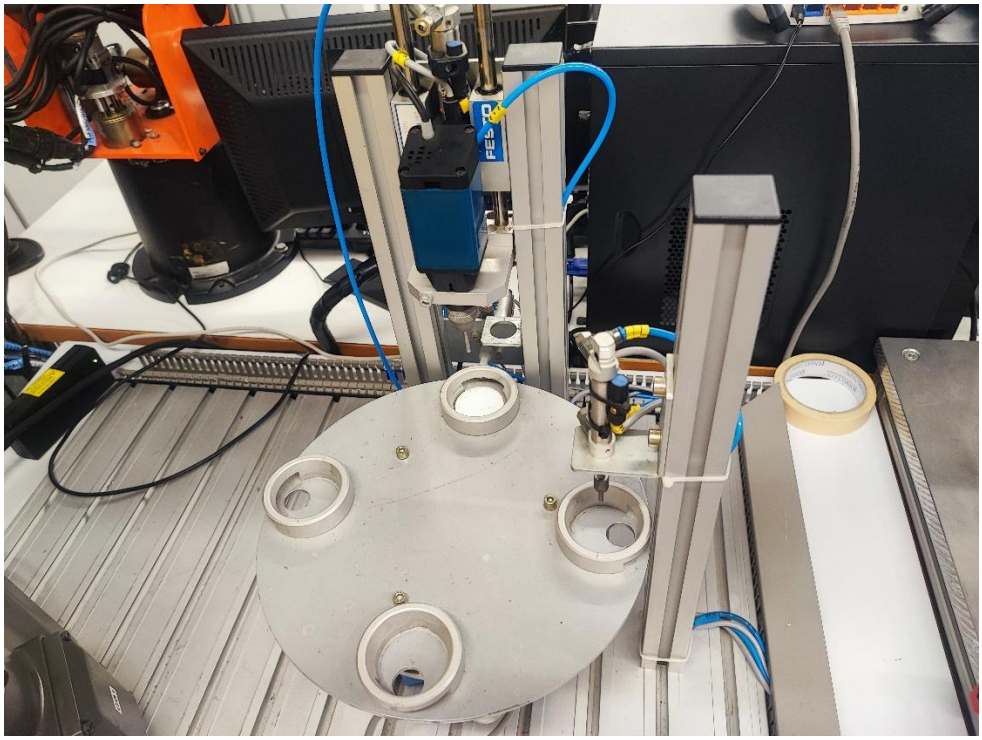


Figura 52: Estació de mecanitzat

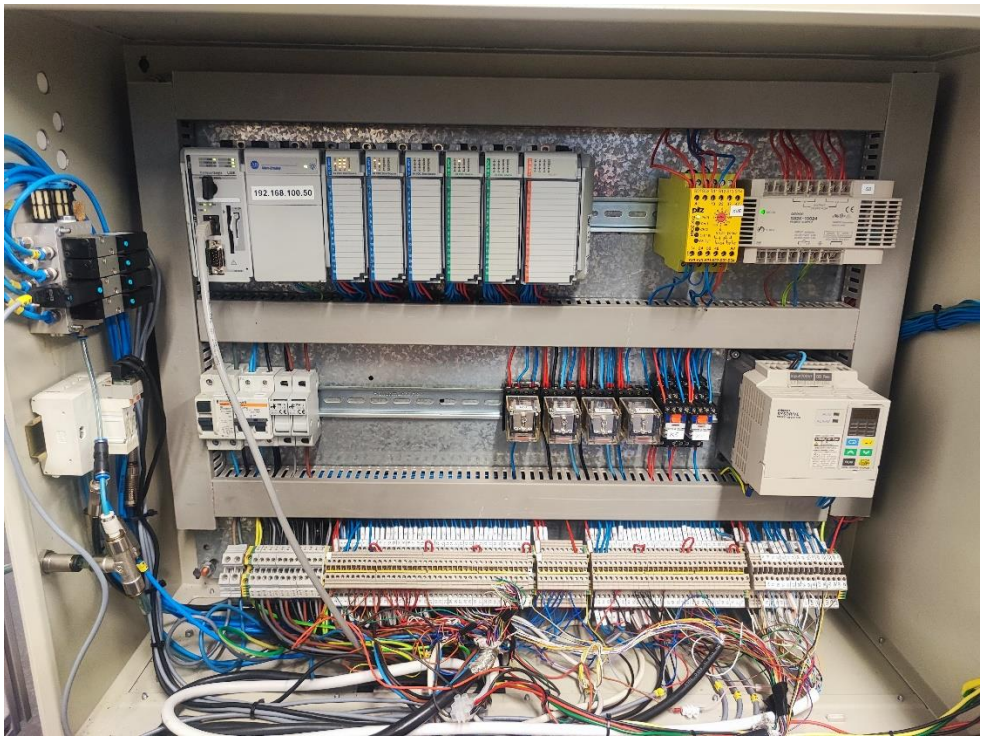


Figura 53: Caixa PLC

Per a controlar el PLC s'ha emprat la llibreria de Python *pylogix*, emprada per a llegir i escriure *tags* del PLC. Els *tags* a llegir i/o escriure són els següents:

- *Program:MainProgram.llum_verd_marxa*
- *Program:MainProgram.llum_vermell_aturada*
- *Program:MainProgram.motor_plat_giratori*
- *Program:MainProgram.EV_trepant_baixar*
- *Program:MainProgram.EV_trepant_pujar*
- *Program:MainProgram.gir_horari_trepant*
- *Program:MainProgram.EV_verificador*
- *Program:MainProgram.EV_subjectador*
- *Program:MainProgram.sensor_inductiu_plat*

Controlant aquests *tags* ens permet controlar tota la estació de fresat. Seguidament es pot veure la seqüencia d'ordres que s'executen. Cal notar que els llums s'utilitzen al inici i al final del sistema, per tant no formen part del mecanitzat de la peça.

1. *motor_plat_giratori* ON
2. quan *sensor_inductiu_plat* detecti 2 flancs de pujada → *motor_plat_giratori* OFF
3. *EV_subjectador* ON
4. *gir_horari_trepant* ON
5. *EV_trepant_baixar* ON
6. *EV_trepant_baixar* OFF
7. *EV_trepant_pujar* ON
8. *EV_trepant_pujar* OFF
9. *gir_horari_trepant* OFF
10. *EV_subjectador* OFF
11. *motor_plat_giratori* ON
12. quan *sensor_inductiu_plat* detecti 1 flanc de pujada → *motor_plat_giratori* OFF
13. *EV_verificador* ON
14. *EV_verificador* OFF
15. *motor_plat_giratori* ON
16. quan *sensor_inductiu_plat* detecti 1 flanc de pujada → *motor_plat_giratori* OFF

4.6. Posada en marxa

Al final del treball s'ha assolit el funcionament del projecte completament. Partint de les posicions actuals, el servidor inicia la navegació del TurtleBot i li indica que es mogui cap al TM5. Mentrestant, el servidor engega MoveIt i prepara l'entorn perquè el braç robòtic no es pugui xocar, seguidament el posiciona a l'espera que el servidor revi un senyal del TurtleBot indicant la seva arribada.

Una vegada el senyal ha sigut rebuda, el servidor mata al tm_driver, sortint així del node listen, iniciant la tasca de visió artificial. Quan s'han determinat les coordenades i s'ha fet el canvi de base, s'envien les coordenades i el braç robòtic torna a entrar al node listen. Des del servidor, es reinicia el driver i des de MoveIt s'envia l'ordre d'anar a les coordenades i es tanca la pinça. Un cop agafada la peça, es duu cap a l'estació de fresat, es realitza un moviment lineal per baixar la peça, s'obre la pinça i es realitza un moviment lineal vertical.

Seguidament, es realitza el mecanitzat de la peça, descrit en l'apartat anterior. Una vegada mecanitzada la peça, es torna a realitzar el mateix moviment lineal per baixar, es tanca la pinça i es puja linealment, seguidament es torna a la posició inicial. Una vegada es troba en la posició inicial, es torna a matar el tm_driver per a tornar a entrar en el node de visió artificial i detectar les coordenades del suport de la peça. Altre cop es fa el canvi de base i s'envien les coordenades, aquest cop però, per a deixar la peça en el suport. Des del servidor es reben les coordenades, es torna a iniciar el tm_driver i es va a les coordenades rebudes, sobre la pinça i es torna a la posició inicial.

Una vegada la peça es troba en el suport del TurtleBot i el braç robòtic es troba en posició de repós, el servidor indica al TurtleBot que vagi fins a la zona de descàrrega, i allà s'acaba el sistema.

En la *Figura 54* es pot veure un esquema de l'arxiu de software que s'ha emprat per a fer aquest projecte possible. En els annexos es pot trobar el codi sencer.

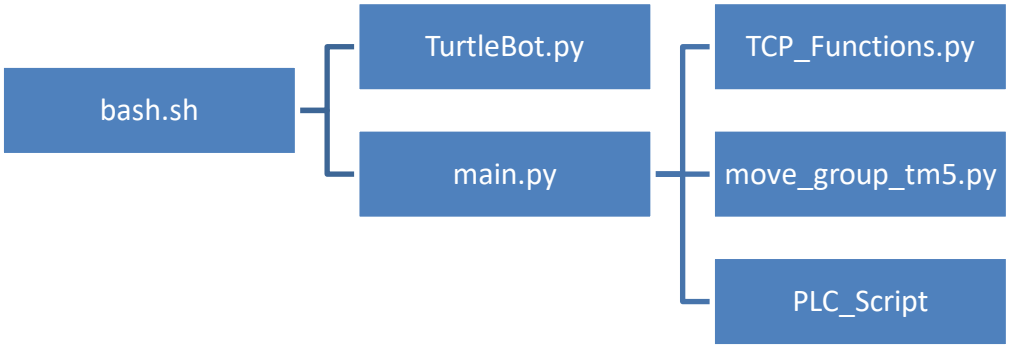


Figura 54: Esquema software

5. Anàlisi de viabilitat mediambiental

En termes generals, s'ha considerat que aquest projecte no te un impacte significatiu en el medi ambient. L'estudi de viabilitat mediambiental s'ha realitzat tenint en compte la fase d'explotació i el final de la vida útil.

En la fase d'explotació El braç robòtic TM5 no consumeix més que electricitat. El TurtleBot, però, va alimentat per bateries recarregables, la qual cosa disminueix el seu impacte, però cal tenir en consideració el canvi de bateries al final de la vida útil d'aquestes. Cal disposar-les en el punt de recollida municipal que li pertoqui per al seu posterior reciclatge.

En el final de vida útil, tant el TurtleBot com el TM5 s'hauran de disposar en el punt de recollida municipal habilitat per al seu reciclatge, tenint en compte que la bateria del TurtleBot va per separat.

6. Anàlisi de perspectiva de gènere

Aquest projecte s'ha realitzat dins el marc general proporcionat per l'Agència per a la Qualitat del sistema Universitari de Catalunya (AQU) per a incloure la perspectiva de gènere en la docència universitària.

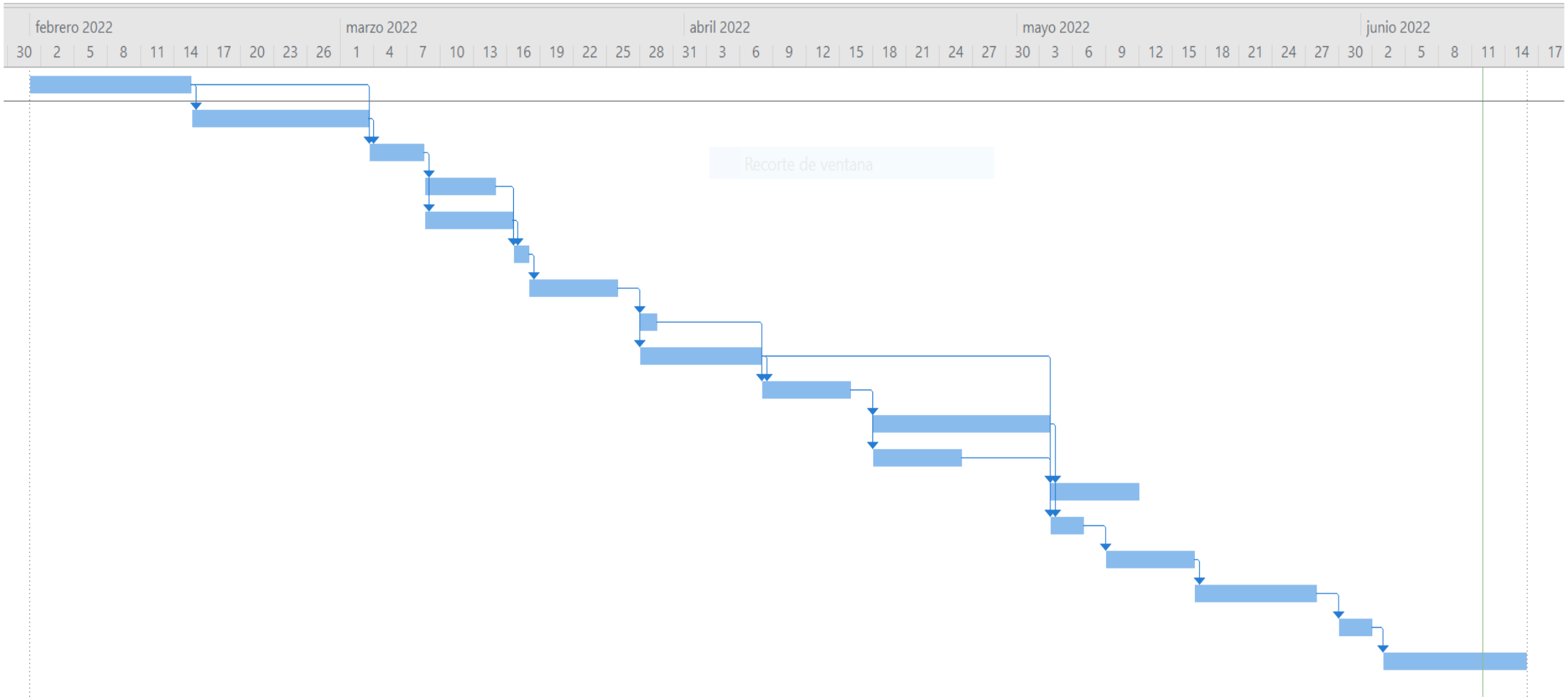
Donada la naturalesa del projecte, es declara neutral en quant a perspectiva de gènere, degut a la dificultat simètrica en realitzar la programació tant per homes com dones.

Tot i que pròpiament la dificultat de programació dels robots és cega al gènere, és habitual trobar un desequilibri en la proporció d'homes i dones que ho realitzen. Això però no és degut a que existeixin dificultats de gènere en la tasca en desenvolupar, sinó en un sexisme inherent al sistema actual, el qual priva d'oportunitats en base al gènere.

En aquest projecte es vol assegurar que no existeixin efectes diferenciats per a homes i dones a conseqüència d'emprar aquest treball.

7. Planificació i tasques del projecte

ACTIVITAT	HORES	INICI	FINAL
Recerca informació preliminar	40	dt. 1/2/22	dt. 15/2/22
Redacció avantprojecte	40	dt. 15/2/22	dj. 3/3/22
Realització esquema general	10	dj. 3/3/22	dt. 8/3/22
Recerca Ros	40	dt 8/3/22	dl. 14/3/22
Configuració TurtleBot	30	dt 8/3/22	dc. 16/3/22
Mapeig TurtleBot	5	dc. 16/3/22	dj. 17/3/22
Navegació TurtleBot	10	dv. 18/3/22	dv. 25/3/22
Suport TurtleBot	10	dl. 28/3/22	dt. 29/3/22
Recerca TM5-900	40	dl. 28/3/22	dj. 7/4/22
Redacció memòria intermitja	30	dv. 8/4/22	dv. 15/4/22
Configuració TMDriver i MoveIt	60	dl. 18/4/22	dt. 3/5/22
Visió artificial	25	dl. 18/4/22	dl. 25/4/22
Enviament coordenades TM5	25	dc. 4/5/22	dc. 11/5/22
Pinça TM5	15	dc. 4/5/22	dv. 6/5/22
Programació PLC	30	dl. 9/5/22	dl. 16/5/22
Programació intercanvi senyals	50	dt. 17/5/22	dv. 27/5/22
Posada en marxa	10	dl. 30/5/22	dc. 1/6/22
Redacció memòria final	40	dv. 3/6/22	dc. 15/6/22


febrero 2022
marzo 2022
abril 2022
mayo 2022
junio 2022
30 2 5 8 11 14 17 20 23 26 1 4 7 10 13 16 19 22 25 28 31 3 6 9 12 15 18 21 24 27 30 3 6 9 12 15 18 21 24 27 30 2 5 8 11 14 17

Descripció de les tasques a realitzar:

Recerca informació preliminar

- Estudi robòtica
- Estudi Història de la Robòtica
- Estudi TurtleBot
- Estudi TM5-900

Redacció avantprojecte

- Definir objectius
- Documentar antecedents
- Viabilitat tècnica, econòmica i mediambiental
- Anàlisi de perspectiva de gènere
- Edició document

Realització esquema general

- Plantejament interacció entre els dos robots i el PLC

Recerca Ros

- Estudi estructura ROS
- Estudi diferents nodes
- Estudi profund navegació

Configuració TurtleBot

- Instal·lació sistema operatiu
- Instal·lació ROS

Mapeig TurtleBot

- Preparació del LAB 4
- Mapeig LAB 4

Navegació TurtleBot

- Navegació mitjançant RViz
- Navegació mitjançant coordenades

Suport TurtleBot

- Disseny amb *Fusion 360* del suport del TurtleBot
- Tall amb làser del suport

Recerca TM5-900

- Estudi TMFlow
- Estudi Visió Artificial

Redacció memòria intermèdia

- Incorporació del progrés a l'avantprojecte
- Millora de l'avantprojecte

Configuració TMDriver i MoveIt

- Instal·lació TMDriver per a ROS
- Proves de comunicació
- Instal·lació MoveIt
- Proves de diversos tipus de moviment
- Programació dels moviments
- Implementació verificació assoliment objectiu
- Implementació limitacions físiques

Visió Artificial

- Proves de diversos tipus de visió artificial
- Aplicació de visió artificial mitjançant TMFlow

Enviament coordenades TM5

- Disseny comunicació amb el servidor
- Transformació de les coordenades
- Canvi de base

Pinça TM5

- Proves amb Modbus
- Compute Box
- Control sortides digitals mitjançant Ethernet slave

Programació PLC

- Configuració de Xarxa
- Identificació dels *tags* a llegir i escriure
- Elaboració codi control PLC

Programació intercanvi senyals

- Disseny i incorporació de comunicació entre robots per coordinar moviments

Posada en marxa

- Verificació sota diverses condicions del funcionament individual d'ambdos robots
- Verificació interacció entre robots
- Verificació funcionament conjunt

Redacció memòria final

- Documentació realització del projecte
- Millora i revisió avantprojecte

8. Conclusions

La realització d'aquest treball implica dominar àmpliament diverses àrees d'electrònica, principalment la robòtica i programació en el llenguatge de programació Python. Per això, s'ha considerat un treball ideal com a culminació del final de carrera.

Tot i els reptes que ha presentat aquest treball, s'ha encarat amb ganes, com a l'últim gran repte de la carrera. Ha sigut l'oportunitat perfecte per a demostrar tot l'aprés en la carrera i abastar àrees que prèviament s'han passat per sobre.

Al començar aquest treball es va entrar com a un estudiant d'enginyeria i al sortir sento que ja em puc dir enginyer. Un enginyer ha de resoldre tots els problemes que li vinguin per davant, en sàpiga molt o poc del tema en qüestió, i sento que he fet exactament això en aquest treball. He programat molt amb Python, he tractat connexions socket, cinemàtica de braços robòtics i Linux. Algunes d'elles ja les sabia fer mínimament, però la majoria no.

Tot això no s'ha enfrontat com una dificultat, sinó com un repte on superar-se a si mateix, i és per això que em sento molt més enginyer que fa un any.

9. Referències

[i] Moran, M. E. (2006). The da Vinci Robot. *Journal of Endourology*, *20*(12), p. 986–990. https://doi.org/10.1089/end.2006.20.986

[ii] Elcacho, J. (Abril de 2013). Més d'un segle de cursa robòtica. *Theknos*, 172, 20–25.

[iii] Spot® - The Agile Mobile Robot. (2022). Boston Dynamics. Recuperat l'11 de febrer de 2022, de https://www.bostondynamics.com/products/spot

[iv] Artificial Intelligence & Autopilot. (2022). Tesla. Recuperat l'11 de febrer de 2022, de https://www.tesla.com/AI

[v] Wikipedia contributors. (10 de juny de 2022). Current loop. Wikipedia. Recuperat l'11 de febrer de 2022, de https://en.wikipedia.org/wiki/Current_loop

[vi] Kosow, I. L. (1993). Máquinas eléctricas y transformadores. Prentice Hall. P. 429-.

[vii] I, Cobot: Future collaboration of man and machine. (2015). The Manufacturer, 5. http://www.themanufacturer.com/articles/i-cobot-future-collaboration-of-man-and-machine/

[viii] International Federation of Robotics. (2022). IFR International Federation of Robotics. Recuperat l'11 de febrer de 2022, de https://ifr.org

[ix] Ronzhin, A., Rigoll, G., & Meshcheryakov, R. (2016). Interactive Collaborative Robotics: First International Conference, Icr 2016, Budapest, Hungary, August 24–26, 2016, Proceedings. Springer. P. 254

[x] ROS: Home. (2022). ROS - Robot Operating System. Recuperat l'11 de febrer de 2022, de https://www.ros.org/