Treball de Final de Grau

Grau Universitari en Enginyeria de Tecnologies Industrials (GETI)

Disseny d'un controlador de força i estudi d'un robot UR3

MEMÒRIA

Convocatòria: Setembre 2021

Autor: Lluís Bonet Ortuño.

Directora: Yolanda Bolea .

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resum:

Aquest TFG consisteix en un estudi d'un braç robòtic UR3 i el disseny d'un controlador de força per aquest robot. Té dues parts diferenciades. L'estudi del robot s'ha enfocat de forma pràctica: S'ha realitzat un curs sobre el llenguatge de programació del robot, *Polyscope*. Amb això s'ha aconseguit adquirir coneixement sobre el funcionament del robot i aprendre a fer servir una sèrie d'eines amb les que poder realitzar la segona part del TFG, el controlador de força.

S'ha implementat un controlador PID de força per a una posició específica del robot. Per a poder realitzar aquests ha estat necessari realitzar un experiment amb el robot, en el que s'ha estudiat la planta del sistema. Posteriorment, es comparen amb altres controladors i amb el controlador de força que incorpora el software de *Polyscope*, fent servir la comanda *Force*.

L'objectiu transversal d'aquest TFG és purament didàctic, un primer contacte amb el món de la robòtica.

ÍNDEX

INDEX DE FIGURES

1. PREFACI

1.1. Motivació del TFG:

La robòtica, juntament amb l'IA (Intel·ligència Artificial), són tecnologies que tenen el potencial de canviar la nostra forma de viure d'una forma molt disruptiva. És un tema molt punter i que em genera molt d'interpes personal, per a aquesta raó vaig voler aprofitar l'oportunitat de realitzar un projecte amb un robot.

La ida d'incorporar al TFG un curs de robòtica va ser idea de la meva tutora. És una forma interessant d'enfocar el treball, ja que permet aprendre de forma molt pràctica treballant amb el robot al laboratori, tant durant el procés de realització del curs com en el disseny del controlador.

1.2. Requirements previs:

Exceputuany l'apartat ed comunicació, el curs que s'ha implementat està pensat per a persones que no tinguin coneixements previs en camps de programació i robòtica. Pertant, els únics requeriments per a poder cursar el curs són materials: Un robot UR3 e-Series, una sèrie de perifèrics del robot, el simulador del software de control del robot (*UrSim*) els i el material didàctic redactat. És necessari per altra banda tenir coneixements bàsics de *Python* per a poder entendre el codi que es fa servir a l'apartat de comunicació del curs.

Per a la realització del controlador s'ha fet servir *Python* per a realitzar els codi per a poder recollir les dades durant els experiments i programar el controlador. També s'ha fet servir *Excel* per a realitzar alguns dels càlculs que han estat necessaris per a poder dissenyar el controlador.

1.3. Abast del TFG

En aquest projecte es preté realitzar un breu curs de robòtica i un controlador de força del robot Ur3 e-series. Tant el curs com el controlador tenen una intenció didàctica.

L'enfocament que s'ha près per a realitzar aquestes tasques ha consistit en crear primer un curs bàsic de *Polyscope*, una interfície gràfica instal·lada a la tauleta del control del robot (*Teach Pendant*) i al simulador del robot *UrSim*. En aquest curs també es parla breiument sobre els sistemes de comunicació que té disponibles el robot, ja que serà necessari establir una comunicació entre l'ordinador i el robot per tal de poder implementar el controlador.

Posteriorment s'han aplicat els coneixements adquirits durant la realització del curs per a dissenyar un controlador de força PID. Per a fer això ha estat necessari realitzar una sèrie d'experiments amb

el robot. L'objectiu d'aqueststst experiments ha estat el d'obtenir la funció de trasnferència de la planta del sistema. Posteriorment, amb aquest valor de la planta, s'ha cdissenyat un controlador PD de temps continu, imposat una sèrie d'especificacions per al sistema.

1.4. Estructura del TFG

El treball s'inicia amb una introducció, en la que s'exposa l'estat de l'art de la robòtica, informació sobre el robot UR 3 e-Series i sobre l'oferta de cursos de formació per a treballar amb aquest robot.

Posteriorment es presenta el curs de robòtica que s'ha realitzat de forma resumida: S'explica el funcionament bàsic de la GUI (graphical user interface) del simulador, quines funcionalitats de seguretat té el robot i una breu explicació de les comandes que es poden fer servir amb *Polyscope*. El cust també comenta, encara que de forma més superficial, quins són els protocols de comunicació que té el robot, així com el funcionament de la llibreria de *Python* que s'ha fet servir per a controlar el robot.

Finalment, es troba el disseny d'un controlador de força. En aquest apartat s'explica el procediment que s'ha realitzat per a obtenir el controlador, així com la seva justificació teòrica i els càlculs realitzats. Per a l'implementació del controlador s'ha fet servir la llibreria *ur_rtde* de *Python*.

2. INTRODUCCIÓ

2.1. Estat de l'art

2.1.1. La robòtica al món actual

2.1.1.1. Introducció

La forma amb la que treballen i produeixen les persones està estretament lligada a la millores tècniques i optimització dels processos de producció. L'invenció de noves eines que faciliten el treball, noves formes d'organitzar-se o l'aparició de noves estratègies de financiació han canviat l'entorn de treball al llarg del temps. Sectors que anteriorment requerien gran part de la població per a produir actualment requereixen un nombre de treballadors mínim que produeixen molt més.

2.1.1.2. Robòtica: Robots d'automatització i cobots.

La robòtica és un dels principals motors de millora de la productivitat en l'actualitat. Es tracta d'una indústria que es troba en creixement constant des de fa més d'una dècada. Tal i com es mostra a la figura 1, el nombre de robots industrials instal·lats està creixent de forma continuada i les projeccions indiquen que això seguirà siguent així [1]. Aquest creixement està impulsat per l'augment de la productivitat que genera l'incporporació de robots a una empresa. Tot i això és destacable que l'utilització de robots no sigui transversal als negocis de tota mida,. La majoria de robots es troben acumulats en grans empreses que tenen volums molt elevats de producció. Un exemple clar d'això són empreses de l'indústria de l'automòbil o de l'indutsria farmacèutica, que es troben altament robotitzades [2][3]. Aquestes tenen esl recursos financers suficients com per a poder assolir l'elevat cost de la maquinaria, operació i instal·lació dels robots industrials tradicionals.

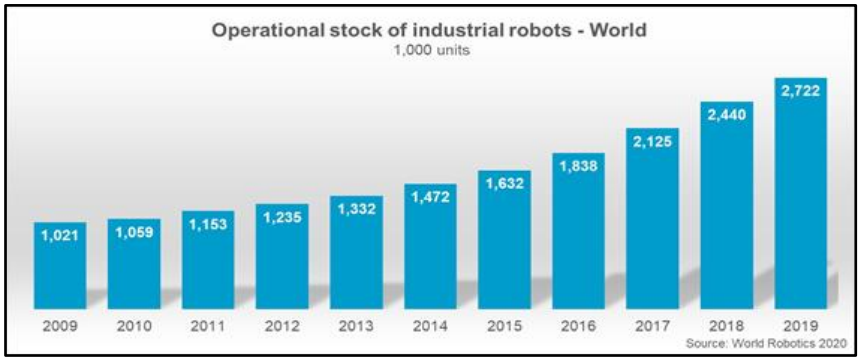


Figura 1: Nombre de robots industraisl en milers d'unitats des de 2009 fins a 2019

.

Aquests són els anomenats robots d'automatització industrial, que ocupen aproximadament un 71% del mercat de robots industrials [4]. Aquests robots tenen la capacitat de produir a grans velocitats i poden treballar en entorns que no serien segurs per a un operari, realitzant funcions amb una força i amb una precisió que són impossibles d'igualar per un treballador humà. Tot i això, aquestes màquines no estan preparades per a treballar amb persones de forma segura. Són eines de treball que estan pensades per a treballar sense intervenció humana, en un entorn controlat, que previament s'ha dissenyat per a que no es requereixi improvisació [5]. Adicionalment, són costosos y de gran pes y volum. En definitiva, requereixen d'una infraestructura tan elevada que la seva incporporació resulta extremadament cara i dificultosa per a la majoria de plantes industriasl amb un pressupost ajustat.

2.1.1.3 Els robots col·laboratius

Les innovacions tecnològiques i abaratiment de preus de l'electrònica i els sensors han promogut l'aparició d'un nou tipus de robot, de mides reduïdes, que està especialment dissenyat per a treballar amb persones. Aquest tipus de robot, s'anomena *Robot col·laboratiu* o *Cobot*.

Aquests aparells van sortir al mercat aproximadament entre els anys 2004 – 2015. El primer cobot en arribar al mercat va ser el LBR3, de l'empresa alemanya KUKA. Seguit al 2008 del robot UR5, de l'empresa danesa *Universal Robots* (UR) i el robot *Baxter* l'any 2011, dissenyat per l'empresa *Rethink Robotics*, amb seu a alemanya.

Figura 2 Robots col·laboratius UR3, Baxter i LBR, respectivament.

A diferència dels robots d'automatització industrial, aquest tipus de robots estan especialment dissenyats per a treballar amb persones. Estan equipats amb una elevada quantitat de sensors que permeten obtenir un major nivell de seguretat per als treballadors. A diferència de la resta de robots industrials, estan dissenyats per a ser fàcilment reprogramables i desmuntables per personal no expert. Això permet que es pugui fer servir el mateix robot en entorns de treball molt diferents. També treballen amb pesos i velocitats menors que els altres robots industrials però a un preu més reduït. Aquestes característiques fan dels robots colaboratius una opció interessant per a empreses

amb poca capacitat de financiació que busquen incorporar maquinaria per a automatitzar processos però que no es poden costetjar els elevats preus dels robots industrials convencionals.

Figura 3 Un operari amb un robot col·laboratiu KUKA LBR

2.1.1.4 Robots col·laboratius a l'empresa

Els cobots estan agafant cada cop una importància més rellevant al món de l'industria. Estan desplaçant als robots industrials convencionals. Algunes prediccions estimen que la quota de mercat dels cobots sigui d'aproximadament el 34% del mercat dels robots industrials [6].

Algunes de les principals avantatges tècniques que suposa la incorporació de robots col·laboratius a l'entorn de treball són:

- Les toleràncies del moviment dels robots, així com l'alta repetibilitat dels robots provoca una menor variabilitat i un augment en la qualitat dels productes finals.
- La facilitat de muntatge i facilitat de programació dels *cobots* aporten flexibilitat a a aquests dispositius. Els *cobots* es poden traslladar i reprogramar en un altre entorn en el que realitzaran tasques completament diferents. Aquesta carcaterística és interessant per a empreses en les que es realitzin diferents activitats repetitives al poder fer servir la mateixa eina per a aquestes activitats.
- Els cobots tenen una sèrie de mseusres de seguretat smolt exigents que aprofiten el conjunt de sensors del robot i la seva connectivitat per a adaptar-se al entorn en situacions inesperades.

Pertant, els *cobots* són una eina excel·lent per a automatitzar tàsques repetitives i aborrides, com les operacions de control de qualitat, manipulació de capses o realitzar acabats superficials.[7], entre altres apliacions. En un entorn amb presència de *cobots*, els treballadors deixen de realitzar tasques monòtons, podent-se dedicar a activitats que no siguin automatizables o bé poden treballar conjuntament amb el robot, obtenint un major volum de producció de millor qualitat.

Dintre dels entorns industrials, l'incorporació de *cobots* ha demostrat que permet augmentar l'eficiència del treball, ja que ajuden a reduir de forma dràstica els temps d'espera en els processos de producció. Per exemple, un estudi que es va realitzar a la empresa alemanya BMW mostra que es van reduir els temps morts d'una sèrie d'operacions en un 85% després d'incorporar a la cadena de muntatge robots col·laboratius [8] .

La versatilitat dels *cobots* permet que no només es puguin incorporar a les empreses que estan dedicades a la indústria, també poden treballar en entorns com oficines, cases, laboratoris, magatzems o en l'educació. [9] .

3 El robot UR3

3.1 Universal Robotics

L'empresa Universal Robotics (UR) és la fabricant del robot amb el que s'ha realitzat el curs. Va ser fundada l'any 2005 per tres investigadors de l'Universitat de Syddansk, a Dinamarca, mentre realitzaven els seus estudis sobre robòtica [10] Fins a dates de 2021 han fabricat set models de robots i actualment dominen el mercat dels robots colaboratius, amb una quota de mercat d'aproximadament el 47% [11].

Els *cobots* que fabrica són de mides reduïdes, amb un abast petit (500-900 mm) i poc pes(11-35 kg). La capacitat de càrrega del robot varia en funció del model,, el UR3 és capaç de treballar amb càrregues de fins a 3 kg, mentre que el robot més potent, l'UR16e, és capaç de treballar amb càrregues de fins a 16 kg.

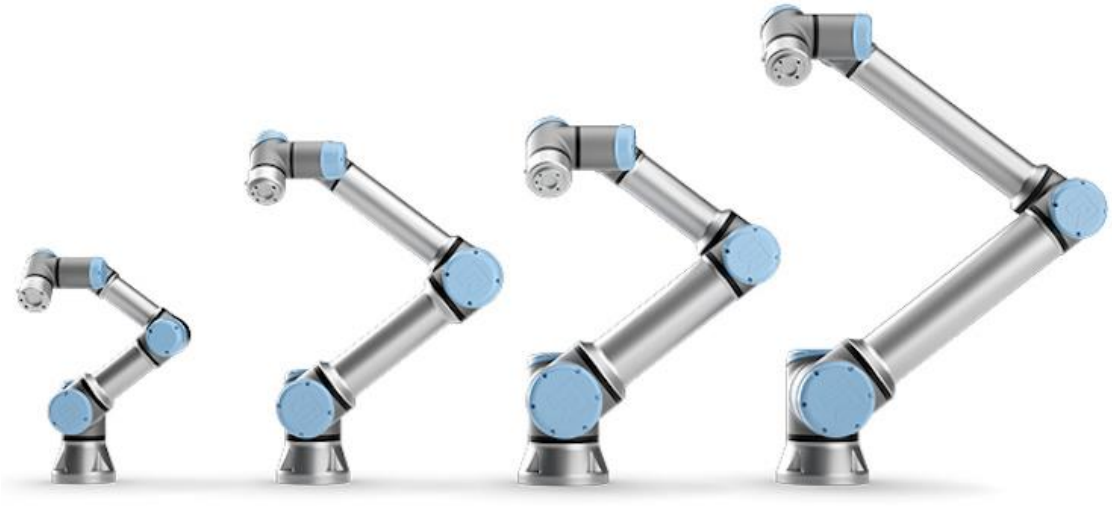


Figura 4 Ur3e, Ur53, Ur16e i Ur10e respectivament

Una de les principals característiques que diferencia a UR de la resta de competidors és la compatibilitat que tenen els robots per a funcionar amb perifèrics, gràcies al ecosistema URCaps. Aquesta plataforma funciona d'una manera similar a les botigues d'aplicacions dels telèfons intel·ligents [12].

Quan una empresa està interessada en fabricar un component per un robot UR, es posen amb contacte amb UR, que certifica si el producte és apte per al funcionament (certificat UR+). Aquest productes un cop certificats es poden instal·lar al robot, que mitjançant un paquet de software

específic per al complement, es podrà controlar amb el propi GUI del robot. Amb aquesta eina es pot fer servir qualsevol perifèric sense cap problema de compatibilitat i de forma senzilla i segura.

3.2 El robot UR3 e-Series

És el robot que s'ha fet servir és el més petit que subministra UR, el UR3 e-Series. Les especificacions tècniques d'aquest robot es troben al annex. Les mides reduïdes del robot el fan perfecte per a aplicacions on l'espai disponible és molt reduït, com ho pot ser dins d'altre maquinaria, com en una eina de control numèric.

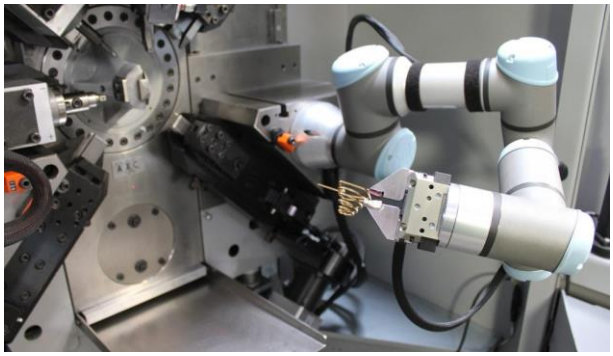


Figura 5 Robot UR3e dins d'una màquina de control numèric

3.2.1 Components del robot

El robot està format per tres elements. El braç robòtic, la pantalla de control *teach Pendant* i la capsa de control. La computadora que executa els controls del robot es troba a la capsa de control. Aquesta és l'element central de la transmissió de dades i on es troben la majoria dels ports de connexió, tant IO com el port ethernet i l'alimentació de tots els components. Les especificacions tèncniques dels diferents components es troben al Annex.

Figura 6 Conjunt d'un UR3: Braç, capsa de control i Teach pendant

3.2.2 Mecànica del robot

Estudiant el mecanisme del braç robòtic es veu que es tracta d'un robot amb 6 actuadors, tal i com estan descrits a la figura 7. Aplicant el criteri de Grübler, tenint en compte que es tacta d'un mecanisme format per 6 sòlids i 6 articulacions, podem veure que té un total de 6 graus de llibertat. Això permet que l'extrem del robot, posició on es troba l'eina que estigui utilitzant el robot, es pugui configurar en qualsevol posició dintre de l'espai de treball. Aquest està delimitat per una esfera de 1145,5 mm de diàmetre atravesada al centre per un cilindre de diàmetre 56 mm, com es mostra a la figura 7.

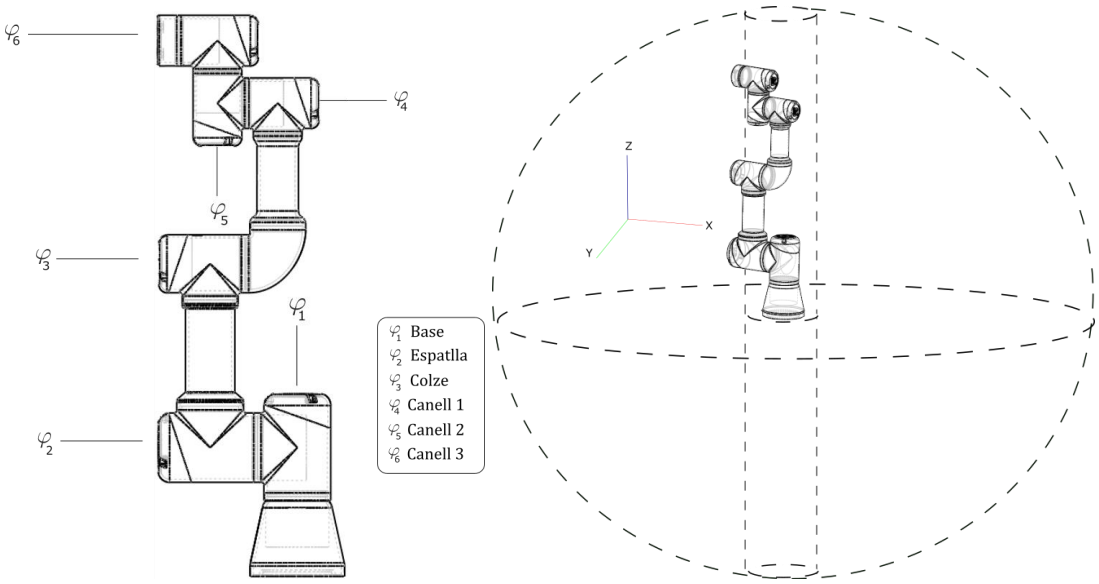


Figura 7 Articulacions del braç robòtic i de l'espai de treball del robot.

S'ha de tenir en compte que, no és recomanable treballar en les zones de treball properes als extrems d'aquest espai de treball, ja que es forcen els actuadors, que han d'aguantar esforços elevats. UR recomana no allunyar-se més de 500 mm de la base. [13].

Finalment, a l'hora de determinar en quines configuracions ha de treballar el robot s'han de considerar dos factors més, el contacte del robot amb elements de l'entorn o ell mateix i les singularitats del mecanisme.

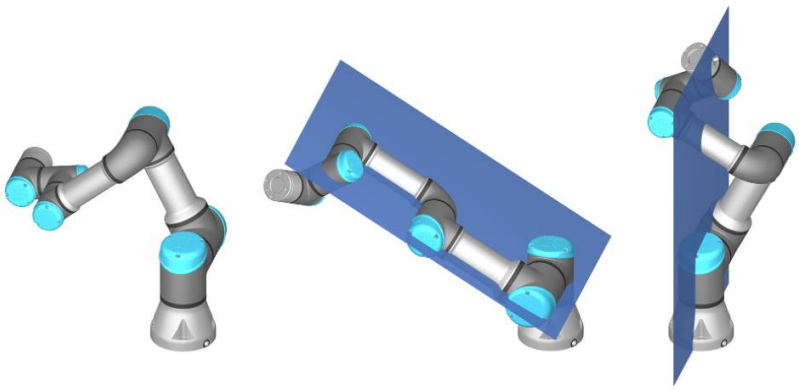


Figura 8 Diferents posicons singualrs del robot

3.2.3 Formes de controlar el robot

Hi ha diferents formes de treballar amb un UR3: El propi programari instal·lat a la teach pendant (*Polyscope*), el llenguatge de programació de ur anomenat *URscript* i altres llenguatges de programació de programari lliure, com *Python*, en el que la comunitat ha realitzat una sèrie de llibreries que fan servir els protocols de comunicació del robot per a poder-lo controlar. Un altre exemple de llenguatge de programació que es fa servir, sobretot al món acadèmic, és ROS, que es fa servir a algunes de les assignatures impartides al MUEI en l'especialitat d'automàtica o al MUAR.

Existeix molt de contingut en línia, a un preu assequible (o fins i tot de franc) que tracta sobre el funcionament de les diferents formes de comunicar-se amb el robot, per posar exemples:

- ***Polyscope**:* UR posa a la disposició de tothom els manuals d'ús de *Polyscope* i dels diferents models de robot que té disponibles. Addicionalment, existeixen cursos online de diverses empreses, com els de UrAcademy [14].

- ***Ur Script**:* De la mateixa manera que amb *Polyscope*, estan disponibles ne format online els manuals de les diferents versions de *UrScript*. Hi ha cursos disponibles al web de UR, encara que no són gratuïts. Existeixen blogs de robòttica en el que es mostren exemples de programes que s'han realitzat amb fent servir aquest llenguatge, com per exemple Zacobria [15].

- ***Python**:* Al comunicar-se el robot amb l'exteriors fent servir una sèrie de protocols especificats. Una de les llibreries de *Python* que peremt això és la ***Ur_rtde***. A la pàgina web del projecte hi ha una sèrie d'exemples, encara que no hi ha cursos disponibles sobre aquesta llibreria [16]

- ***ROS*** És una aplicació dissenyada expressament per al desenvolupament de codi per al control de robots. Hi ha molts cursos disponibles de ros en general, també alguns específics de l'implementació de ROS a rtobots UR3, encara que son de pagament. Com a exemple: The Construct [17].

Dintre de les opcions, s'ha decidit enfocar-se en els llenguatges de programació *Polyscope* i *Python,* donat a l'accessibilitat del primer i els coneixements previs adquirits de *Python* durant la realització del grau. Addicionalment, aprendre a fer servir *Polyscope* té una utilitat afegida, és el llenguatge de programació implementat a l'ordinador de control instalat al robot. Pertant, aprendre a fer servir *Polyscope* permetrà entendre amb més profunditat el funcionament del robot.

3.2.4 Perifèrics del laboratori

Una de les principals avantatges que té el fer servir hardware de les empreses associades a UR és que el software de control dels perifèric està especialment dissenyat per tal de poder-se fer servir amb el software de control del robot. Com a alternativa, es poden controlar amb mitjançant els ports d'entrada i del braç i de la capsa de control si no es compte amb un programari de control de URCaps En el cas del laboratori on es realitzaran les pràctiques, es compta amb material dels dos tipus.

3.2.4.1 Robotiq 2F-85 gripper:

Es tracta d'una pinça elèctrica paral·lela, en la que es fa servir un mecanisme de quadrilàter articulat per a realitzar un moviment de tancament de la pinça. Les especificacions tècniques complete ses troben resumides al annex.

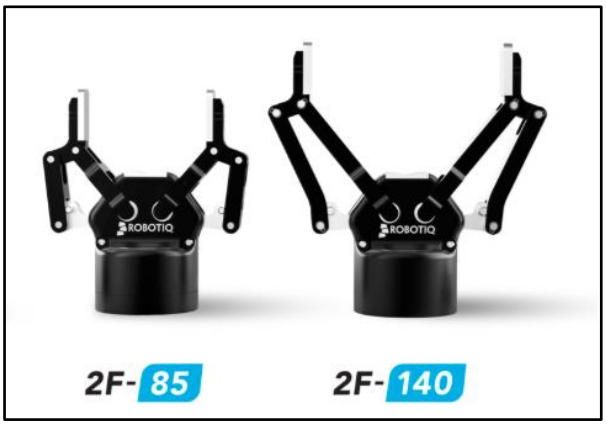


Figura 9 Models 2F-85 i 2F-140 de Robotiq. Ambdós models estan programats per a ser compatibles amb l'ecosistema UR+

L'empresa Robotiq és un desenvolupador de l'entorn UR+, de manera que el control de la pinça es pot realitzar directament des de la *Teach Pendant* instal·lant un mòdul proporcionat per *Robotiq* a la *Teach Pendant*.

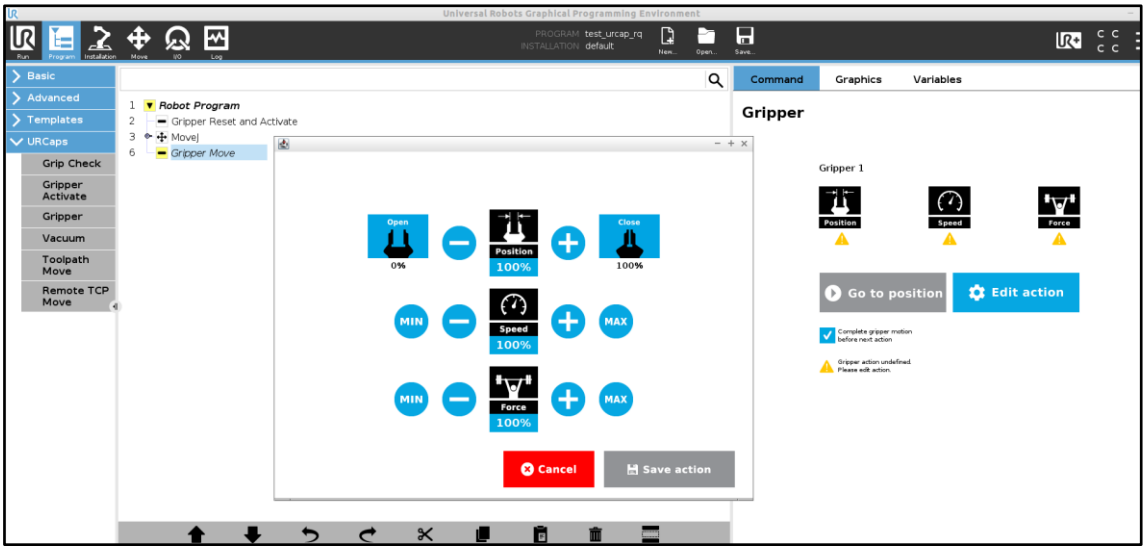


Figura 10 Visualització de Polyscope del control de la pinça robòtica.

3.2.4.2 Cinta transportadora: Dobot Mini Conveyor Belt:

Es tracta d'un dispositiu que funciona a partir d'un motor pas a pas. En activar-se una de les sortides digitals de la capsa de control, s'envia un input a un circuit, que consta d'un convertidor DC/DC i un arduino que està connectat a la cinta transportadora. En rebre aquesta senyal, l'arduino controla el motor pas a pas, produint-se un moviment de la cinta. Aquest moviment es produeix a velocitat constant.

Figura 11 Cinta transportadora disponible al laboratori.

3.2.4.3 Sensor de presència

Finalment, el laboratori també consta d'un sensor de presència, útil per a detectar blocs i implementar iun control senzill i segur per a la cinta transportador a o qualsevol altera aplicació del robot que necessiti informació sobre la presència d'un objecte present en alguna posició. Al laboratori hi ha diversos tipus de sensors d'aquest tipus, a la figura 12 es mostra un dels models dels que es diposa al laboratori. Aqueststst sensors estan configurats per enviar un senyal difital quan detecten un objecte a una distància especificada que es pot regular.

Figura 12 Sensor de presència de l'empresa Honeywell

3.2 Ensenyament dels cobots UR

3.2.1 UR Academy

Una de les millors ofertes de cursos per aprendre *Polyscope*, per la seva qualitat i el fet de que son de franc, són els cursos que ofereix UR a la seva pàgina web, això és l'anomenada UR Academy. En aquests cursos es fan servir exemples pràctics mitjançant una plataforma online, en la que la interfície es mostra a l'usuari unes imatges que simulen un comportament igual al del programa de *Polyscope* que apareix a la pantalla de la teach pendant.

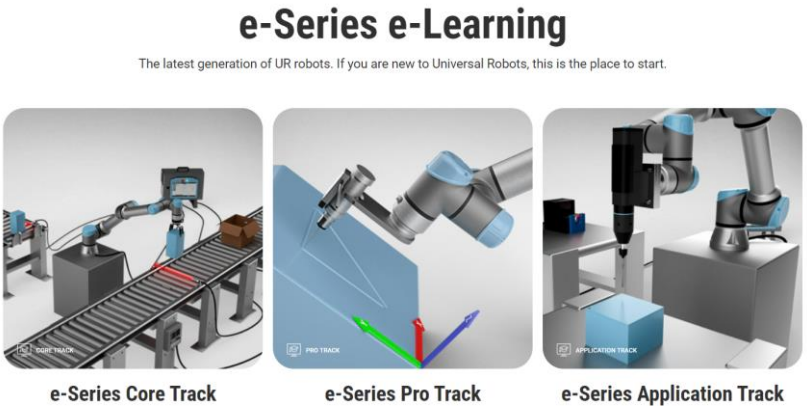


Figura 13 Diferents mòduls d'ensenyament oferits

Per tal de complementar i afegir material disponible en els cursos sobre *Polyscope* s'ha decidit realitzar material didàctic orientat a redactar una guia de les diferents comandes que permet realitzar el programa i deixar una sèrie d'exercicis resolts i pràctiques que puguin realitzar-se amb un robot UR3 amb la intenció de poder posar a prova els coneixements adquirits sobre el funcionament i programació del robot i corregir els errors que s'hagin comès durant l'aprenentatge.

4 Curs de robòtica

4.1 Metodologia docent

La metodologia docent que s'ha pensat per a aquest curs consisteix en realitzar una sèrie de classes teòriques sobre el funcionament del robot. Posteriorment es realitzaran una sèrie de pràctiques per a posar a prova el coneixement adquirit per l'estudiant. En aquestes pràctiques, l'alumne ha de resoldre una sèrie de problemes i exercicis fent servir codi de *Polyscope*. La majoria d'aquests exercicis es poden realitzar fent servir el simulador gratuït sense necessitat de fer servir el robot, encara que és especialment interessant realitzar-los amb ell.

4.1.1 Objectius d'aprenentatge del curs

L'objectiu del curs és que al finalitzar, l'alumne ha de ser capaç de poder fer servir un robot *UR3 e-series* amb el llenguatge de programació de *Polyscope*, així com ser capaç de comunicar-se amb el robot per extreure informació i controlar-lo de forma remota fent servir *Python*. Amb aquests coneixements s'espera poder fer servir el robot per a poder implementar el controlador de força del robot.

4.1.2 Material:

El material que s'ha redactat està dividit en tres parts:

- ***Manuals de comandes de Polyscope***: S'ha realitzat un manual d'instruccions de polyscope Es tracten les comandes bàsiques, comandes avançades i un de comandes de tipus *Template*, que són els tres tipus de comandes que permet realitzar *Polyscope*. S'han realitzat aquests amb l'objectiu de simplificar i clarificar els manuals redactats per *UR*, ja que aquests son força farragossos.
- ***Exercicis pràctics***: S'ham redactat una sèrie d'exercicis que s'han d'executar amb un robot o amb el simulador amb l'objectiu de posar a prova els coneixements adquirits durant les classes teòriques.
- ***Presentacions***: S'han creat tres presentacions per tal d'assistir al personal docent durant les classes teòriques quan s'hagin d'exposar i en
- ***Manuals d'instal·lació***: Per tal de facilitar la instalació de diferents programes als ordinadors dels alumnes.

Tot el material citat anteriorment es troba al annex del document.

4.2 Conceptes bàsics de *Polyscope*

4.2.1 Entorn de programació de *Polyscope*:

Al primer mòdul del curs s'exposa el funcionament del GUI de *Polyscope*. Aquest és idèntic tant en el simulador instal·lat a la computadora com el que apareix a la *Teach Pendant* del robot, de manera que el que s'aprengui al simulador es totalment aplicable al cas del robot.

A la imatge X es mostra l'aspecte de la pantalla inicial del GUI. Està dividida en tres zones:

1. **Capçalera:** Es fa servir per a navegar dins de les diferents pantalles que té disponibles *Polyscope*. Per a accedir a una certa pantalla, s'ha de seleccionar el seu icone.
2. **Pantalla:** En funció de la pestanya que es tingui seleccioanda, apareixerà un menú diferent. Es la zona principal de treball, en la que es poden escriure les comandes, programar el robot i consultar el comportament d'aquest.
3. **Peu:** En aquesta zona es pot regular l'execució del programa, així com encendre i apagar el robot.

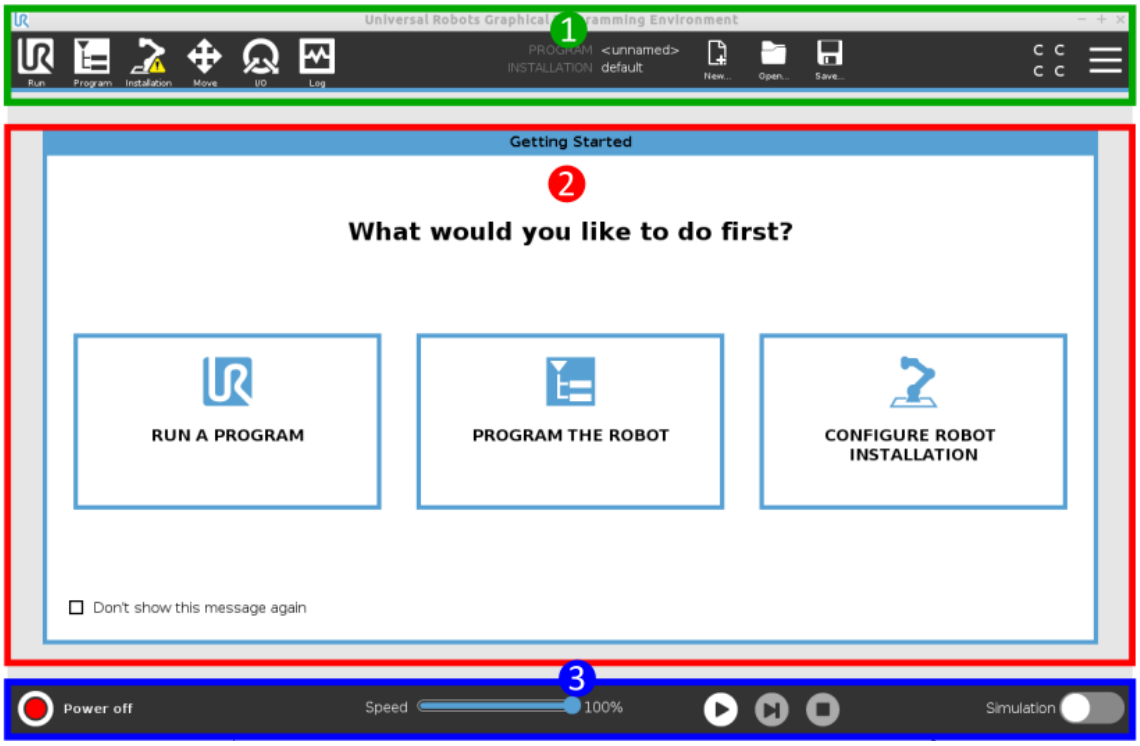


Figura 14 Visualització de la pantalla d'inici de Polyscope.

.

4.2.1.1 Icones del menú superior

Executar: És una finestra en la que es poden selecccionar programes per a executar-los.

Obrir: Obre un programa o instal·lació ja creat i guardat previament.

Programa: En aquesta finestar és on s'escriu el codi.

Guardar: Guarda un programa, instal·lació o ambdós.

Instal·lació: Aquí es poden configurar els ajustaments del braç robòtic i l'equip extern.

Manual: El robot està configurat per a poder-sse contorlar només amb la *Teach Pendant*.

Moviment: Permet moure el robot.

Automàtic: El robot està configurat per a poder controlar-se només amb ús d'interfícies alienes a la *Teach Pendant*.

IO: Permet controlar, visualitzar i configurar els estats de les entrades i sortides del robot

Configuració de seguertat: Mostra un resum. de la configuració de seguretat activa

Registre: Indica l'estat del robot (temperatura, voltatge…) , així com els missatges d'error.

Menú hamburguesa: Habilita l'accés a Ajuda, Sobre i Ajuntaments i també permet apagar el robot.

Nou: Crea un nou porgrama o instal·lació.

Ruta d'arxiu: Indica el porgrama i instal·lació oberts per al robot.

4.2.1.2 Icones del menú inferior:

Inicialitzar: Aquest és unicone indicatiu de l'estat del robot: verd(normal), groc (inactiu), vermell (apagat)-

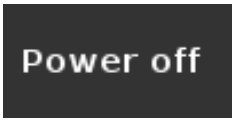
Passar: Permet executar el programa amb una única etapa.

Reproduir: Inicia el programa carregat actualment per al robot.

Parar: Atura el programa carregat actualment.

Control de velocitat: Permet reduir la velocitat amb la que es realitzen les accions del programa. Útil per a quan s'estan fent proves amb codi sense revisar.

Simulació: Permet activar o desactivar el mode simulació.



Apagat: Permet apagar o reiniciar el robot.

4.2.2 Seguretat del robot

La seguretat és un dels elements principals per a un robot co·laboratiu. Els robots d'UR compten amb una sèrie de configuracions i mesures de seguretat que minimitzen la possibilitat d'accident si es configuren degudament.

4.2.2.1 Sistema de configuracions de seguretat

El robot està dissenyat de manera que no pot funcionar sense tenir una configuració de seguretat instalada Aquestes configuracions es troben en els arxius *Installation*, que contenen tota l'informació necessària per al correcte funcionament del robot, com les característiques de l'eina que s'estigui fent servir, la configuració dels IO i tots els paràmetres de seguretat que es tinguin instal·lats.

Els paràmetres de seguretat que té disponibles el robot són els límits d'actuació del robot, els plans de seguretat, la frontera de l'eina, l'orientació de l'eina i la configuració dels ports de seguretat del robot.

4.2.2.2 Limits d'actuació del robot

N'hi ha de dos tipus, els límits del robot i els límits de les articulacions. Es poden configurar al menú *Installation*. Els límits del robot són una sèrie de variables físiques que estan relacionades amb el perill que pot implicar el robot si aquest està en funcionament. A la figura 15 es mostra la pantalla de configuració d'aquestes variables.

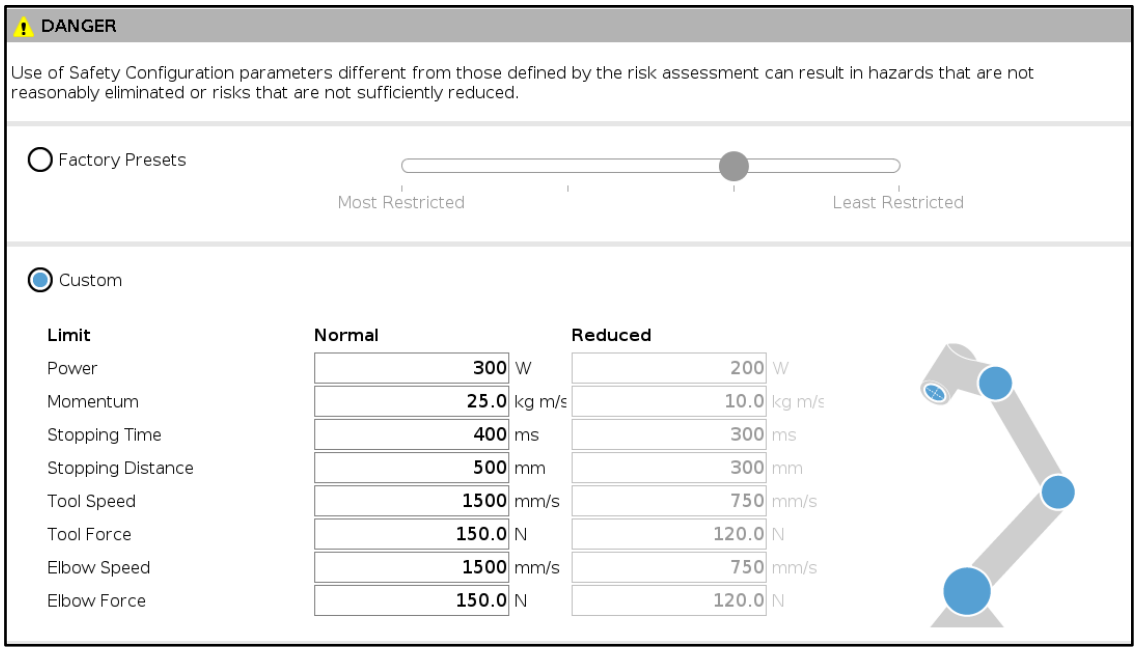
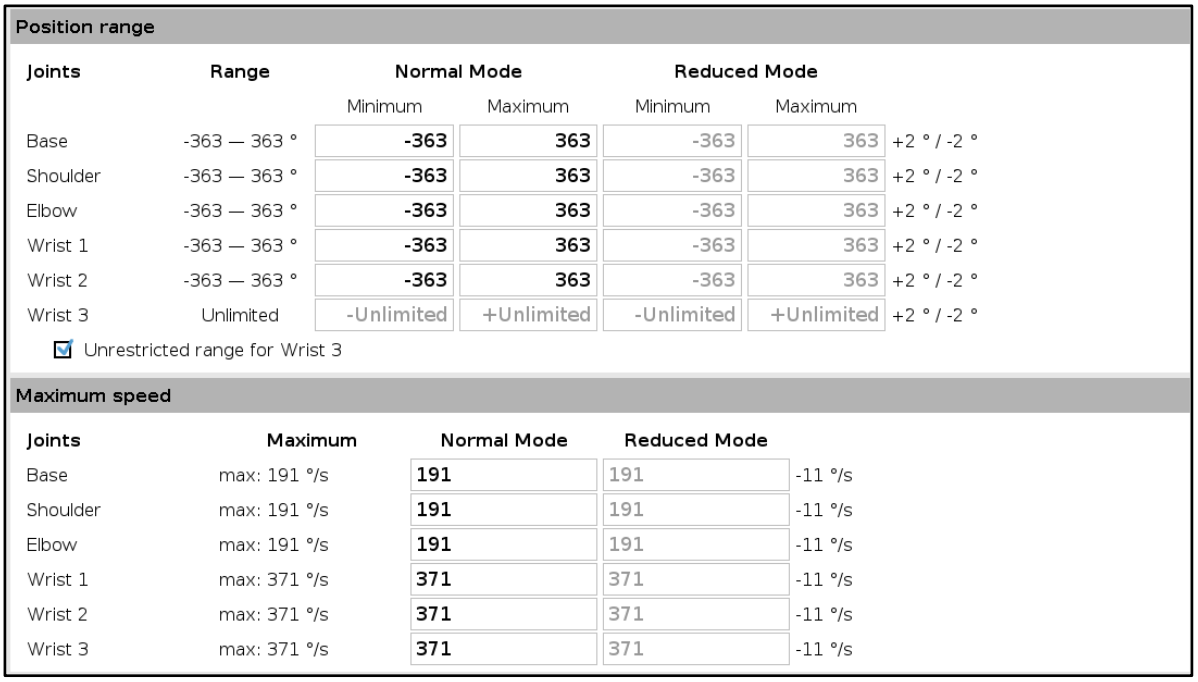


Figura 15 Pantalla de configuració dels límits del robot.

Per altra banda, esl límits de les articulacions, són el rang de valors en els que pot rotar una articulació del robot. S'ha de tenir en compte que els valors de les mesures tene un error de ± 3°. Aquesta configuració és especialment interessant quan l'espai de treball del robot és limitat, de manera que es vol mantenir la rotació d'una de les articulacions dins d'un rang determinat de valors. De la mateixa manera que els límits del robot, els límits de les articulacions es poden modificar a la pestanya *Installation*. La pantalla que apareix al GUI és la que es mostra a la figura 16.



Position range

Joints	Range	Normal Mode Minimum	Normal Mode Maximum	Reduced Mode Minimum	Reduced Mode Maximum	
Base	-363 — 363 °	-363	363	-363	363	+2 ° / -2 °
Shoulder	-363 — 363 °	-363	363	-363	363	+2 ° / -2 °
Elbow	-363 — 363 °	-363	363	-363	363	+2 ° / -2 °
Wrist 1	-363 — 363 °	-363	363	-363	363	+2 ° / -2 °
Wrist 2	-363 — 363 °	-363	363	-363	363	+2 ° / -2 °
Wrist 3	Unlimited	-Unlimited	+Unlimited	-Unlimited	+Unlimited	+2 ° / -2 °

☑ Unrestricted range for Wrist 3

Maximum speed

Joints	Maximum	Normal Mode	Reduced Mode	
Base	max: 191 °/s	191	191	-11 °/s
Shoulder	max: 191 °/s	191	191	-11 °/s
Elbow	max: 191 °/s	191	191	-11 °/s
Wrist 1	max: 371 °/s	371	371	-11 °/s
Wrist 2	max: 371 °/s	371	371	-11 °/s
Wrist 3	max: 371 °/s	371	371	-11 °/s

Figura 16 Configuració dels límits de les articulacions del robot.

4.2.2.3 Plans de seguretat

Aquesta funcionalitat permet definir el comportament del robot en funció de la regió en la que aquest es trobi. Pot ser interessant per a quan s'estigui treballant en entorns en els que s'espera trobar-se amb algun treballador dins de la fàbrica.

Per a poder definir un plànol de seguretat, és necessari donar informació sobre la posició i orientació del plà que es vol fer servir com a límit. Per a fer això, s'ha de fer servir un objecte de *Polyscope*, anomenat *Feature*. N'hi ha de tres tipus, punt, linia i pla. Per a poder definir un plà de seguretat és necessari accedir a la pestanya de *Features*, que també es troba en el menú de *Installation*.

Figura 17 Menú de sel·lecció d'installation

Per a definir completament el plà que es vulgui fer servir com a límit, es necessari marcar tres posicions, fent servir la TCP (*Tool Centyral Point*) moguent el robot. A la figura 18 es mostra el procediment que s'ha de seguir per a definir correctament una *Feature* de tipus plà.

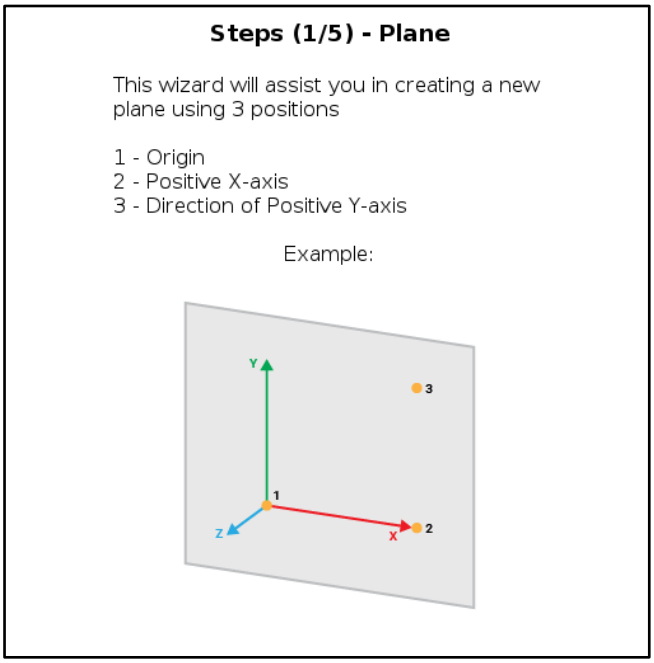


Figura 18 Procediment necessari per a definir un plà.

Un cop es té definit quin és el plà que es vol fer servir per a fixar la configuració de seguretat, es selecciona al menú l'opciói *Safety*, es selecciona l'opció de *Add Plane*. Després, es selecciona la *Feature* que es vulgui i es fixa la resposta que es vol que tingui el robot quan creui aquest plà o quan es trobi en una regió en la que Z>0.

Hi ha quatre accions diferents:

- ***Normal*:** Que impedeix que el robot creui el plà quan està treballant en mode normal.
- ***Reduced:*** Impedeix que el robot entri a la regió definida pel pla quan està treballant en mode reduït.
- ***Both***: Impedeix el pas del robot en qualsevol estat de treball.
- ***Trigger Reduced mode:*** El robot podrà creuar el plà però al fer-ho entrarà en modee reduït, un mode en el que la configuració de seguretat és més estricte i els moviments es realitzen més lentament.

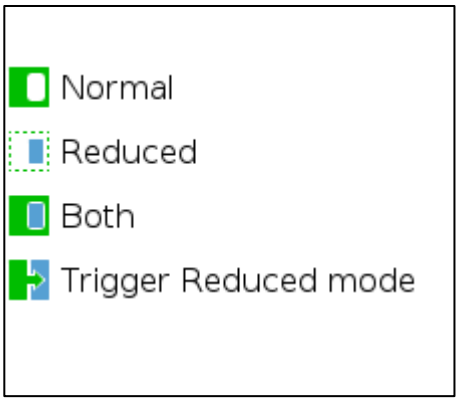


Figura 19 Comportaments del plà de seguretat

Figura 20 Missatge que apareix al creuar un plà de seguretat.

4.2.2.4 Frontera de l'eina

Aquesta funcionalitat està pensada per a eines que siguin molt perilloses o molt voluminoses. Els plans de seguretat en una configuració normal només s'activen en el cas que la TCP creui el plà. Si el robot està fent servir una eina que és perillosa en altres regions, apart del punt TCP convé definir un volum de seguretat al voltant de l'eina. Un cop s'hagi definit aquest, els plans de segruetat s'activaran quan entrin en contacte amb aquesta regió. Per a configurar aquesta opció, cal prémer el botó de *Tool Position* del submenú *Safety* dins de la pestanya *Intsallation* (firgua X). Després s'ha d'indicar el centre i el radi del volum de seguretat.

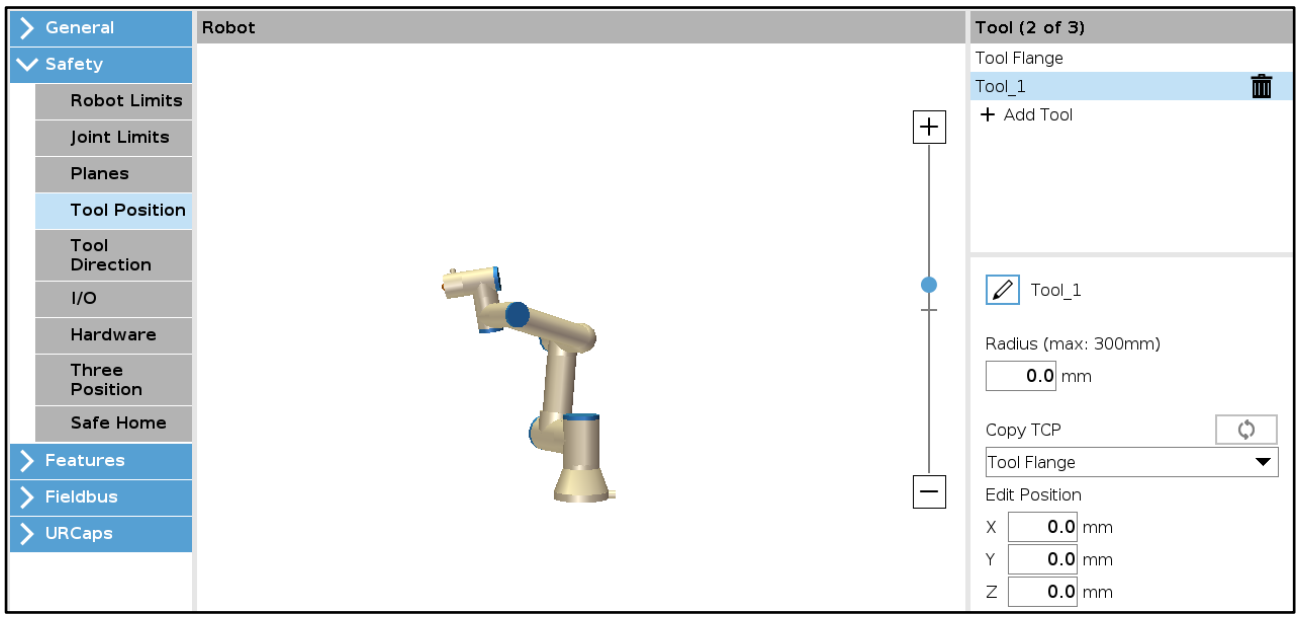


Figura 21 Configuració de la frontera de l'eina

4.2.2.5 Orientació de l'eina

Es poden configurar orientacions de l'eina no vàlides. Això pot ser útil en el cas que el robot estigui treballant amb una eina tallant, per exemple. En aquest cas seria convenient que el robot no orientés

l'eina en una posció perillosa. Per a configurar això cal accedir a l'opció *Tool Direction* del submenú *Safety* i seleccionar una *Feature*. Posteriorment, cal que es seleccioni la direcció en la que es troba l'eina variant els angles de *Pan* i *Tilt*. Desrpés d'això s'ha de seleccionar l'angle de desviació màxim permès i quina serà l'acció que executarà el robot quan la direcció establerta surti del rang de desviació.

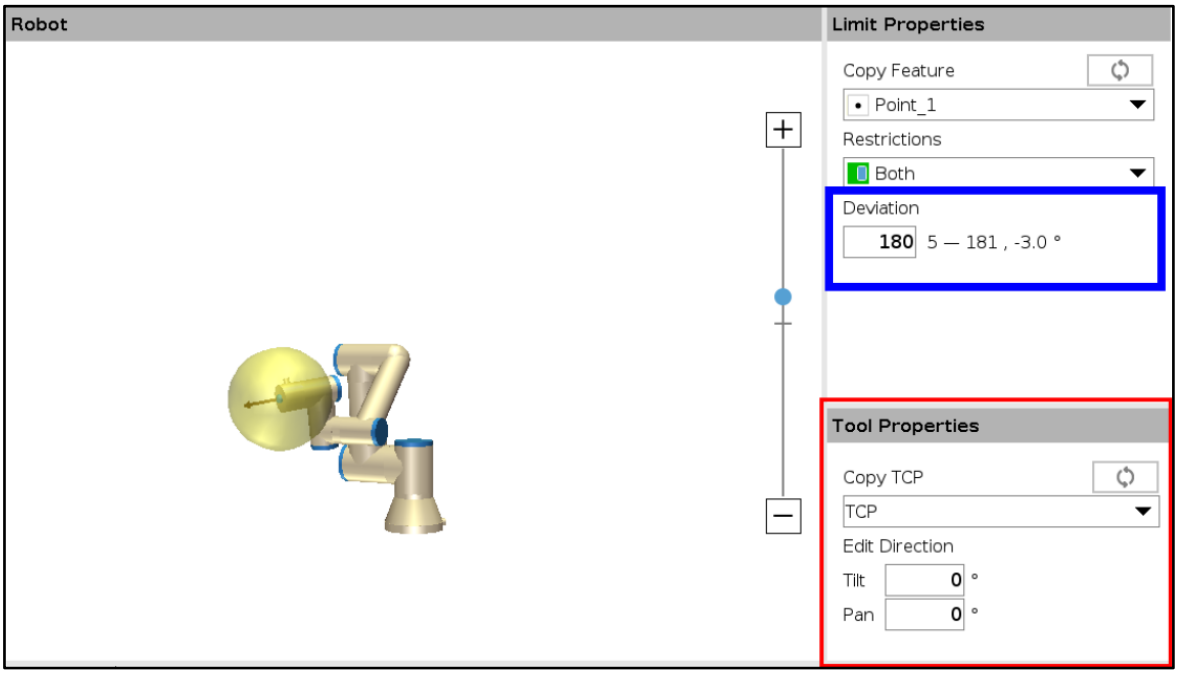


Figura 22 Configuració del rang d'orientació de l'eina. Devsiació permesa (blau) i definició de la direcció de l'eina (vermell).

Figura 23 Reducciói gradual de l'àngle de desviació permès per l'eina.

4.2.2.5 IO de seguretat

El robot UR3 està dissenyat per a treballar en entorns variables. Per a poder adaptar-se als diferents entorns, té habilitat suna sèrie de ports d'entrada i sortida per a connectar sensors o altres computadores amb els que s pot comunicar. Alguns d'aquests ports estan dissenyats per a fer-se servir per a implementar sistemes de seguretat. N'hi ha de dos tipus, els configurables i els no configurables. Els ports no configurables no es poden tractar amb *Polyscope* i estan sempre habilitats, de manera que no es tractaran en aquest apartat.

4.2.2.5.1 Inputs de seguretat

Els inputs configurables de seguretat sevreixen per a que el robot realitzi una operació específica un cop es rebi un senyal en algun d'aquests ports. Les accions que e spoden configura amb els inputs de seguretat són les següents.

- **Emergency Stop:** En rebre una senyal, el robot realitzarà una parada d'emergència. Aquest tipus de parada es equivalent a la que es du a treme quan es prem el botó de parada d'emergència de la *Teach Pendant*. En realitzar-se, el robot activarà els frens de seguretat i per a reactivar-se s'haurà de tornar a encendre el robot.

- **Reduced Mode**En rebre un senyal en un port amb aqueta funcionalitat configurada, el robot entrarà en mode reduït, com si hagués creuat un pla de seguretat amb configurat amb *Reduced mode*.

- **Safeguard Reset:** A diferència de les parades d'emergència, les parades de seguretat es produeixen de forma més habitual i són menys severes. Aquest tipus de parada és la que es realitza quan el robot creua un plà de segureat configurat en *Both*, per exemple. El robot es para i pot tornar al funcionament normal després d'una indicació del usuari mitjançant la *Teach Pendant*. En estar un port configurat amb aquesta funcionalitat, només permetrà tornar a l'execució normal del programa en el cas que aquets port rebi un senyal.

- **3-Position switch:** Aquesta funcinalitat limita la capcacitat de moviment *Freedrive*. A l'apartat 4.2.3 es parla sobre com és el moviment amb freedrive activat. Quan un port de seguretat es troba configurat amb aquesta opció, només es podrà activar l'opció *Freedrive* quan es rebi un senyal en aquest port i es s'activi aquets tipus de moviment a la *Teach Pendant*.

4.2.2.5.2 Outputs de seguretat

Els outputs de seguretat es fan servir per a que el robot envii un senyal per a transmetre informació sobre el seu estat de forma segura. Es pot habilitar l'enviament d'un senyal quan s'hagi realitzat una parada d'emergència, quan el robot està en moviment, quan es troba parat, quan no es pot parar o bé per a que indiqui si el robot està en mode reduït o si es troba treballant de forma normal.

La configuració dels ports de seguretat es realitza també a la finestra *Installation*, clicant l'opció *I/O* al submenú *Safety*. A la figura 24 es mostra la visualització de la pantalla de comnfiguració dels ports de seguretat.

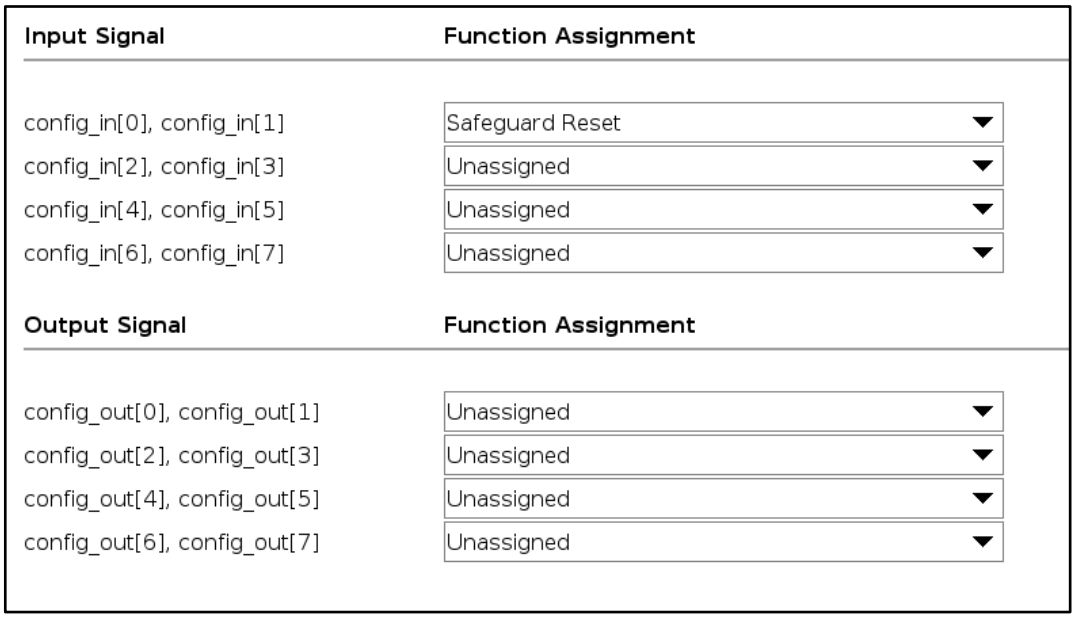


Figura 24 Pantalla de configuració dels ports IO de seguretat.

4.2.2.5.3 Ports físics de seguretat

Els ports de seguretat es troben a la capsa de control del robot, marcats de color groc. Es poden diferenciar els ports configuyrables i els no configurable en funció del color de les fonts de les entrades. Si és de color negre, es tracten de ports configurables, si són de color vermell són ports no configurables. A la figura 25 es mostren aquests ports. És important veure que en la configuració que es mpostra a l'imatge, els portts de seguretat no configurables estan connectats a une ssortides de 24 V. Això es fa així perquè els ports de seguretat s'activen en rebre un senyal de zero lògic per qüestions de seguretat.

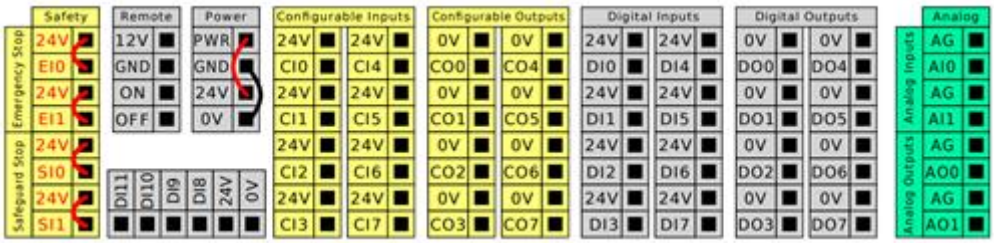


Figura 25 Ports IO disponibles a la capsa de control del robot UR3e.

4.2.3 Moure el robot

Si es vol moure el robot sense necessitat de programar-lo es pot fer mitjançant la finestra *Move*. A aquesta finestra es mostra en temps real quina és la posició del robot i es pot indicar que el robot es mogui de tres formes diferents:

1. Indicant la direcció en la que es vol moure i orienti el *Tool Central Point* (TCP), que és el punt en el que se suposa que treballa l'eina que té el robot.
2. Indicant les coordenades que es vol que tingui el TCP:
3. Modificant la configuració del mecanisme del robot, moguent le sbarres deslliçants que indiquen el valor de rotació de les diferents articulacions.

4. Habilitant el moviment en mode Freedrive. En activar aquesta opció de moviment, el robot es podrà moure de manera lliure. És a dir, l'usuari que el fassi servir podrà agafar el robot i moure'l sense que aquest oposi resistència. Aquesta opció és molt útil per a definir els punts de pas del robot, ja que amb aquesat opció el robot és extramadament fàcil de moure, fins hi tot en posicions complicades.

Per a poder morue el robot fent servir aquesat pestanya és necessari que el mode de control estigui en mode manual. En cas contrari la pestanya *Move* serà inaccessible.

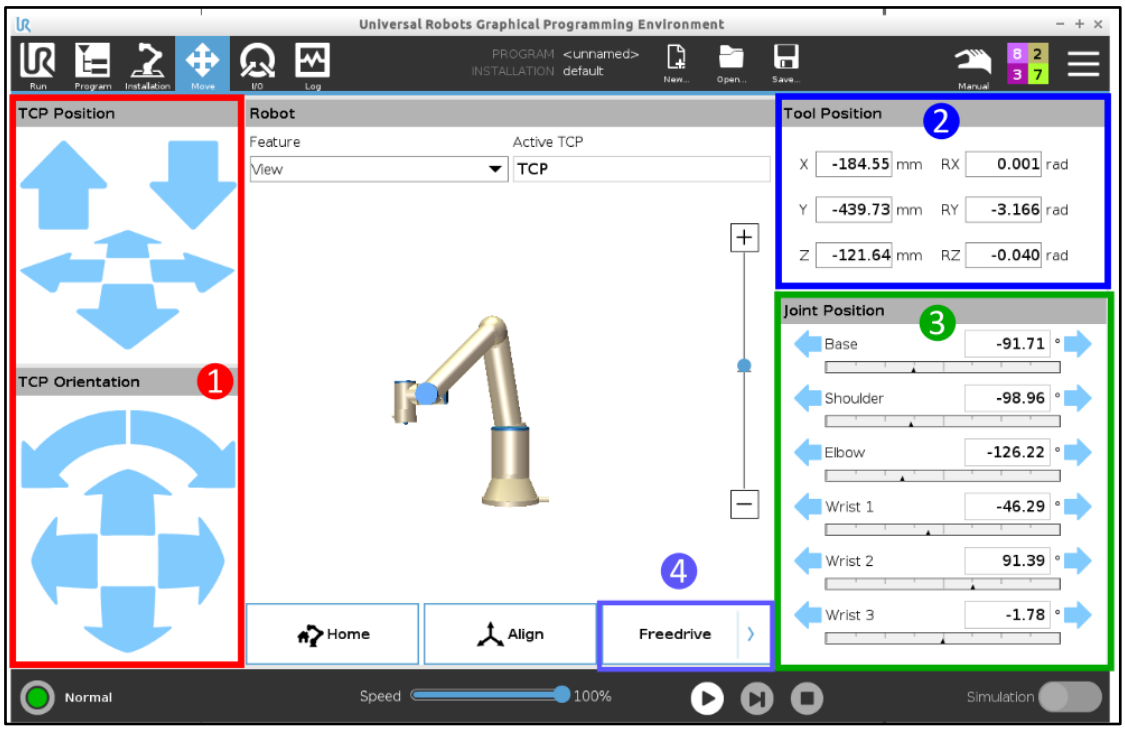


Figura 26 Visualització de la finestra Move del GUI de Polyscope.

3.5 Programar el robot

La creació i edició de programes per al robot es realitza a la pantalla program. Aquesta està dividida en tres parts:

1. Menú de selecció de comandes.
2. Arbre de programa.
3. Pestanyes auxiliars:
    a) Pestanya command: En aquesta pestanya es configura la comanda que estigui seleccionada dins de l'arbre del programa.
    b) Graphic: Mostra una representació gràfica del robot en temps real.
    c) Variables: Mostra les variables existents i el seu valor en temps real.

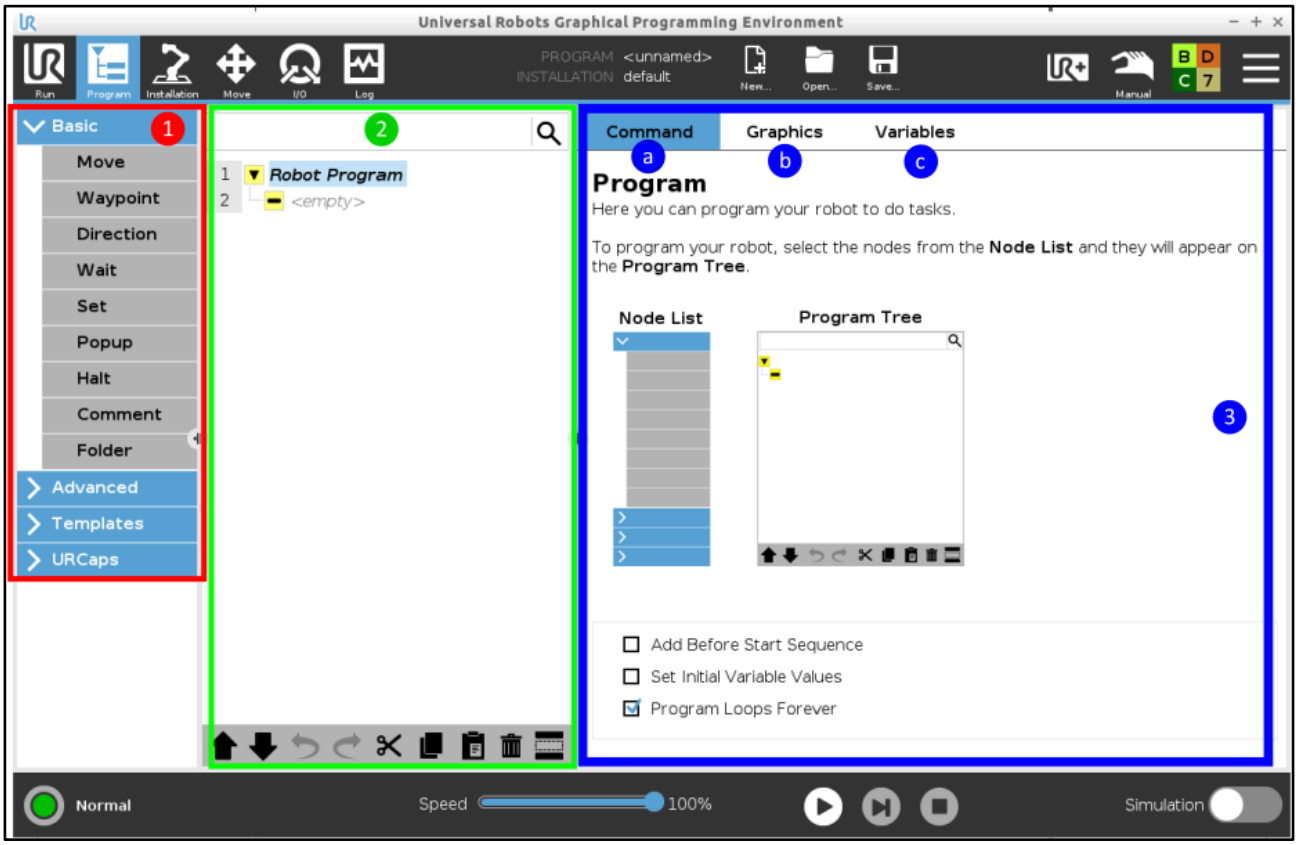


Figura 27 Visualització de la finestra Program del GUI de Polyscope.

4.2.4 Estructura del programa

El codi s'estructura en branques diferents. Dintre de cada branca, s'executa el codi de forma ordenada, de més amunt a més avlalt. Exsteixen quatre tipus de branca diferents:

- ***Before Start***: Aquesta branca del programa s'executa només al inci del programa, està pensada per a prepara el robot avans de l'execució del programa.
- ***Robot Program****:* És el cos del programa, en aquesta branca s'ecriuen les funcions principals del programa.
- ***Thread***: Aquesta comanda avançada permet escriure una branca paral·lela a la de *Robot Program*. Aquesta comanda es va dissenyar per a controlar esdeveniments sense interrompre l'execució de codi dins de la branca principal. És especialment útil per a gestionar IO, per a gestionar el funcionament d'una llista transportadora per exemplee, o l'enviament i recepció de dades fent servir les interfícies de comunicació.
- ***Event***: Aquesta comanda és molt similar a *Thread*, amb la diferència que només s'executa el codi dins de la branca quan es compleix una certa condició, mentre que el codi de *Thread* s'executa de forma automàtica. *Event* es fa servir per a gestionar situacions poc freqüents. Només es pot executar una branca *Event* a la vegada. Qualsevol altre *Event* que s'activi quan s'està executant el codi d'un altra *Event* serà ignorat. Per aquesta raó es recomana fer servir aquesta branca només per a situacions poc habituals.

Totes les comandes que s'executen en paral·lel estan subjectes a la possibilitat de que es produeixi algún conflicte, per exemple, que dues branques diferents executin un moviment diferent a la vegada. En el cas que això passi, el robot donarà un error i s'aturarà el programa. Per tal d'evitar inconveniets UR recomana fer servir només comandes de moviment a la branca *Robot Program* i deixar la gestió d'IO i de la connexió del robot a les altres branques.

4.2.5 Comandes bàsiques

Aquestes es poden dividir en quatre blocs diferents: moviments, IO, aspecte de codi i comandes de control de flux.

4.2.5.1 Tipus de Moviments

- ***Movej***: Es tracta d'un tipus de moviment no lineal a l'espai, en el que les articulacions realitzen la mínima rotació possible. Amb això s'aconsegueix la màxima velocitat del moviment, ja que els actuadors han de realitzar un menor recorregut que amb els altres tipus de moviments.
- ***MoveL***: Es tracta d'un moviment lineal. Aquest moviment, degut als actuadors, serà necessariament més lent que la resta de moviments que es poden dur a terme amb el robot. És especialment útil quan és important la trajectòria que ha de seguri el robot.
- ***MoveP***: És un moviment lineal, amb la característica de que aquest es realitza amb velocitat constant. Està pensat per a realitzar aplicacions en les que això sigui un factor determinat, com un procés de soldadura o encolat. Per a poder realitzar aquest moviment és necessari evitar les trajectòries brusques. Aquesta funció habilita modificar l'opció de *Blend Raduis*. En la que s'introdueix un radi d'arrodoniment dels moviments per tal de garantir que el desplaçament es realitzi de forma suau i a velocitat constant.
- ***Circle***: Es tracta d'un moviment circular. Es defineix amb el GUI de Polyscope indicant tres punts de pas: Punt inicial, punt de pas i punt final.

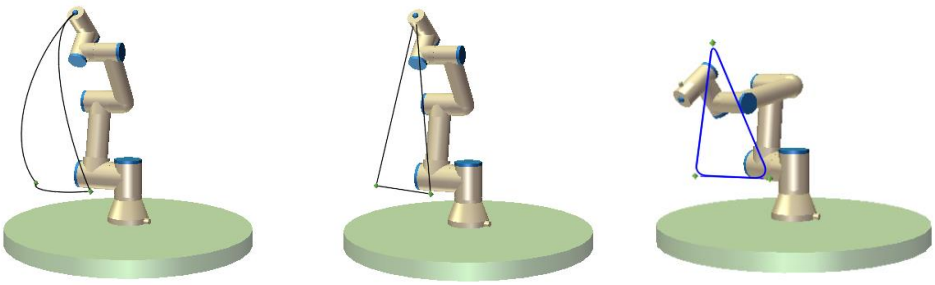


Figura 28 Moviments J,L i P respectivament.

4.2.5.2 Punts de pas i moviment indefinit:

- ***Waypoint***: Amb aquesta comanda serveix per a indicar els puts de pas del robot. Waypoint, en català punt de pas, defineix els punts per als que ss'ha de pasar en un determinat moviment. En funció del tipus de moviment que s'hagi seleccionat, el robot realitzarà una sèrie de càlculs per a trobar la trajectòria del moviment, així com velocitats i acceleracions requerides per a realitzar-la. Hi ha tres formes de definir un punt de pas:
    - ***Fixe***: S'indica la configuració que es vol per al robot moguent-lo. També es pot definir la posició indicant les coordenades del TCP indicant la referència que es fa servir o indicant els valors de les rotacions de les articulacions del robot.
    - ***Relatiu***: Es defineix un vector de 6 coordenades en el que s'indica quina ha de ser la nova posició del TCP i la seva orientació. El robot calcularà automàticament quina és la posició final en funció de la pòsició en la que es trobi i aquest vector.
    - ***Variable***: Es guarda en una variable la posició que es vol que prengui el robot. Aquesta posició és una posició absoluta que es defineix a partir d'una llista de 6 elements en la que es guarden la posició i orientació de la TCP. Alternativament es pot fer servir un tipus de variable anomenat *pose* en el que s'han de guardar les rotacions de les diferents articulacions del robot.
- ***Direction***: És un subtipus de moviment lineal. La comanda requereix la definició d'un vector. El robot mourà la TCP en aquesta direcció de forma indefinida fins que es compleixi alguna de les següents condicions:
    - Que es compleixi una expressió lògica fixada per l'usuari.
    - Que es detecti un contacte fent servir el sensor de força incorporat al robot.
    - Que es superi un cert deplaçament, fixat per l'usuari.
    - Que una de les entrades IO prengui un cert valor. Es poden fer servir tant entrades digitals com analògiques. És especialment útil si hi ha sensors connectats al robot.

4.2.5.3 Entrades i Sortides

- ***Set***: Aquesta comanda permet controlar els valors dels ports de sortida del robot, tant digitals com analògics, així com altres funcionalitats:

- Fixar valors dels ports digitals (Alt/Baix) i valors de voltatge i intensistat dls ports analògics.
- Pot emetre polsos en els ports digitals. Això està pensat per a que el robot es pugui comunicar amb perifèrics fent servir senyals PWM.
- També permet canviar la configuració de l'eina que s'està fent servir. Això és especialment útil per a les situacions d càrrega i descàrrega, en les que varien tant el pes com la posició del centre de gravetat de la càrrega de l'eina.

4.2.5.4 Aspecte de codi

- ***Comment***: Permet afegir comentaris, per tal de faccilitar l'enteniment del codi.
- ***Folder****:* Es poden agrupar comandes dins d'una carpeta, deixant el codi més entenible.

4.3.5.5 Control de flux bàsic

- ***Wait***: És una comanda que permet aturar l'execució del codi sense la necessitat d'aturar el programa. Té diferents opcions:
  - Esperar un temps indicat per l'usuari.
  - Esperar a un cert valor d'una entrada, tant digital com analògica.
  - Esperar a que una expressió lògica definida per l'usuari sigui certa.
- ***Popup***: Mostra a la *Teach Pendant* un missatge que atura l'execució del programa. Posteriorment l'operari del robot pot triar si acabar amb el programa o continuar.
- ***Halt***: Atura per complet l'execució del programa.

4.2.6 Comandes Avançades

Hi ha quatre grups diferents de comandes avançades. Són les de control de flux, crida de programes, gestió de variables i comandes d'asssitència. Auqetses comandes, combinades amb les bàsiques, permeten controlar absolutament el funcionament del robot.

4.2.6.1 Control de flux avançat

- ***Loop***: Permet implementar bucles. Té diferents formes de
- ***If...else***: Permet implementat condicionals.
- ***Event***: Obra una branca de codi que s'executa en paral·lel al robot en el cas de que es donguin unes certes condicions. Pensada per a controlar esdeveniments poc usuals sense

la necessitat d'aturar l'operació del robot, com pugui ser l'emissió de senyals mitjançant els ports de comunicació del robot.

- ***Thread***: Crea una branca de codi que s'executa en paral·lel a la resta de branques. S'executa de forma automàtica, a diferència de *Event*.

4.2.6.2 Crida de programes

- ***SubProg***: Crida i executa un subprograma de codi creat amb *Polyscope*.
- ***Script***: Executa una línia o programa de codi escrit en *UrScript*.

4.2.6.3 Gestió de variables

- ***Assignment***: Permet crear o modificar variables. No es pot canviar el tipus de varuable quan s'intentin modificar.
- ***Timer***: Permet crear o gestionar una variable de tipus temporitzador.

4.2.6.4 Assitència

- ***Screwdriving***: És una comanda que facilita la prigramació del robot a l'hora de realitzar operacions de cargolat i descargolat.
- ***Home***: És una comanda que mou el robot a una configuració de seguretat

4.2.7 URCaps

- ***Grip Check*:** Aquesta comanda deteta si està instal·lat correctament un actuador de *Robotiq*.
- ***Gripper Activate*:** Aquesta comanda executa el codi de *URSim* rq_activate_and_wait(). En executar-se, s'activen totes les pinces connectades.
- ***Gripper Move***: Aquesta comanda controla la posició, força i velocitat dels dits de la pinça. És la comanda que s'ha de fer servir per tal d'obirr i tancar una pinça mecànica.
- ***Vacuum***: Aquesta comanda permet controlar la força de succió que realitza un actuador pneumàtic. Està pensat per a altres productes de *Robotiq* que no es tenen al laboratori.

4.2.8 Templates:

Per a acabar l'exposició de les comandes, s'ha realitzat una explicació teòric ade les diferents comandes de tipus template que té *Polyscope*. Aquestes comandes aconsegueixen simplificar tasques que serien d'altra manera molt repetitives, com ho son un paletitzat o despaletitzat, que consisteix en la repetició d'un exercici de *Pick and Place* (agafar i colocar) o el seguiment d'un objecte en una cinta transportadora. Les commandes de tipus Template (plantilla) que incorpora *Poyscope* són *Seek*, *Force*, *Palletizing* i *Conveyor Tracking*.

- ***Seek:*** Eina que simplifica accions d'apliament i desapilament.
- ***Force*****:** Asegura que una certa acció es realitza amb una força adequada.
- ***Palletizing*****:** Eina que assisteix per a dissenyar aplicacions de paletitzat i despaletitzat.
- ***Conveyor Traking*****:** Eina que dóna suport en el seguiment d'objectes en cintes transportadores.

Al annex s'han incorporat el material docent que s'ha realitzat sobre aquests apartats.

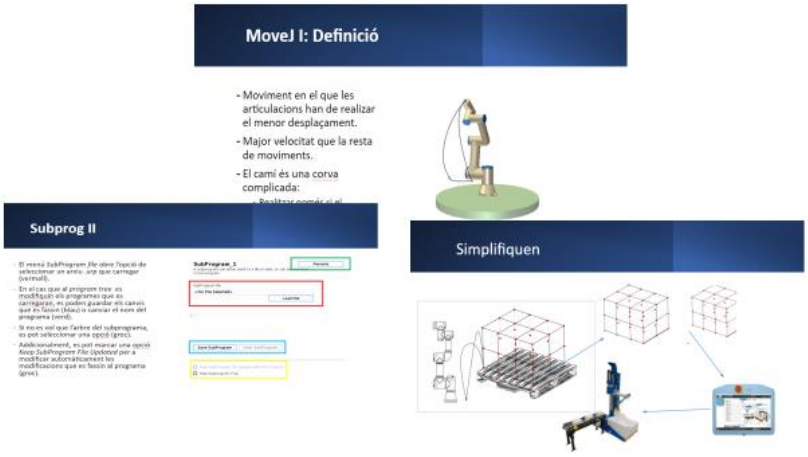


Figura 29 Imatge del material docent incorporat al annex.

4.3 Exercici: Escriure amb el robot UR3

Un cop vistes les comandes que es poden fer servir amb *Polyscope*, veurem un exemple en el que es fan servir algunes de les comandes que s'han comentat anteriroment. En aquest exercici el robot agafa un rotolador i escriu diferents lletres o paraules, en funció dels paràmetres que indiqui l'usuari fent servir la *Teach Pendant*. Es tracta d'un programa força senzill en el que es fan servir varies de les comandes bàsiques i avançades que s'han comentat anteriorment. A la figura 30 apareix el codi que es pot visualitzar al GUI de *Polyscope*.

4.3.1 Branca BeforeStart

Es realitzen dues tasques molt senzilles, s'activa la pinça robòtica fent servir al comanda *Gripper activate* i *Gripper Open* per tal de tenir la pinça preparada i en posició oberta.

4.3.2 Branca Robot Program

Primer es mou el robot a la posició boli, fent servir la comanda *MoveJ*. Posteriroment es fa servir un *Popup* per a enviar un missatge a l'usuari: "Inserir un rotolador". En aquest moment, l'usuari ha d'aguantar el rotolador a la punta de la pinça del robot. Posteriorment es prem el botó *Continue* que apareix a la pantalla i la pinça es tanca. Un cop acabat el procediment de tancat, es recoloca el rotolador si és necessari. Després del tancament de la pinça apareixerà un nou *Popup*. Quan estigui correctament col·locat el rotolador s'haurà de premer *Continue* i, posteriorment es començarà el cos del programa (linia 10).

```
1  BeforeStart
2    Gripper Activate
3    Gripper Open (1)
4  Robot Program
5    MoveJ
6      boli
7    Popup: Inserir el rotolador
8    Gripper Close (1)
9    Popup: Iniciar programa
10   Loop 7 times
11     MoveJ
12       Origen
13       Tipus_oper:='Introduir el tipus d'operació'
14     If Tipus_oper≟1
15       Call A
16     If Tipus_oper≟2
17       Call S
18     If Tipus_oper≟3
19       Call U
20     If (Tipus_oper≠1) and (Tipus_oper≠2) and (Tipus_oper≠3) and (Tipus_oper≠4)
21       Popup: Tipus d'operació no vàlid, insereix 1,2,3 o 4
22     If Tipus_oper≟4
23       Call imalive
24   MoveJ
25     Acabat
26   Popup: Recoloca el rotolador
27 imalive
70 U
76 S
84 A
```

Figura 30 Codi de Polyscope de l'exercici d'escriptura

Aquest consisteix en un bucle que itera 7 vegades. El bucle s'ha implementat amb la comana *Loop*.

Dins d'aquest bucle es realitza comença amb un moviment del robot a una posició d'espera, denominada Origen. Posteriorment es fa ús de la comanda *Assignment*, configurada per a que l'usuari hagi d'introduir un número. Després el programa evalua el valor introduit per l'operar mitjançant una cadena de *IfElse*. En funció del valor de la variable, el robot executa la comanada *Subprogram*, que crida el programa *A.urp*, *S.*urp, *U.urp* o *imalive.*urp en el que el robot realitza un

seguit de comandes *MoveL* per a escriure les lletres A, S, U o la frase *I'm alive*, estic viu en anglès. En el cas que el nombre introduït per l'usuari no tingui assignat cap tipus de comanda, el robot farà servir la comanda *Popup* per indicar que hi ha hagut un error i, durant aquesta iteració, el bucle no realitzarà cap funció. Un cop el robot hagi realitzat 7 iteracions del bucle, el robot es mourà a la posició *Acabat*, amb un moviment de tipus *MoveJ* .

4.3.3 Explicació del codi del programa

4.3.3.1 Gripper

Per a fer servir la pinça robòtica cal fer servir les comandes de URCaps que la controlen. A cada programa on es vulgui fer servir aquest dispositiu s'ha d'inclorue la comanda *Gripper Activate* que habilita la configuració de la pinça robòtica. Després s'ha d'indicar quin ha de ser el moviment que ha de realitzar la pinça robòtica., fent servir la comanda *Gripper*. S'ha de seleccionar la comanda dins del submenú *UR Caps* i configurar prement el botó *Edit action* que apareix a la secció *Command*. Després apareixerà una petita finestra en la quees pot indicar la posició en la que es vol que es coloqui la pinça i amb quina força i velocitats ha de realitzar aquest tancament o obertura.

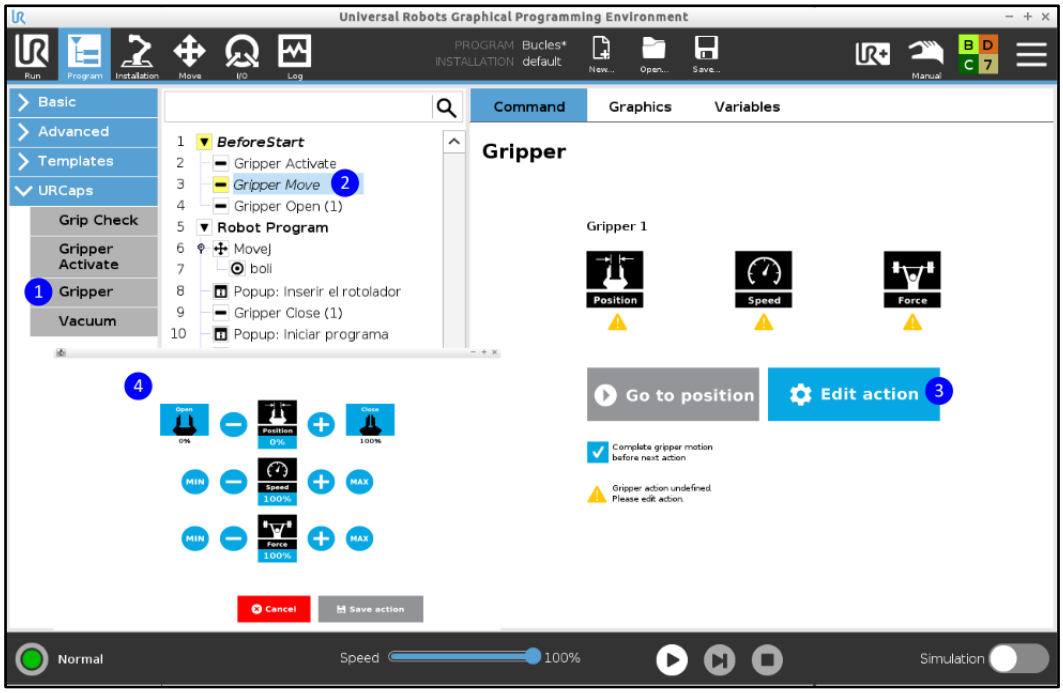


Figura 31 Configuració d la comanda Gripper move.

4.3.3.2 MoveJ i Waypoint

Per a definir els moviments del robot es fa servir la comanda *MoveJ*. Un cop seleccionada s'ha d'obrir el menú desplegable dins de la pestanya *Command* i seleccionar el tipus de moviment que

es vol fer servir. Un cop seleccionat el moviment, s'ha de configurar la velocitat i acceleració amb les que es vol que res realitzi el moviment.

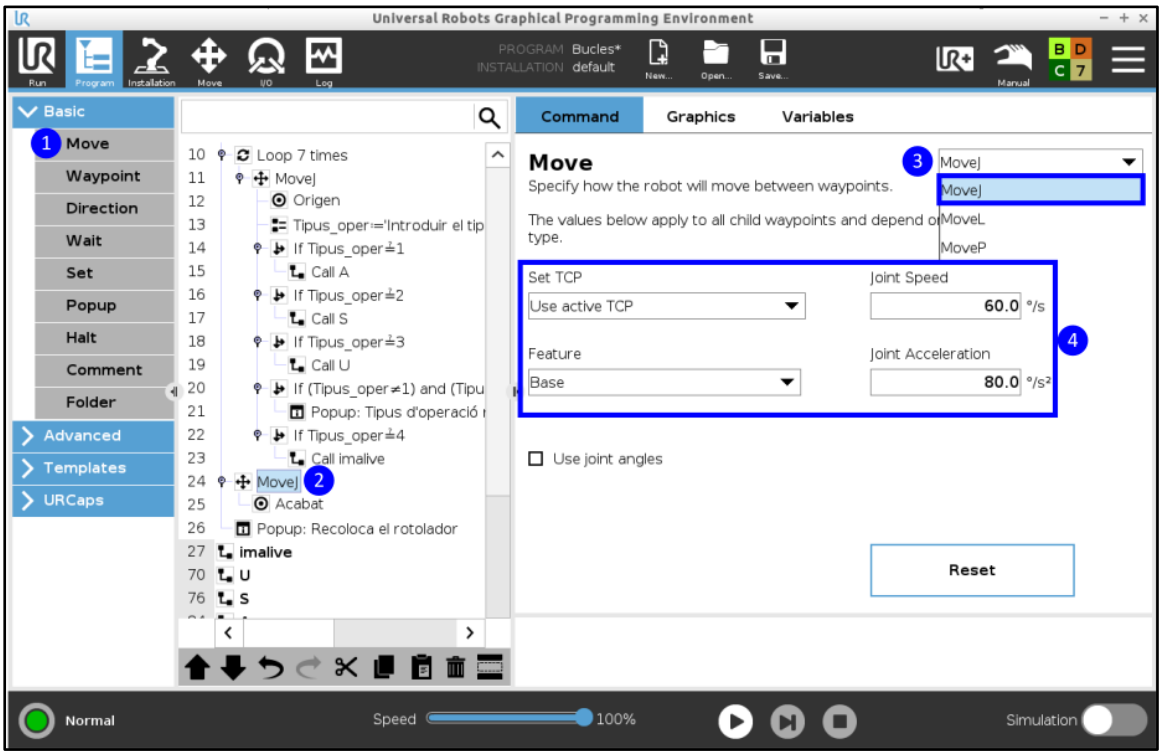


Figura 32 Configuració d'un moviment MoveJ.

Els moviments del programa principal que s'han realitzat estan programats amb la comanda *MoveJ* perquè la trajectòria que recorre el braç robòtic no es rellevant. En els subprogrames els moviments son de tipus *MoveL*, ja que en aquest cas, com que es vol mantenir el contacte entre el rotolador i el paper, és necessari que la trajectòria del punt de contacte sigui plana.

Un cop s'ha definit el moviment, cal indicar el punt objectiu al que es vol que el robot es mogui. Per a fer això cal crear un *Waypoint* dins de la comanda *Move* en la que es té la configuració del moviment. Primer s'ha de seleccionar el moviment on es vol configurar el punt de pas, després s0ha de premer la comanda *Waypoint* del submenú *Basics* i procedir a la definció del punt de pas. La forma més habitual de definir-los es moguent el robot fins a la posició en la que es vol que es sitúi i guardar aquesta posició prement el botó *Set Point* i prement el símbol verd d'*Ok*.

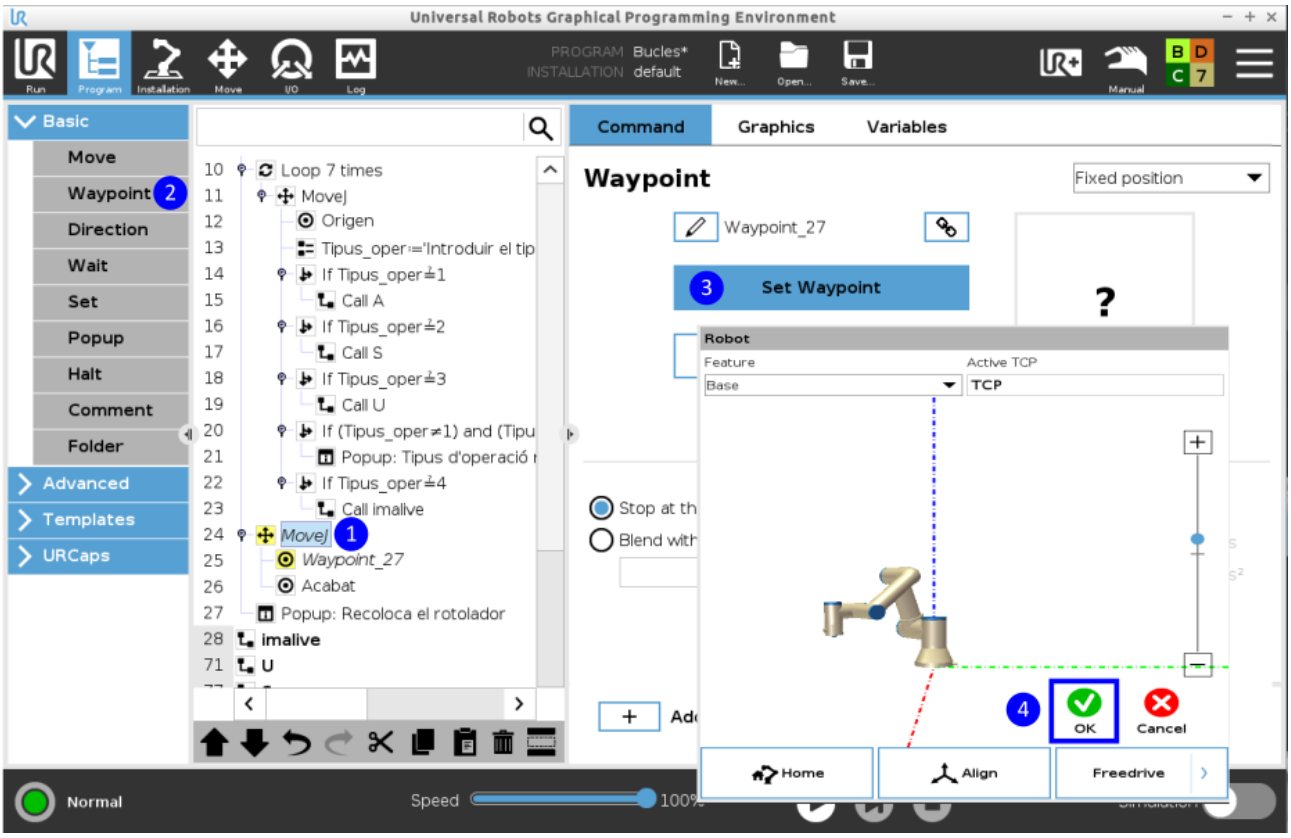


Figura 33 Procediment que cal seguir per a configurar un Waypoint.

4.3.3.3 Popup

Per a configurar el missatge que es vol que s'escrigui al usuari s'ha de seguir el procediment indicat a la figura 34. S'ha d'indicar el tipus de missatge que es vol fer servir: Missatge normal, avís o error, així com el text que es vol que contingui el missatge que rebrà l'usuari.

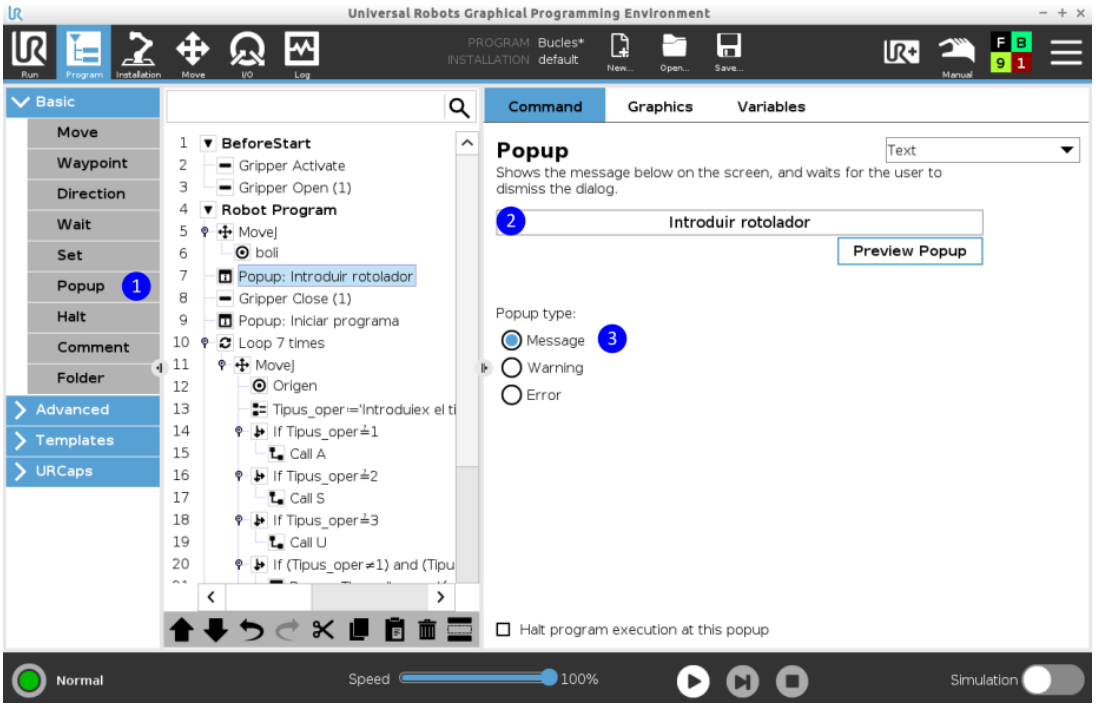


Figura 34 Configuració d'un Popup.

Per a tancar la pestanya s'ha de premer *Stop Program* si es vol aturar el programa o *Continue* si es vol continuar.

Figura 35 Missatge que apareix a l'usuari quan es fa servir la comanda Popup.

4.3.3.4 Loop

Per a programar un bucle s'ha de seleccionar la comanda *Loop* del menú de comandes avançades. Després s'ha d'indicar quin criteri s'ha de tenir per a limitar les iteracions. Per al cas que s'exposa al exemple, s'ha seleccionat l'opció d'iterar un nombre enter de vegades i s'ha introduit el nombre d'iteracions que es vol que es realitzin. Finalment, només cal afegir el codi que es vol que s'executi en bucle. Per a fer això s'ha de prémer l'icone de la comanda *Loop* dins de la branca principal i programar de forma habitual.

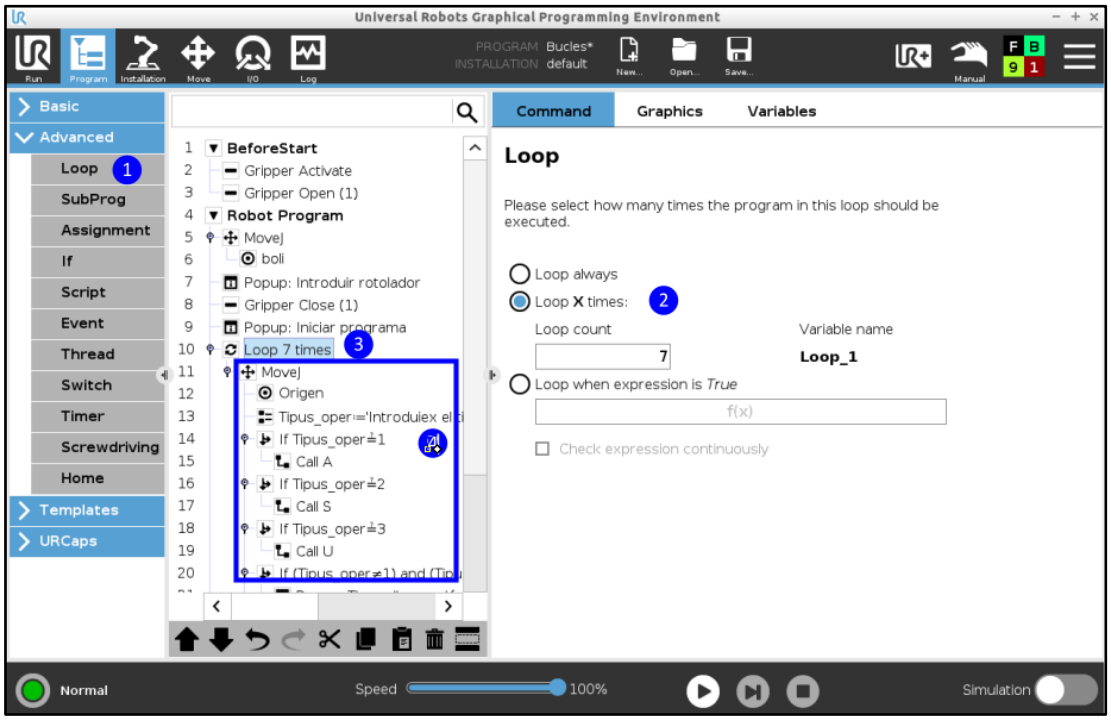


Figura 36 Configuració de la comanda Loop.

4.3.3.5 If Else

Dins del bucle principal de codi hi ha una sèrie de comandes *If else* que es fan servir per a que el robot actui d'una forma diferent en en funció del valor que prengui la variable entrada per l'operari del robot. Per a programar una d'aquestes comandes, s'ha de seleccionar la comanda dins del sumbenú de comandes avançades. Posteriorment, s'ha d'indicar quina expressió l`gica s'ha de complir per tal que s'executi el codi dins de la comanda. Finalment s'ha d'escriure quines són les operacions que ha de realitzar el robot en el cas que es compleixi aquesta expressió. Addicionalmment,es poden afergir una sèrie de *ElseIf* o *Else* en el cas que es cregui necessari fer-los servir.

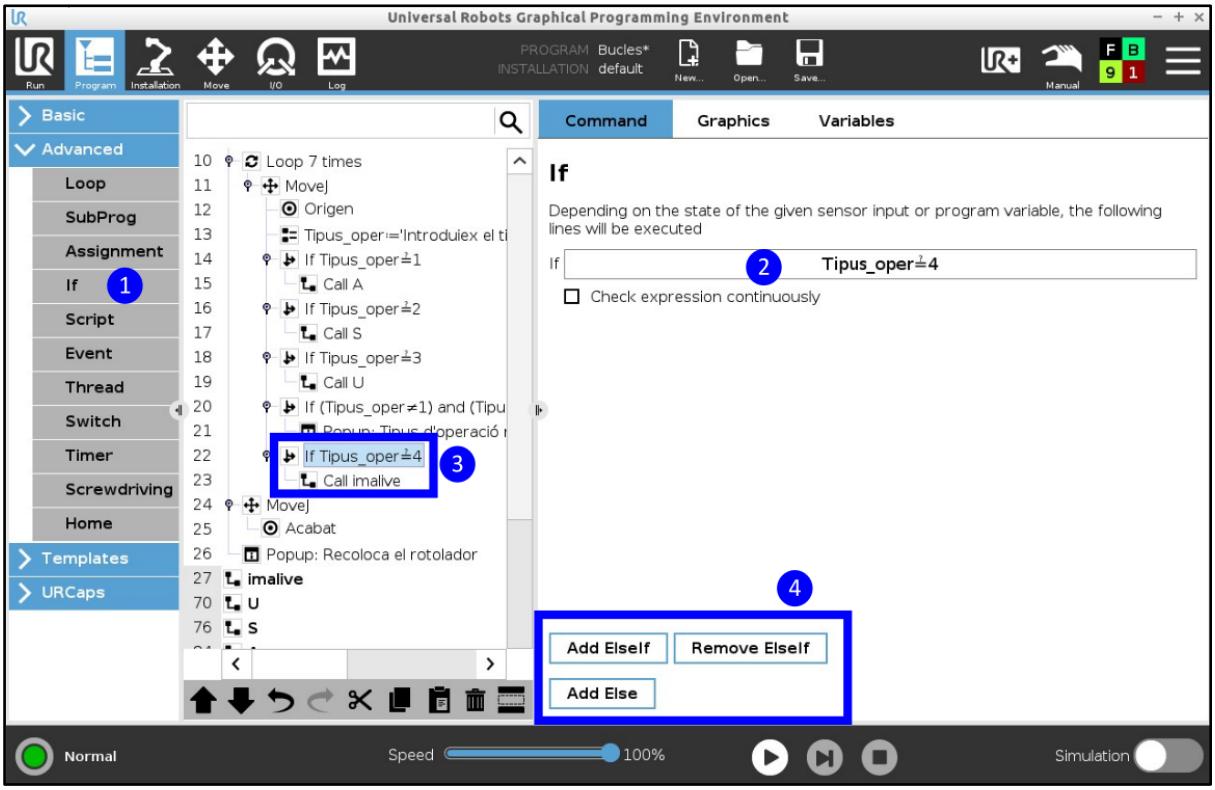


Figura 37 Configuració de la comanda If Else.

4.3.3.6 Subprogram

L'últim tipus de comanda que s'ha fet servir en la realització d'aquest exercici ha estat *Subprogram*. Aquetsa comanda crida un programa escrit en *Polyscope* per a ser executat. Dins de la finestra *Command* de la comanda *Subprogram* s'ha de seleccionar quin programa es pot cridar dintre d'una llista de programes ja carregats. En el cas que no es trobi el porgrama que es vol fer servir, s'ha de seleccionar l'opció de *Create New* i donar la ruta on es troba el programa. Els programes que s'hagin carregat despr´s es podran modificar dins de la branca en la que es trobin com si fossin comandes del programa principal.

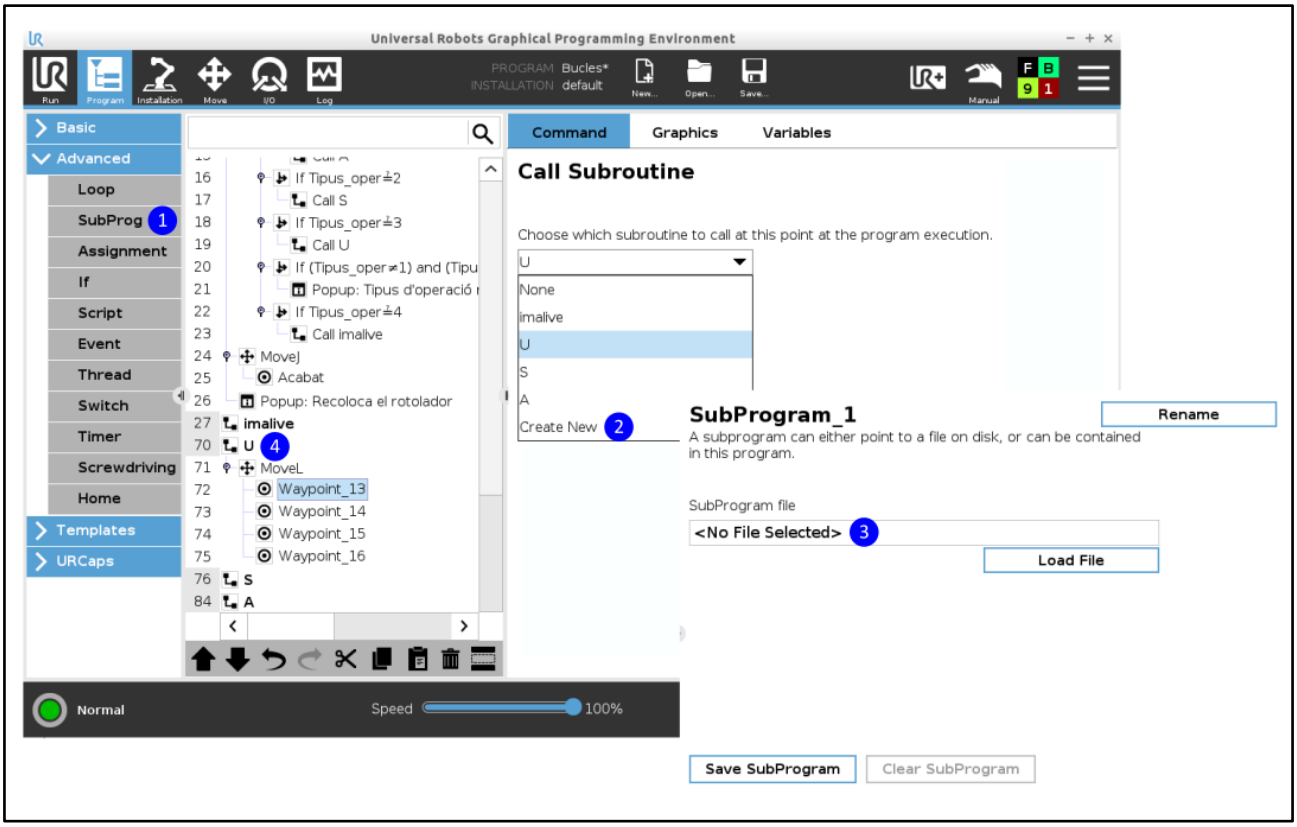


Figura 38 Procés de configuració de la comanda Subprogram.

4.4 Comunicació amb el robot

El robot UR3 té implementats diferents sistemes de comunicació amb l'exterior. Aquests sistemes es fan servir per a diferents apliccions, des d'elements de seguretat a consulta de dades o coordinació amb altres aparells i perifèrics. [https://www.universal-robots.com/articles/ur/interface-communication/overview-of-client-interfaces/] El sistema de comunicació del robot està composat per diferents canals de comunicació i una sèrie de ports IO digitals i analògics.

4.4.1 Ports IO

El conjunt del braç robòtic i la capsa de control té un total de 42 ports IO. El braç robòtic en té 6, la capsa de control té els 36 restants.

4.4.2 Interfícies de comunicació del robot

Mentre que els ports IO es fan servir per a transmetre informació senzilla, el robot té habilitat un sistema de comunicació més complex. L'ordinador de la capsa de control gestiona una sèrie d'interfícies de comunicació, amb les que es es poden rebre i enviar informació o codi URScript per a que el robot l'executi. Aquests

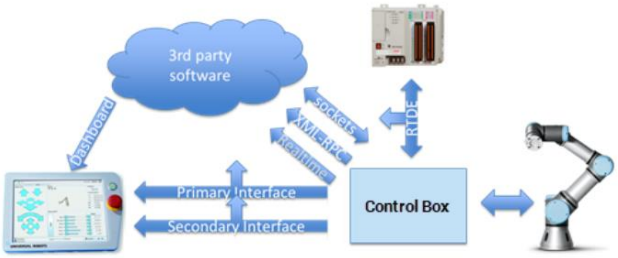


Figura 39 Diagrama de interfícies de comuncació del robot.

4.4.2.1 Arquitectura client-servidor

El robot actua com a client. Quan s'indiqui que en una intefície s'envia informació o es rep una comanda, s'està parlant del client, és a dir el software implementat al robot.

4.4.2.2 Interficies primària i secundària

Aquestes es comuniquen directament amb la *Teach Pendant*. L'interfície primària transmet informació sobre l'estat del robot, així com una sèrie de missatges adicionals. La interfície secundària només pot transmetre informació sobre l'estat del robot. Al annex s'ha afegit un document proprcionat per UR en el que s'explica el tipus d'informació que es pot transmetre per cadascuna d'aquestes intefícies. Addicionalment, aquestes dues poden rebre comandes de

URScript, de manera que es pot controlar el robot de forma remota sense la necessitat de que estigui executant un programa de *Polyscope*.[18]

4.4.2.3 Real-time (RT)

Aquestea interfície funciona de forma similar a la primària o secundària, en el sentit que pot enviar informació sobre el robot i rebre comandes de URScript. La diferència pricnipal entre les dues consisteix en que la velocitat de refrescament. En el cas de l'interfície RT, aquets ainformació es transmet a 500 Hz.

4.4.2.4 Servidor Dashboard

Es pot controlar el robot remotament enviant comandes senzilles al GUI de la *Teach Pendant* fent servir un socket amb protocol TCP/IP. Algunes de le sprincipals funcionalitats que habilita aquest servidor són controlar l'execució d'un programa (carregar, executar, pausar o executar), fixar el nivell d'accés de l'usuari o rebre informació sobre l'estat del robot.

4.4.2.5 Comunicació mitjançant sockets

El robot es pot comunicar amb equipament exterior fent servir el cable ethernet i utilitzant el protocol TPC/IP. Quan es fa servir aquets tipus de comunicació el robot actúa com a client, mentre que l'altre dispositiu actua com a servidor. Per a obrir i tancar els sockets així com per a que robot envii o rebi diferents tipus d'informació es fa servir URScript.

4.4.2.6 XML-RPC

XML-RPC (Extensive Markup Language - Remote Procedure Call) és un mètode de comunicació que fa servir XML per a transmetre informació entre programes fent servir un socket. Aquest tipus de comunicació, el robot pot cridar funcions a un serrvidor remot i que aquest li retorni informació estructurada. Això permet que el robot realitzi operacions que no estan disponibles amb URScript.

4.4.2.7 RTDE (Real-Time Data Exchange)

Aquesta interfície de comunicació permet que el controlador del robot es comuniqui amb aplicacions exteriors amb una conexió TCP/IP. En aquesta comuniació està pensada per a treballar en paral·lel amb el controlador del robot, és a dir, mentre el controlador del robot està executant codi, aquesta interfície segueix oberta. És especialment interessant per a manipular els IO del robot o rebre informació del estatus del robot, de manera que és possible representar la trajectòria del robot.

Per a aclarir, en aquesta interfície el controlador del robot actúa com a servidor, mentre que el client és un programa executat a una computadora.

Aquest procés de comunicació està dividit en dues etapes:

- Procediment de configuració (setup): En aquesta etapa el client ha d'assignar quina és la informació que es vol transmetre. Per a fer això, ha d'enviar un fitxer XML en el que s'especifica l'informació que s'ha de sicnronitzar durant el bucle de sincronització.
- Bucle de sincronització: Un cop configurada la comunicació, s'executa el bucle de sincronització. El robot enviua l'informació sol·licitada pel client en el mateix ordre amb el que el client l'ha solicitada.

4.4.2.8 Consideracions generals

Cadascuna de les interfícies de comunicació del robot té diferents formes de ser configurada i pot transmetre un tipus d'informació diferent, encara que tots fan servir els sockets de comunicació del robot. Per tal d'evitar que dos interfícies de comunicació fassin servir el mateix port de comunicació, convé tenir en compte l'assignació de ports recomanada per part d'UR. Al Annex X s'han incorporat les taules que resumeixen quins ports es poden fer servir per a cada interfíce de comunicació.

4.4.3 Exemples de comunicació

A continuació, s'exposen una sèrie de programes que s'han implementat per a

4.4.3.1 Interfície primària

En aquest exercici es realitza una tasca molt simple, el robot, que actua com a client, envia informació al servidor, el programa de *Python* que es mostra a la figura 40. Amb aquesta informació, el robot executa una operació (escull una paraula aleatòria d'una llusta) i envia la resposta al robot. Finalment el robot envia una confiració al servidor, en funció del missatge que hagi rebut.

4.4.3.1.1 Explicació del codi del client

1. Hi ha una branca de codi inicial *BeforeStart*, en la que s'obre un socket de comunicació dels disponibles al robot fent servir la comanda *socket_open()* de URScript. Aquest bucle es realitza de forma indefinida mentre no s'estableixi una connexió.
2. En el bucle principal, s'envia un string, anomenat *str*, al servidor fent servir la commanda *socket_send_string(str)* de URScript.
3. Si rep la resposta adequada, el robot contesta amb un string que provocarà que el servidor realitzi una operació.
4. El robot contesta un missatge diferent en funció del string seleccionat pel servidor per a informar del missatge que s'ha rebut

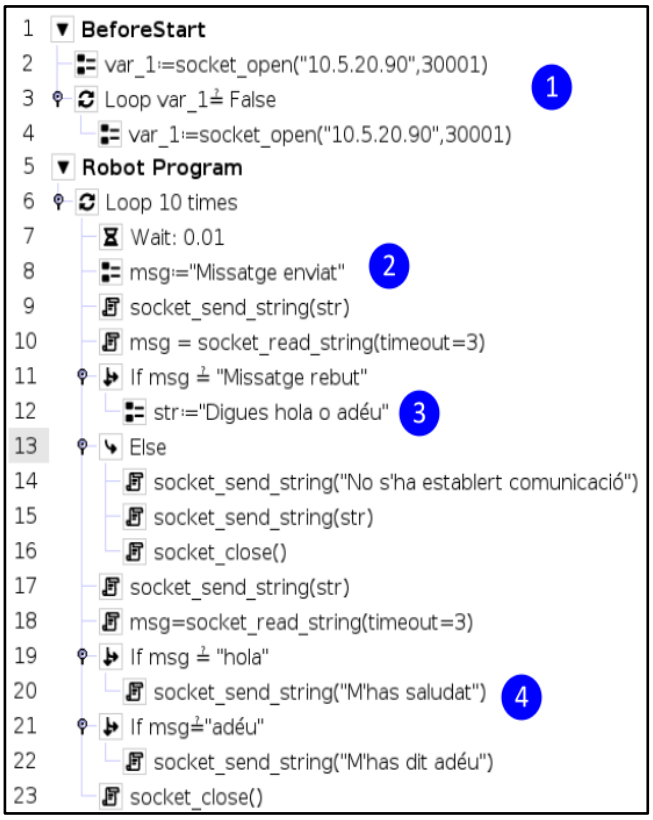


Figura 40 Codi instal·lat al client (robot) en l'exercici de comunicació.

Tot aquest procediment es repeteix 10 cops. Després es tanca el socket de comunicació.

4.4.3.1.2 Explicació del codi del servidor

1. S'indiquen l'adressa IP del robot i el port que es farà servir per a establir la comunicació. El valor del port és 30001 ja que es fa servir la interfície primària de comunicació.
2. Es configura el socket que es farà servir amb els valors indicats a l'apartat 1. S'espera a la resposta del client i s'estableix connexió.
3. El servidor rep un missatge, en el cas que sigui l'esperat envia una resposta al client.
4. Rep una resposta, la processa i realitza una operació: selecciona un element d'una llista. Posteriorment rep un missatge per a confirmar si el missatge rebut és el correcte.

```
import socket
import time
import random as rnd

1 HOST= "10.5.20.90" #IP del PC: Que actuca com a servidor.
  PORT=30002  #Número del port, ha de ser el mateix al que enviarà info el Client.

print("Iniciem el programa"+"\n")
count = 0

while (count < 10):

    #Inicialització del socket de comunicació:
2   s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind((HOST, PORT)) # Es vincula el port al socket: Configuració
    s.listen(5) # S'espera a la connexió del client estigui disponible.
    c, addr = s.accept() # S'estableix la connexió amb el client.

    msg = c.recv(1024)
    if str(msg) == "b'Missatge enviat'":
3       print("Ur: "+str(msg) +"\n")
        count = count +1
        print("Nombre d'iteració: "+str(count))
        #time.sleep(0.5)
        print("\n")
        #time.sleep(0.5)
        cmd="Missatge rebut"
        c.send(cmd.encode("utf-8"));
        print("PC:Missatge rebut\n")
        msg2=c.recv(1024)
        print("Ur: "+str(msg2)+"\n")
        if str(msg2)=="b'Digues hola o adéu'":
4           print("DHA \n")
            cmd=rnd.choice(["hola","adéu"])
        print("PC: "+str(cmd)+"\n")
        c.send(cmd.encode("utf-8"))
        msg2=c.recv(1024)
        print("Ur: "+str(msg2)+"\n")

print(count)
c.close()
s.close()

print("Final del programa\n")
```

Figura 41 Codi instal·lat al servidor en l'exercici de comunciació

Els missatges que envia el servidor s'han de codificar, tal i com estableix el protocol de comunicació de l'interfície primària, en fomrat utf-8. Per a fer això es fa servir la comanda *cmd.encode*.

4.4.3.2 Interfície RTDE

Per a comunicar-se fent servir la interfície RTDE s'ha fet servir la llibreria ur_rtde de *Python*. Aquesta llibreria permet fer servir l'interfície RTDE de manera senzilla, ja que s'encarrega de realitzar totes les operacions del procés de sincronització.

4.4.3.2.1 La llibreria UR_RTDE

Aquesta llibreria habilitat tres noves interfícies de comunicació:

- RTDE Receive: Es l'intefície encarregada de gestionar l'enviament de dades del robot al programa.
- RTDE Control: Serveix per a poder moure el robot i executar funcions qeu s'encarreguen del control del robot.

- RTDE IO: Es fa servir per a gestionar els IO digitals i analògics i per a controlar la velocitat del *speed slider* del robot, que s'encarrega de regular la velocitat amb la que s'executen les comandes del robot.

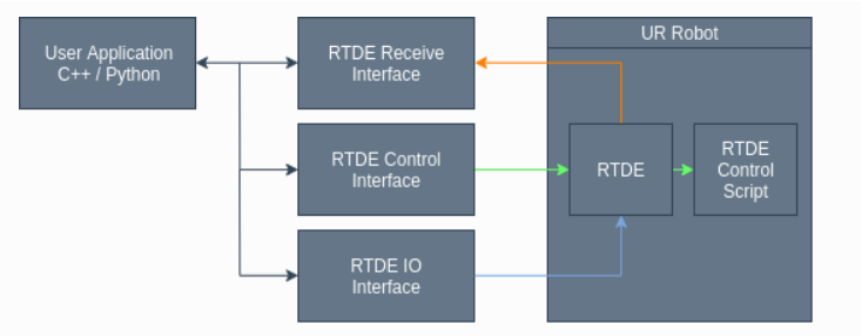


Figura 42 Esquema del funcionament de la llibreria.

Com es pot observar a la figura 42, per a que el robot funcioni, aquest ha d'estar executant un script ed control. Un cop s'està executant aquest script no es pot modificar, s'ha de tornar a configurar des de zero tot el prtocol de comunciació. Això implica que, un cop s'executi el codi amb aquesta llibreria, no es podra modificar el moviment del robot, tot i que les accions de seguretat com la pausa i execució del programa e spodrán seguir realitzant fent servir la *Teach Pendant*. Les altres interfícies seguiran en funcionament, de manera que es podrà rebre informació en temps real de l'estat del robot i controlar els IO del robot, que 'shan situat en una interfície per separat expressament.

4.4.3.2.2 Exemple de llibreria UR_RTDE

S'ha realitzat un programa exemple en el que es fa servir aquesta llibreria per a rebre dades en temps real del robot.. Aquest exemple consisteix en la recollida de dades per a realitzar un experiment, amb el que es preté estudiar quines són les forces i moments que rep el TCP (Tool Central Point) quan es mou perpendicularment contra el terra a una certa velocitat.

En primer lloc s'ha executat des de la GUI Polyscope un programa que mou l robot en linea recta en una posició perpendicular al terra, de manera que l'eina entra en contacte amb una superfície i, subsequentment, rep una serie d'esforços.

Un cop executat aquest programa, s'ha procedit a executar el codi que es presenta a l'apartat 4.1.3.2.3.

El resum del codi és el següent:

1. Es crea una classe que permet accedir a l'informació transmesa mitjançant RTDE.
2. Declaració de variables.
3. Bucle principal. En aquest bucle es llegeixen els valors de forces, moments i posicions enregistrats durant l'experiment. Per a poder fer això es criden els mètodes

getActualTCPForce() i getActualTCPPose(). Que retornen una llista de 6 elements cadascuna, corresponent als esforços que pateix el robot i coordenades de la TCP.

4. S'imposa que el període de mostreig sigui de 2ms. Aquesta limitació es va imposar per a evitar que es realitzessin lectures falses. Segons la documentació aprotada per UR, la velocitat de transmissió de dades del protocol RTDE és de fins a 500 Hz [19], per a aquesta raó s'ha imposat que com a mínim el temps entre medicions hagi de ser de 2 ms.
5. Es guarda l'informacio dins d'un fitxer d'excel.
6. Es mostren els gràfics de les dades que s'han mostrejat.

4.3.3.2.3 Codi de Ur_rtde

```
import rtde_receive
import time
import openpyxl as pyxl
import matplotlib.pyplot as mplot

#Variables
rtde_r=rtde_receive.RTDEReceiveInterface("10.5.20.84") #Ip robot
index=0
num=0
temps=0
Forces=[]
Fx=[]
llFx=[]
Fy=[]
llFy=[]
Fz=[]
llFz=[]
Mx=[]
llMx=[]
My=[]
llMy=[]
Mz=[]
llMz=[]
Row=[]
X=[]
llX=[]
Y=[]
llY=[]
Z=[]
llZ=[]
ang1=[]
llang1=[]
ang2=[]
llang2=[]
ang3=[]
llang3=[]
Posicions=[]
inici_experiment=time.time()
```

1

2

```
while time.time()-inici_experiment < 5:
    temps_mesura=time.time()*1000
    index=index+1
    element=[index]+[rtde_r.getActualTCPForce()]+[time.time()-inici_experiment]
    Forces.append([index]+[rtde_r.getActualTCPForce()]+[time.time()-inici_experiment])
    Posicions.append([index]+[rtde_r.getActualTCPPose()]+[time.time()-inici_experiment])
    if(time.time()*1000-temps_mesura<2):
        time.sleep((2-(time.time()*1000-temps_mesura))/1000)
wb =pyxl.Workbook()
ws=wb.active
```

3

4

```
for i in range(len(Forces)):
    num=Forces[i][0]
    Fx=Forces[i][1][0]
    llFx.append(Forces[i][1][0])
    Fy=Forces[i][1][0]
    llFy.append(Forces[i][1][1])
    Fz=Forces[i][1][0]
    llFz.append(Forces[i][1][2])
    Mx=Forces[i][1][0]
    llMx.append(Forces[i][1][3])
    My=Forces[i][1][0]
    llMy.append(Forces[i][1][4])
    Mz=Forces[i][1][0]
    llMz.append(Forces[i][1][5])
    temps=Forces[i][2]
    X=Posicions[i][1][0]
    Y=Posicions[i][1][1]
    Z=Posicions[i][1][2]
    ang1=Posicions[i][1][3]
    ang2=Posicions[i][1][4]
    ang3=Posicions[i][1][5]
    llX.append(X)
    llY.append(Y)
    llZ.append(Z)
    llang1.append(ang1)
    llang2.append(ang2)
    llang3.append(ang3)
    Row=[num,Fx,Fy,Fz,Mx,My,Mz,X,Y,Z,ang1,ang2,ang3,temps]
    #print(Row)
    ws.append(Row)
data=str(time.time())
wb.save(data+'experiment.xlsx')
```

5

```
#Ploteado

mplot.plot(llFx)
mplot.xlabel('Mesurement')
mplot.ylabel('Fx')
mplot.show()

mplot.plot(llFy)
mplot.xlabel('Mesurement')
mplot.ylabel('Fy')
mplot.show()

mplot.plot(llFz)
mplot.xlabel('Mesurement')
mplot.ylabel('Fz')
mplot.show()

mplot.plot(llZ)
mplot.xlabel('Mesurement')
mplot.ylabel('Z')
mplot.show()

mplot.plot(llX)
mplot.xlabel('Mesurement')
mplot.ylabel('X')
mplot.show()

mplot.plot(llY)
mplot.xlabel('Mesurement')
mplot.ylabel('Y')
mplot.show()
```

6

5. Control del robot amb UR_RTDE

5.1 Definició del sistema a controlar

Es vol controlar la força que realitza el robot en l'eix Z (Fz) quan el robot estigui realitzant un moviment en perpendicular al pla Z=0 en una postura determinada, que s'obtindrà de forma experimental al laboratori. S'ha agafat com a sistema de referència per a definir aquests plans i direccions el sistema de referència definit per la *Feature Base* del robot. El sistema de referència del que es parla es pot consultar a la figura 43.

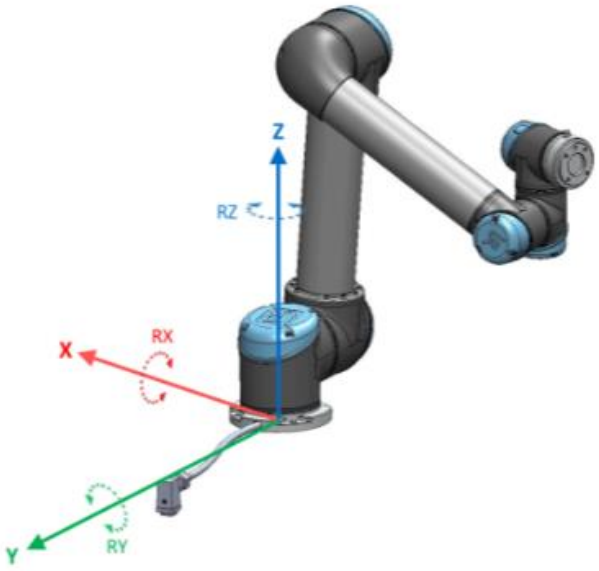


Figura 43 Sistema de referència que es fa servir per a implementar el controlador.

El tipus de controlador que s'implementarà serà un PID. S'ha escollit aquest tipus de controlador, ja que és un control eficaç que s'utilitza habitualment en moltes aplicacions de l'enginyeria. Inicialment es va plantetjar fer servir altres tipus de control, però al tractar-se d'un controlador amb interès quasi exclusivament acadèmic, que només controla una única variable, s'han descartat al ser excessivament complexes.

5.2 Controladors de força existents

La primera etapa del disseny va consistir en estudiar altres controladors de força similars als que s'ha implementat en aquest TFG. Es va trobar un treball [Proyecto Ferriz] en que es tracta de controlar la mateixa variable, F_z, mitjançant la velocitat . L'esquema de blocs del controlador en llaç tancat és el que es pot veure a la figura 4.1.

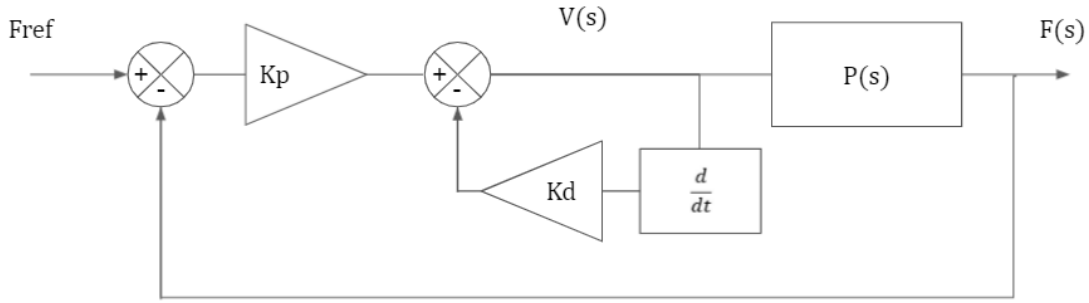


Figura 44 Esquema de blocs del controlador mitjançant velocitat.

En el treball mencionat anteriorment, es van trobar uns coeficients de forma empírica, sense tenir en consideració la funció de transferència, que donaven una resposta impulsional al sistema amb una entrada de 10 N estable.

Els valors dels coeficients obtinguts en el treball anterior van són $K_p = 2 \cdot 10^{-5}$ i $K_d = 0$.

Els resultats de la planta segons els valors del controlador anterior són els que es mostren a la figura 45.

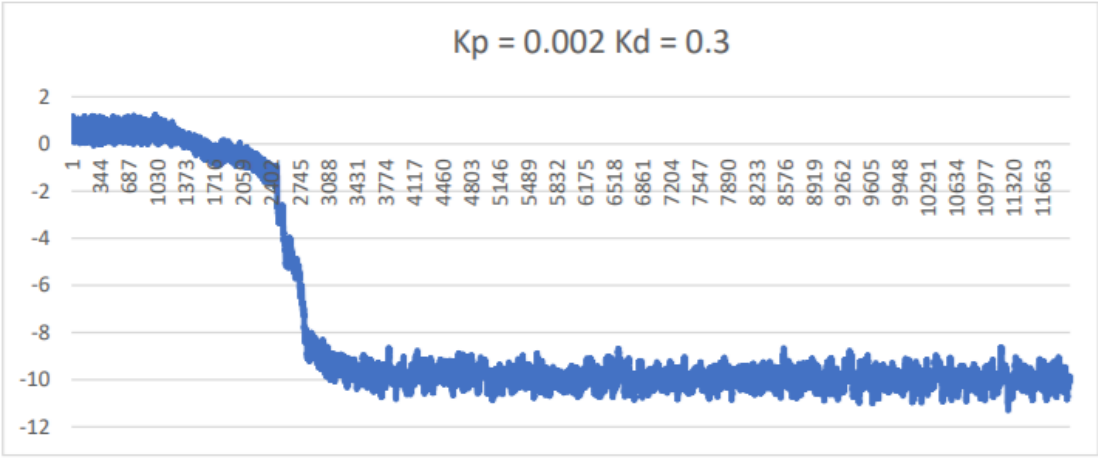


Figura 45Resposta del sistema per a una entrada impulsional de graó -10 N. Gràfic extret de [20].

Com es pot comprovar, la resposta és la d'un sistema estable,encara que les mesures mostren molt soroll.

5.3 Model del controlador propi que es vol implementar

Es va plantejar llavors un nou model de controlador. L'intenció que es van tenir va ser la de controlar el valor de F_z fent servir la posició del robot, en comptes de fer servir la velocitat. Això es va fer així per tractar d'evitar les oscil·lacions que es podrien esperar d'un control de velocitat, ja que, en el cas de El sistema de blocs del sistema que es vol controlar és el que es mostra a la figura 46.

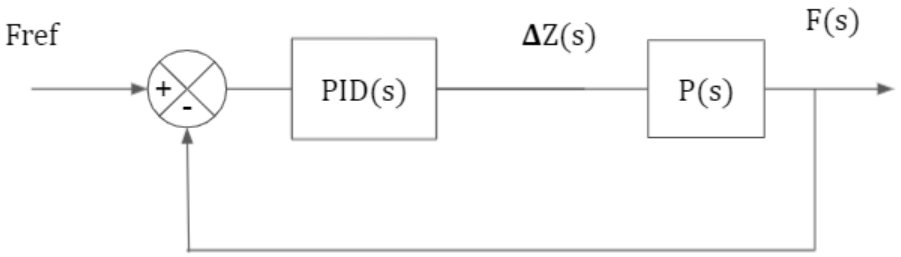


Figura 46 Esquema de blocs del controlador implementat.

A partir de les lectures del error en la força, s'indicarà al robot si ha d'augmentar o disminuir la

5.4 Obtenció de la planta

5.4.1 Plantejament teòric

Per tal d'obtenir la planta del sistema es va plantejar el següent. Guiant-se amb els valors de K_p i K_d que es van obtenir en el treball anteriorment mencionat, es va procedir a bucar de forma empírica un sistema amb un controlador proporcional tal que la resposta impulsional del sistema per a un graó de 10 N fos estable. Posteriorment, amb aquest sistema estable, es procediria a calcular quin és el valor de la funció de transferència del sistema a partir dels resultats de la resposta impulsional. Amb aquests resultats de la funció de trasnferència, tenint en compte que PID(s) = K_p, es va procedir a trobar la planta del sistema P(s).

5.4.2 Experiment

De forma empírica es va comprovar que el sistema amb $K_p = 2 \cdot 10^{-6}$ era un sistema estable. Realitzant lectures durant un experiment de 8 segons es va obtenir la següent resposta impulsional, amb un valor de $T_{mostreig}$ = 0,015152 s.

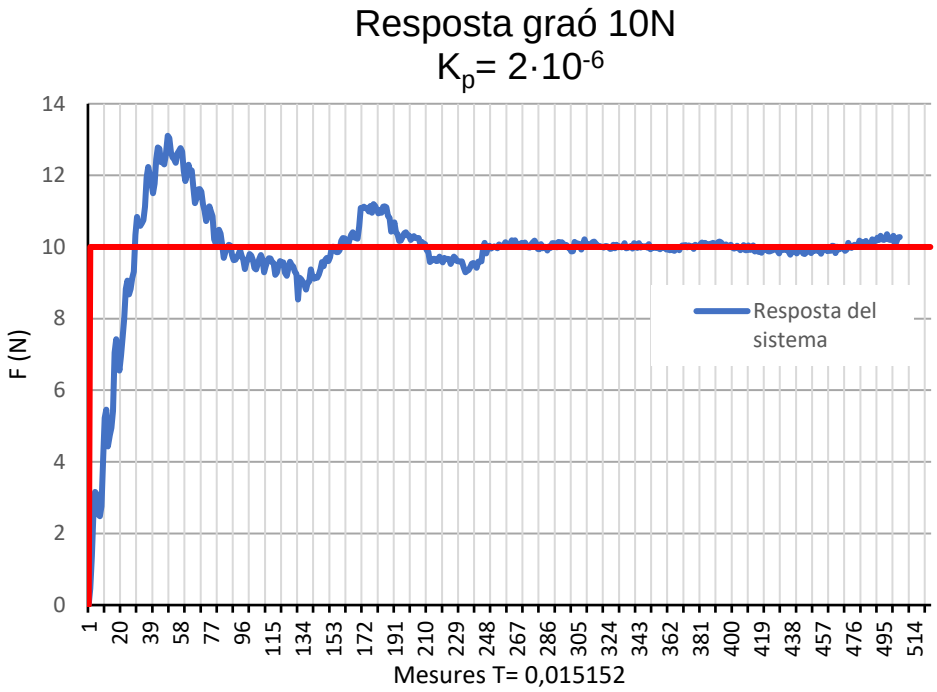


Figura 47 Resposta del sistema estudiat amb una Kp =2·10-6 amb una entrada graó d'amplitud 10 N

5.4.3 Obtenció de la funció de transferència del sistema

5.4.3.1 Obtenció de la resposta temporal

A partir dels resultats ateriors, es va procedir a aproximar el valor de les mesures una funció de temps continu amb la que calcular el valor de la funció de trasnferència de la planta. Donat l'aspecte de les mesures, es va suposar que la resposta es tracta d'un sistema de segon ordre.

Aquestes són la funció de trasnferència d'un sistema de segon ordre genérica:

$$(1)\ G(s) = \frac{K_1\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$$

Realitzant els càlculs necessaris [pág 34 llibre dinámica de sistemes] s'arriba a que la resposta davant d'una entrada graó $x(t) = K_2u(t)$ és la següent:

$$(2)\ y(t) = K_1K_2\left(1+\frac{e^{-\xi\omega_n}}{\sqrt{1-\xi^2}}\sin\left(\omega_n\sqrt{1-\xi^2}t-\alpha\right)\right)u(t)\ on\ \alpha = \text{acos}\,(-\xi)$$

Per tal d'obtenir una bona aproximació és va plantetjar i resoldre un problema d'optimització: Es vol minimitzar el valor de la suma de les diferències elevades al quadrat entre la funció (2) i els resultats experimentals $\bar{y}(k)$, variant el valor de les variables $\xi\ i\ \omega_n$.

$$Obj: \min f(k,T) = \left(\overline{y}(kT) - y(kT)\right)^2$$

$$var: \xi \; i \; \omega_n$$

Notar que s'ha fer servir kT en comtes de la variable temps, ja que els valors mesurats no són una funció temporal, sino una serie discreta.

Fent servir el solver d'Excel, es va obtenir una solució per a aquest problema:

$$\xi = 0{,}407045 \; i \; \omega_n = 4{,}717118$$

El valor del guany canònic s'ha imposat, $K_1 = 1$, ja que, com es comprova a la figura 47, la resposta del sistema davant un graó de 10 N té un valor en estat estacionari de 10 N.

Com es pot comprovar a la figura 48, la funció aproximada té un comportamnet molt similar al de la funció. La regió entre les mesures 120 i 239 és la que té un comportamnet més distant, però es pot comprovar que la funció calculada aproxima degudament el comportament de la funció al primer sobrepic i al estat estacionari.

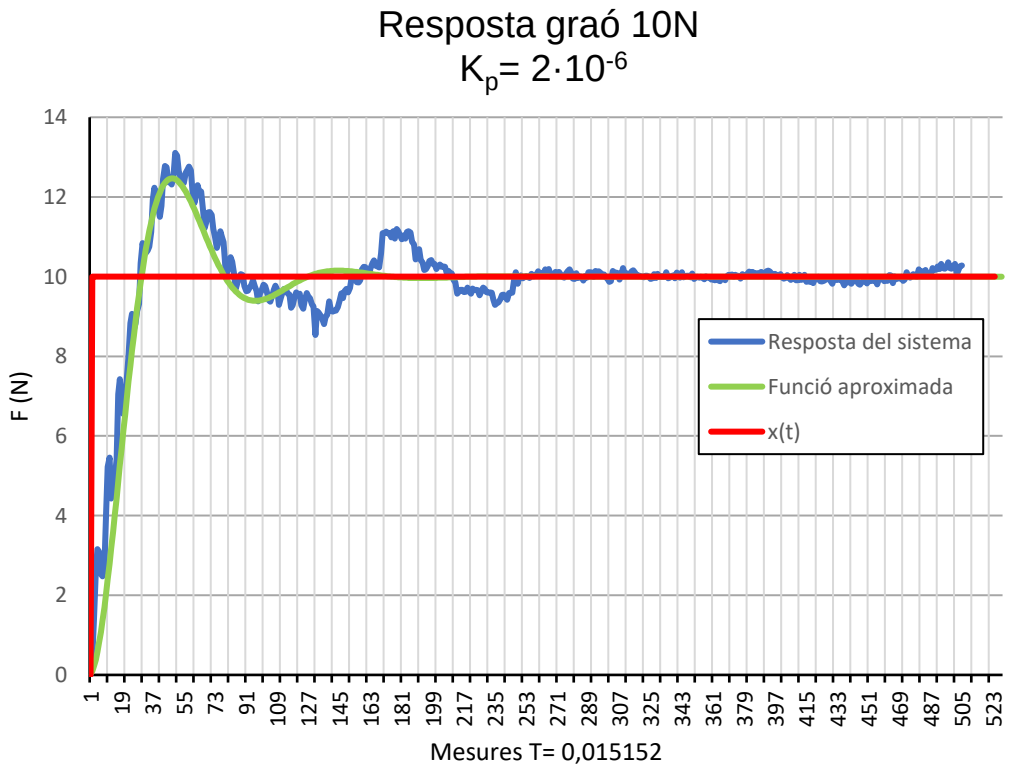


Figura 48 Resposta temporal de la funció aproximada i les dades experimentals.

Aquesta divergència entre els resultats experimntals deu ser causada per errors experimentals, així com pel fet s'assumir que la planta es comporta com un sistema de segon ordre sense zeros.

Tot i aquesta diferència entre resultats experimentals i la funció aproximada, s'ha cosniderat que l'aproximació és acceptable, i s'ha procedit a realitzar els càlculs necessaris per al disseny del controlador

5.4.3.2 Funció de tranferència del sistema i la planta

A partir d'aquí es va obtenir la funció de trasnferència del sistema, substituïnt pels valors de ξ, ω_n i K_1 obtinguts:

$$G(s) = \frac{K_1\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} = \frac{22{,}2510}{s^2 + 3{,}8402s + 22{,}2510}$$

A partir d'aquesta funció de transferència, es va procedir a calcular el valor de la trasmitància de la planta. A partir del esquema de blocs del controlador, i suposant que $PID(s) = K_p = 2 \cdot 10^{-6}$, s'obté el següent valor de la transmitància de la planta:

$$P(s) = \frac{G(s)}{K_p - K_p G(s)} = \frac{1{,}11255 \cdot 10^7}{s(s + 3{,}8402)}$$

5.4.4 Caracterització del sistema

A partir de la funció de trasnferència del sistema es pot caracteritzar la resposta indicial del sistema:

- Temps de sobrepuig: $t_p = \frac{\pi}{\omega_n\sqrt{1-\xi^2}} = 0{,}7291\ s$
- Sobrepuig absolut: $S_a = A \cdot e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} = 2{,}466\ N$
- Sobrepuig relaitu: $S_r = e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} \cdot 100 = 24{,}66\ \%$
- Període d'oscil·lació: $T = \frac{2\pi}{\omega_n\sqrt{1-\xi^2}} = 1{,}4583\ Hz$
- Temps d'establiment (2%) : $t_s = \frac{4}{\omega_n\xi} = 2{,}0832\ s$
- Pols del sistema: $s_{1,2} = -\omega_n\xi \pm \omega_n\sqrt{1-\xi^2} \cdot i = -1{,}92 \pm 4{,}3087\ i$

5.5 Càlcul del controlador

5.5.1 Tipus de controlador emprat

El tipus de controlador que es vol fer servir és un controlador PID, donada la seva senzillesa a l'hora d'implementar-se. Estudiant el valor de la funció de trasnferènica del sistema, es pot veure que és de tipus 1.

$$P(s) = \frac{1{,}11255 \cdot 10^7}{s(s + 3{,}8402)}$$

Això vol dir que l'error al estat estacionari per a una entrada graó és 0, tal i com s'ha comprovat a l'hora de realitzar els experiments. Donat a que la precissió del sistema està garantida, s'ha decidit implementar un controlador PD per tal d'assolir major rapidesa en la resposta del sistema:

$$PD(s) = K_p + K_d s$$

5.5.2 Obtenció dels valors dels coeficients

Els requeriments que s'han fixat per al controlador són els següents:

- El sistema ha de ser estable.
- Ha de tenir un temps d'establiment un 20% inferior al del primer sistema analitzat.
- Es vol que el sobrepuig relatiu de la resposta sigui d'un 30%.

A partir d'aquests requeriments, es va aplicar el mètode d'ubicació de pols per a definir els valors dels coefcients K_p i K_d del controlador.

5.5.2.1 Mètode de l'ubicació de pols

Un cop trobada la funció de trasnferència del sistema es va procedir a fixar els paràmetres següents.

- Temps d'establiment (2%): $t_{s2} = 0{,}8 \cdot t_{s1} = 2{,}0832 \cdot 0{,}8 = 1{,}6666\ s$
- Sobrepuig relatiu: $S_{r2} = e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} \cdot 100 = 30\ \%$

De manera que es van obtenir els següents pols per al sistema:

$$s_{1,2} = -2{,}40025\ \pm 6{,}2909\ i$$

Pertant, l'equació carcaterística del sistema:

(1) $D_{G2}(s) = s^2 + 4{,}80025s + 45.336$

Aqui es presenten els càlculs realitzats per a trobar els coeficients del controlador

(2) $PID(s) = \frac{N_c(s)}{D_c(s)} = K_p + K_d s \rightarrow N_c(s) = K_p + K_d s \ i \ D_c(s) = 1$

(3) $P(s) = \frac{N_p(s)}{D_p(s)} = \frac{1{,}11255 \cdot 10^7}{s(s+3{,}8402)} \rightarrow N_p(s) = 1{,}11255 \cdot 10^7 \ i \ D_p(s) = s(s + 3{,}8402)$

(4) $G_2(s) = \frac{N_c(s)N_p(s)}{D_c(s)D_p(s)+N_c(s)N_p(s)} \rightarrow D_{G2}(s) = D_c(s)D_p(s) + N_c(s)N_p(s)$

Imposant que els pols del sistema són $s_{1,2} = -2{,}40025 \pm 6{,}2909 \, i$ s'obté l'equaicó (5), d'on saïllen els valors de $K_p \ i \ K_d$:

(5) $D_{G2}(s) = s^2 + 4{,}80025s + 45.336 = s^2 + (3.8402 + K_d \cdot 1.1126 \cdot 10^7) + K_p \cdot 1.1126 \cdot 10^7$

Resolguent aquesta equació s'obté $K_p = 4{,}07496 \cdot 10^{-6} \ i \ K_d = 8{,}629 \cdot 10^{-8}$.

El controlador dissenyat tindrà pertant, la següent funció de transferència:

$$PD(s) = 4{,}07496 \cdot 10^{-6} + 8{,}629 \cdot 10^{-8} s$$

5.6 Resultats del controlador implementat

Finalment es va implementar el controlador PD que s'ha dissenyat durant la realització d'aquest TFG. Un controlador PD en temps continu es comporta de manera que en un instant de temps t, el valor de la sortida és la suma d'una resposta proporcional al error més una repsposta proporcional a la derivada del error en aquell mateix instant: $u(t) = K_p \cdot e(t) + K_d \cdot \frac{de}{dt}(t)$. Al estar implementant el controlador amb una computadora, no es pot tenir el valor de $\frac{de}{dt}(t)$, només es pot aproximar a partir de les dades que s'han obtingut amb els sensors. L'aproximació que s'ha implementat fa servir les dades actuals i de l'instant de temps anterior, així com el temps trasncorregut entre els dos instants per a calcular la velocitat a un instant de tems qualsevol, tal i com s'inidica a l'equació (1).

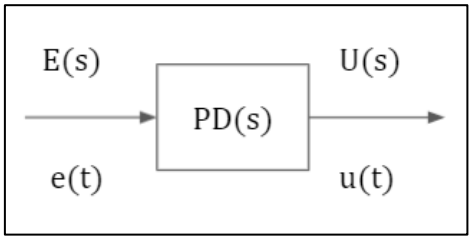


Figura 49 Representació de la trasnmitància d'un PID.

(1) $u(t) = K_p \cdot e(t) + K_d \cdot \frac{de}{dt}(t) \approx K_p \cdot e(kT) + K_d \cdot \frac{e(kT) - e(kT-1)}{T}$

Al Annex 3 s'incorpora una captura de pantalla del codi que es va fer servir per a implementar aquest controlador.

5.6.3. Comparació entre sistema controlat i sense control

De l'anàlisi visual de la comparació entre els dos sistemes, es pot extreure que, efectivament, el controlador de força PID implementat sembla tenir un comportament més ràpid que el del sistema sense el controlador.

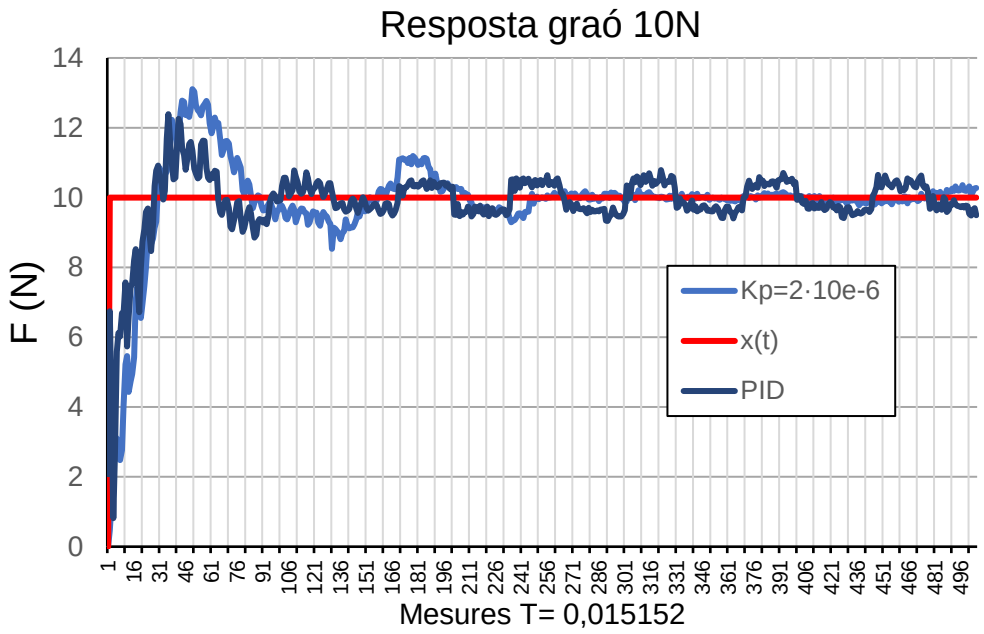


Figura 50 Representació de la trasnmitància d'un PID.

El sistema amb el controlador esmorteeix més rapid l'oscil·lació. Aproximadament al voltant de la mostra 136, el comportament del sistema amb controlador es repeteix durant la duració restant del experiment. És notable veure que el sistema presenta una certa oscil·lació, aparentment periòdica, que dura al llarg de tot l'experiment que no apareixia en el sistema sense controlador. Aquest comportament inusual es podria explicar a causa dels zeros del sistema. Al calcular els pols del sistema controlat, en el cas d'existir zeros per a la transmitància $G(s)$ els valors dels pols del nou sistema serien totalment diferents als que s'havia calculat inicialment.

PLANIFICACIÓ TEMPORAL

Taula 1 Taula d'activitats

Valor	Activitat	Valor	Activitat	Valor	Activitat
T1	Planificació del TFG	**T5**	Redacció de l'entrega parcial	**T9**	Realització d'experiments del control de força
T2	Recerca sobre robòtica	**T6**	Aprenentatge sobre comunicaicó del robot	**T10**	Dissseny del controlador
T3	Aprenentatge Polyscope	**T7**	Realització de pràctiques i exercicis de comunicació	**T11**	Implementació del controlador.
T4	Realització de pràctiques i exercicis de Polyscope	**T8**	Aprenentarge UR_RTDE	**T12**	Redactat de la memòria del TFG

Taula 2 Diagrama de Gantt de la planificació del projecte.

	Febrer	Març	Abril	Maig	Juny	Juliol	Agost	Set
T1								
T2								
T3								
T4								
T5								
T6								
T7								
T8								
T9								
T10								
T11								
T12								

ETSEIB

ESTUDI ECONÒMIC: PRESSUPOST

Per a calcular el cost del TFG s'han valorat els principals costos materials, les hores laborals i el preu de l'electricitat.

Taula 3 Distribució dels costos del projecte.

Salaris				
Tipus de treball	**Hores realitzades**	**Sou**		**Cost total**
Enginyer	175 h	45 €/h		7875 €
Programador	75 h	25 €/h		1875
Oficina	125	12 €/h		1500 €
Amortització de material				
Material	**Cost d'amortització**	**Duració del projecte**		**Cost total**
Robot	2252,08 €/any	1 any		2252,08 €
Perifèrics	700 €/any	1 any		700 €
Cost de producció				
Tipus	**Cost de l'electricitat**	**Hores**	**Potència**	**Cost total**
Electricitat	0,253 €/kWh	140 h	240 W	8,5 €
Cost total				14.210,58 €

IMPACTE SOCIAL I AMBIENTAL

Impacte social del projecte

Aquest TFG tracta sobre robòtica. Aquest camp de l'indústria ha tingut un cert punt de polèmica al etsar destinat a dissenyar i fabricar maquinària que està destinada a competir amb els treballadors. Encara que s'hagi demostrar que a llarg termini l'automatització que s'ha dut a terme fins a hores d'ara ha generat més llocs de feina dels que ha tret (el % de població activa [wolrd developement report 2019] creix en comparació amb la de segle 20), a curt termini pot destruir sectors laborals sencers.

Donat que l'automatització i la robotització de tots els sectors laborals no és un fenòmen que sigui evitable i que afecta sobretot a la població que està menys formada (figura 51). És necessari dotar d'eines a les persones que tinguin menys estudis per tal de facilitar la seva mobilitat a altres llocs de treball o, com a mínim, dotar-les de formació assequible per tal de que siguin més competitus, en un mercat laboral que serà cada cop més complicat.

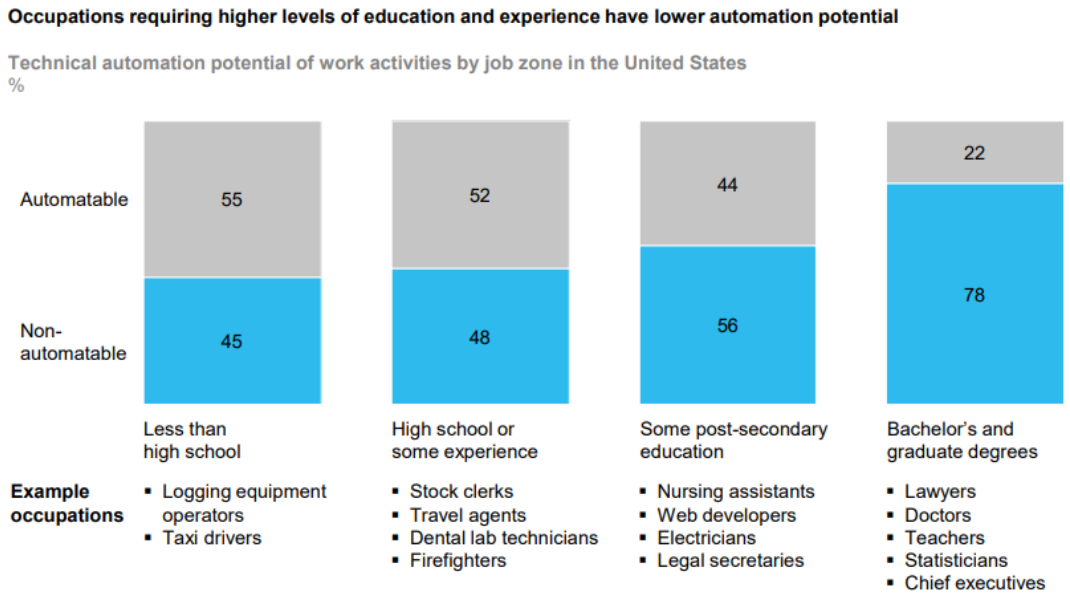


Figura 51 Potencial d'automatització de les activitats laborals segons el nivell d'estudis mínims requerits per a realitzar-les als EUA.

Aquest TFG està enfocat a fer una molt petita contribució a aquesta necessitat, en forma d'un curs de robòtica i material docent, que en acabar aquest projecte es farà disponible al públic, després de passar per un procés de reedició i correcció.

Impacte ambiental

L'impacte ambiental d'aquest projevcte no ha estat especialment elevat. Les úniques emissions que es poden tenir en compte són les del consum elèctri i les del trasnport que s'ha fet servir per a accedir al laboratoir.

Tot i això sí que s'ha de tenir en ment el potencial problema que pot presentar l'industria de la robòtica per al medi ambient. Es tracta d'una indústria que necessariament contamina el medi ambient, ja que per a la fabricació dels components no es fa servir material reciclat, i que es troba en un important creixement. És necessari que lees empreses líders en el secrtor prenguin mesures quan avans millor per tal de que en el sector es fomentin les bones pràctiques envers al medi ambient. Un exemple d'això és el programa de préstecs de robots de UR[21].

Finalment, comentar que un dels principals impactes ambientals que genera el robot són les emissions de CO_2 provinents del consum d'electricitat del robot. Si es vol minimitzar l'impacte ambiental generat per a aquests dispositius. Si es vol que l'indústria robòtica minimitzi les seves emissions és necessari garantir que l'electricitat que consumeixien provingui de fonts renovables o de baixa o nula emissió de gassos d'efecte hivernàcle.

CONCLUSIONS

Es considera que el resultat d'aquest TFG, en general, compleix amb els objectius que s'havien proposat al inici del projecte. S'ha redactat una sèrie de material didàctic que afegeix contingut sobre el funcionament del robot. És especialment interessant el contingut pràctic. Ja existeixen manuals sobre *Polyscope*, però no hi ha tant de contingut d'exercicis pràctics o exemples de codi que estiguin explicats detalladament.

Sobre el controlador, el resultat obtingut no és exactament el desitjat. L'oscil·lació que presenta la resposta del sistema és un fenòmen inesperat que s'hauria d'estudiar amb més detall. S'ha pogut implementar i s'ha demostrat que funciona amb una aplicació pràctica. Com a crítica, hagués estat bé afegir més controladors per a estudiar-los i comparar-los. També mencinar que hagués estat millor implementar un controlador digital, en comptes de fer l'aporximació de considerar que el controlador treballava en temps continu.

A nivell personal, la realització del projecte m'ha ajudat a aprendre sobre conceptes que no havia tingut l'oportunitat d'aprofundir durant el transcurs de la carrera, com ho són el funcionament de la comunicació entre diferents ordinadors o el funcionament d'un robot.

Passes següents

Per una banda, cal estudiar si el curs que s'ha realitzat necessita alguna modificació per posteriorment posar-se a l'abast del públic. Si es troba que el material que s'ha redactat és d'interès, es contemplarà l'opció de crear un blog en el que penjar aquets material.

En el que es refereix al controlador, seria interessant provar de realitzar més controladors, canviant les especificacions, per a comparar-los entre ells. També seria interessant pantetjar el problema del disseny del contrlador des del punt de vista de la dinàmica de sistemes discrets.

BIBLIOGRAFIA

Referències bibliogràfiques

[1] Industrial Robots Market Size & Share | Global Report [2028] (fortunebusinessinsights.com) (accedit febrer 2021).

[2] "The contribution of the automobile industry to technology and value creation" https://www.es.kearney.com/automotive/article/?/a/the-contribution-of-the-automobile-industry-to-technology-and-value-creation (accedirt febrer 2021)

[3] "The benefits of automation in pharmaceutical manufacturing" https://www.theengineer.co.uk/automating-pharmaceuticals/ (accedit el març de 2021)

[4] What Are Collaborative Robots, Cobots | A3 Robotics Collaborative Robots (automate.org) (accedit el març de 2021).

[5] "Collaborative Robots vs Industrial Robots: key differences - RoboAds" (accedit el febrer de 2021)

[6] "Cobots, the new collaborative robots - Iberdrola"(accedit el febrer de 2021).

[7] "Aplicaciones de la Robótica | Universal Robots" (universal-robots.com) (accedit abril 2021)

[8] "How Human-Robot Teamwork Will Upend Manufacturing"https://www.technologyreview.com/2014/09/16/171369/how-human-robot-teamwork-will-upend-manufacturing/ (accedit l'agos de 2021)

[9] la Fp Navarra Aumenta La Tasa De Empleabilidad En Robótica Colaborativa Tras La Introducción De Cobots De Universal Robots (universal-robots.com) (accedit el maig de 2021)

[10] "Our History "[https://www.universal-robots.com/about-universal-robots/our-history/] (accedit el febrer de 2021.)

[11] "Universal Robots revenue 2019**"**https://statzon.com/insights/universal-robots?gclid=CjwKCAjwp_GJBhBmEiwALWBQk-JQnOeZ-N1gfFjddg5SYhklc7Hp8Rd6Vl0mP-ZKDdr11R_-P0TwDhoC-EkQAvD_BwE (accedit el juny de 2021).

[12]" LAUNCH OF URCAPS: THE NEW PLATFORM FOR UR ACCESSORIES AND PERIPHERALS" https://www.universal-robots.com/about-universal-robots/news-centre/launch-of-urcaps-the-new-platform-for-ur-accessories-and-peripherals/ (accedit el maig de 2021).

[13] "UR3 workspace"https://s3-eu-west-1.amazonaws.com/ur-support-site/16336/100403.PDF (accedit el maig de 2021).

[14] UR Academy | Free Online Collaborative Robot Training (universal-robots.com) (accedit el abril de 2021)

[15] Zacobria - Universal robots - customer support training hints tips - 6 axis robot (accedit el maig de 2021)

[16] Universal Robots RTDE C++ Interface — ur_rtde 1.4.5 documentation (sdurobotics.gitlab.io) (accedit el maig de 2021)

[17] ROS for Industrial Robots Course | The Construct (theconstructsim.com) (accdeit el maig de 2021.

[18] Primary/Secondary Client Interface guide https://s3-eu-west-1.amazonaws.com/ur-support-site/16496/ClientInterfaces_Primary.pdf

[19] REMOTE CONTROL VIA TCP/IPhttps://www.universal-robots.com/articles/ur/interface-communication/remote-control-via-tcpip/ (accdit el maig de 2021)

[20] Porjecte ferriz, UPV

[21] https://www.universal-robots.com/about-universal-robots/news-centre/universal-robots-launches-financial-services-leasing-program-in-collaboration-with-dll/

ANNEXOS:

Annex 1: Codi fet servir durant l'experiment.

```
import rtde_control
import rtde_receive
import rtde_io
import time
import openpyxl as pyxl
import matplotlib.pyplot as mplot

#Variables
rtde_r=rtde_receive.RTDEReceiveInterface("10.5.20.84") #Ip robot
rtde_io=rtde_io.RTDEIOInterface("10.5.20.84") #Ip robot
index=0
num=0
temps=0
Forces=[]
Fx=[]
llFx=[]
Fy=[]
llFy=[]
Fz=[]
llFz=[]
Mx=[]
llMx=[]
My=[]
llMy=[]
Mz=[]
llMz=[]
Row=[]
X=[]
llX=[]
Y=[]
llY=[]
Z=[]
llZ=[]
ang1=[]
llang1=[]
ang2=[]
llang2=[]
ang3=[]
llang3=[]
Posicions=[]
inici_experiment=time.time()
```

```
while time.time()-inici_experiment < 5:
    temps_mesura=time.time()*1000
    index=index+1
    element=[index]+[rtde_r.getActualTCPForce()]+[time.time()-inici_experiment]
    Forces.append([index]+[rtde_r.getActualTCPForce()]+[time.time()-inici_experiment])
    Posicions.append([index]+[rtde_r.getActualTCPPose()]+[time.time()-inici_experiment])
    if(time.time()*1000-temps_mesura<2):
        time.sleep((2-(time.time()*1000-temps_mesura))/1000)
wb =pyxl.Workbook()
ws=wb.active
print(len(Forces))
print(index)
for i in range(len(Forces)):
    num=Forces[i][0]
    Fx=Forces[i][1][0]
    llFx.append(Forces[i][1][0])
    Fy=Forces[i][1][0]
    llFy.append(Forces[i][1][1])
    Fz=Forces[i][1][0]
    llFz.append(Forces[i][1][2])
    Mx=Forces[i][1][0]
    llMx.append(Forces[i][1][3])
    My=Forces[i][1][0]
    llMy.append(Forces[i][1][4])
    Mz=Forces[i][1][0]
    llMz.append(Forces[i][1][5])
    temps=Forces[i][2]
    X=Posicions[i][1][0]
    Y=Posicions[i][1][1]
    Z=Posicions[i][1][2]
    ang1=Posicions[i][1][3]
    ang2=Posicions[i][1][4]
    ang3=Posicions[i][1][5]
    llX.append(X)
    llY.append(Y)
    llZ.append(Z)
    llang1.append(ang1)
    llang2.append(ang2)
    llang3.append(ang3)
    Row=[num,Fx,Fy,Fz,Mx,My,Mz,X,Y,Z,ang1,ang2,ang3,temps]
    #print(Row)
    ws.append(Row)
data=str(time.time())
wb.save(data+'experiment.xlsx')
```

```
#Ploteado

mplot.plot(llFx)
mplot.xlabel('Mesurement')
mplot.ylabel('X force')
mplot.show()

mplot.plot(llFy)
mplot.xlabel('Mesurement')
mplot.ylabel('Y force')
mplot.show()

mplot.plot(llFz)
mplot.xlabel('Mesurement')
mplot.ylabel('Z force')
mplot.show()

mplot.plot(llZ)
mplot.xlabel('Mesurement')
mplot.ylabel('Z pos')
mplot.show()

mplot.plot(llX)
mplot.xlabel('Mesurement')
mplot.ylabel('X pos')
mplot.show()

mplot.plot(llY)
mplot.xlabel('Mesurement')
mplot.ylabel('Y pos')
mplot.show()
```

Annex 2: Codi control de velocitat.

```
import rtde_control
import rtde_receive
import rtde_io
import time
import openpyxl as pyxl
import matplotlib.pyplot as mplot
import sys
#Variables
rtde_r=rtde_receive.RTDEReceiveInterface("10.5.20.84") #Ip robot
rtde_io=rtde_io.RTDEIOInterface("10.5.20.84") #Ip robot
rtde_c= rtde_control.RTDEControlInterface("10.5.20.84")#Ip robot
experiment_duration=8
target_pos=rtde_r.getActualTCPPose()
F=0
Ft=10
Values=[]
Z1p=250/1000 #mm
Z1n=120/1000 #mm
dt=1.0/500 #2ms
kp=0.0017
kd=0.15
index=0
llF=[]
llFt=[]
llFi=[]
llZ=[]
llV=[]
llVz=[]
llcanvi=[]
lltemps=[]
lldif=[]
lla=[]
llerror=[]
ex_duration = 5 #s
Values=[]
```

```
#Experiment starts

# Initial movement untill contact

rtde_c.moveJ_IK([0.23345,-0.31572,0.16818,0.0,3.14,0.0],0.5,True)
rtde_c.moveUntilContact([0.0,0.0,-0.07,0.0,0.0,0.0],[0.0,0.0,-1,0.0,0.0,0.0],True)
rtde_c.speedStop()

#Control implementation
temps_inici=time.time()
speed=kp*(rtde_r.getActualTCPForce()[2]-Ft)
while time.time()-temps_inici < ex_duration:
    index=index+1
    t_i_iter=time.time()
    speed_z=speed
    F=rtde_r.getActualTCPForce()[2]
    Z=rtde_r.getActualTCPPose()[2]
    speed=kp*(F-Ft)
    speed=speed-kd*speed_z
    rtde_c.speedL([0.0,0.0,speed,0.0,0.0,0.0],10,1/1000)
    Values.append([index,speed,speed_z,F,F-Ft,Z])
    aux=time.time()-t_i_iter
    if dt>aux:
        time.sleep(dt-aux)
#Stopping routine
rtde_c.speedStop()
rtde_c.stopScript()

    #Savig data:
    wb =pyxl.Workbook()
    ws=wb.active
    print(len(Values))
    print(index)
    for i in range(len(Values)):
        num=Values[i][0]
        V=Values[i][1]
        llV.append(V)
        Vz=Values[i][2]
        llVz.append(Vz)
        F=Values[i][3]
        llF.append(F)
        error=Values[i][4]
        llerror.append(error)
        Z=Values[i][5]
        llZ.append(Z)
        Row=[num,V,Vz,F,error]
        ws.append(Row)
    data=str(time.time())
    wb.save(data+'kp'+str(kp)+'kd'+str(kd)+'Ferriz_controller.xlsx')

    #Plotting
    mplot.plot(llF)
    mplot.xlabel('Mesurement')
    mplot.ylabel('Force')
    mplot.show()

    mplot.plot(llV)
    mplot.xlabel('Mesurement')
    mplot.ylabel('Speed')
    mplot.show()

    mplot.plot(llerror)
    mplot.xlabel('Mesurement')
    mplot.ylabel('Error')
    mplot.show()

    mplot.plot(llerror)
    mplot.xlabel('Mesurement')
    mplot.ylabel('Z')
    mplot.show()

    sys.exit()
```

Annex 3: Codi control de posició

```
import rtde_control
import rtde_receive
import rtde_io
import time
import openpyxl as pyxl
import matplotlib.pyplot as mplot
import sys
#Variables
rtde_r=rtde_receive.RTDEReceiveInterface("10.5.20.84") #Ip robot
rtde_io=rtde_io.RTDEIOInterface("10.5.20.84") #Ip robot
rtde_c= rtde_control.RTDEControlInterface("10.5.20.84")#Ip robot
experiment_duration=8
target_pos=rtde_r.getActualTCPPose()
F=0
Ft=10
Values=[]
Z1p=250/1000 #mm
Z1n=120/1000 #mm
dt=1.0/500 #2ms
kp=0.0017
kd=0.15
index=0
llF=[]
llFt=[]
llFi=[]
llZ=[]
llV=[]
llVz=[]
llcanvi=[]
lltemps=[]
lldif=[]
lla=[]
llerror=[]
ex_duration = 5 #s
Values=[]
```

```
#Experiment starts

# Initial movement untill contact

rtde_c.moveJ_IK([0.23345,-0.31572,0.16818,0.0,3.14,0.0],0.5,True)
rtde_c.moveUntilContact([0.0,0.0,-0.07,0.0,0.0,0.0],[0.0,0.0,-1,0.0,0.0,0.0],True)
rtde_c.speedStop()

#Control implementation
temps_inici=time.time()
pos=kp*(rtde_r.getActualTCPForce()[2]-Ft)
while time.time()-temps_inici < ex_duration:
    index=index+1
    t_i_iter=time.time()
    pos_z=pos
    F=rtde_r.getActualTCPForce()[2]
    Z=rtde_r.getActualTCPPose()[2]
    pos=kp*(F-Ft)
    pos=pos-kd*pos_z
    q=rtde_r.getactualTCPPose()
    q[2]=q[2]+pos
    rtde_c.moveL(q,0.1,10,True)
    Values.append([index,pos,pos_z,F,F-Ft,Z])
    aux=time.time()-t_i_iter
    if dt>aux:
        time.sleep(dt-aux)
#Stopping routine
rtde_c.speedStop()
rtde_c.stopScript()
```

```
#Savig data:
wb =pyxl.Workbook()
ws=wb.active
print(len(Values))
print(index)
for i in range(len(Values)):
    num=Values[i][0]
    V=Values[i][1]
    llV.append(V)
    Vz=Values[i][2]
    llVz.append(Vz)
    F=Values[i][3]
    llF.append(F)
    error=Values[i][4]
    llerror.append(error)
    Z=Values[i][5]
    llZ.append(Z)
    Row=[num,V,Vz,F,error]
    ws.append(Row)
data=str(time.time())
wb.save(data+'kp'+str(kp)+'kd'+str(kd)+'Ferriz_controller.xlsx')

#Plotting
mplot.plot(llF)
mplot.xlabel('Mesurement')
mplot.ylabel('Force')
mplot.show()

mplot.plot(llV)
mplot.xlabel('Mesurement')
mplot.ylabel('Speed')
mplot.show()

mplot.plot(llerror)
mplot.xlabel('Mesurement')
mplot.ylabel('Error')
mplot.show()

mplot.plot(llerror)
mplot.xlabel('Mesurement')
mplot.ylabel('Z')
mplot.show()

sys.exit()
```

Annex 4: Codi PID

```
import matplotlib.pyplot as mplot
import sys
#Variables
rtde_r=rtde_receive.RTDEReceiveInterface("10.5.20.84") #Ip robot
rtde_io=rtde_io.RTDEIOInterface("10.5.20.84") #Ip robot
rtde_c= rtde_control.RTDEControlInterface("10.5.20.84")#Ip robot
experiment_duration=8
target_pos=rtde_r.getActualTCPPose()
F=0
Ft=10
Values=[]
Z1p=250/1000 #mm
Z1n=120/1000 #mm
dt=1.0/500 #2ms
kp=4.07496e-6
kd=8.629e-8
index=0
llF=[]
llFt=[]
llFi=[]
llZ=[]
llY=[]
llX=[]
llV=[]
llVz=[]
llcanvi=[]
lltemps=[]
lldif=[]
lla=[]
llerror=[]
ex_duration = 10 #s
Values=[]

#Experiment starts

# Initial movement untill contact

rtde_c.moveJ_IK([0.23345,-0.31572,0.16818,0.0,3.14,0.0],0.5,True)
rtde_c.moveUntilContact([0.0,0.0,-0.07,0.0,0.0,0.0],[0.0,0.0,-1,0.0,0.0,0.0],True)
rtde_c.speedStop()
q=rtde_r.getActualTCPPose()
```

```
#Control implementation
temps_inici=time.time()
t0=temps_inici
pos=kp*(rtde_r.getActualTCPForce()[2]-Ft)
while time.time()-temps_inici < ex_duration:
    index=index+1
    t_i_iter=time.time()
    rtde_c.stopL(10)
    pos_z=pos
    F=rtde_r.getActualTCPForce()[2]
    pos=kp*(F-Ft)
    dt=t_i_iter-t0
    pos=pos+kd*(pos/kp - pos_z/kp)/(dt)
    t0=t_i_iter
    q[2]=q[2]+pos
    Z=q[2]
    Y=q[1]
    X=q[0]
    rtde_c.moveL(q,0.1,10,True)
    Values.append([index,pos,pos_z,F,F-Ft,X,Y,Z])
    aux=time.time()-t_i_iter
    if dt>aux:
        time.sleep((dt-aux))
#Stopping routine
rtde_c.speedStop()
rtde_c.stopScript()
```

```
#Savig data:
wb =pyxl.Workbook()
ws=wb.active
print(len(Values))
print(index)
for i in range(len(Values)):
    num=Values[i][0]
    V=Values[i][1]
    llV.append(V)
    Vz=Values[i][2]
    llVz.append(Vz)
    F=Values[i][3]
    llF.append(F)
    error=Values[i][4]
    llerror.append(error)
    Z=Values[i][7]
    llZ.append(Z)
    Y=Values[i][6]
    llY.append(Y)
    X=Values[i][5]
    llX.append(X)
    Row=[num,V,Vz,F,error,X,Y,Z]
    ws.append(Row)
data=str(time.time())
wb.save(data+'kp'+str(kp)+'kd'+str(kd)+'propi_controller.xlsx')
```

```
#Plotting
mplot.plot(llF)
mplot.xlabel('Mesurement')
mplot.ylabel('Force')
mplot.show()


mplot.plot(llerror)
mplot.xlabel('Mesurement')
mplot.ylabel('Error')
mplot.show()

mplot.plot(llZ)
mplot.xlabel('Mesurement')
mplot.ylabel('Z')
mplot.show()

mplot.plot(llX)
mplot.xlabel('Mesurement')
mplot.ylabel('X')
mplot.show()

mplot.plot(llY)
mplot.xlabel('Mesurement')
mplot.ylabel('Y')
mplot.show()

sys.exit()
```