UR Universal-Robots

Els robots UR3(3kg), UR5(5kg), UR10(10Kg) de l'empresa Universal-Robots son programats amb un terminal tàctil i una interfície gràfica (GUI-java) que permet operar el robot, posar en marxa els programes o crear nous programes. Aquest mateix terminal pot fer-se anar també amb un ordinador o terminal GNU/linux. Per la qual cosa interactua molt be amb aquest sistema, que li dona molta potencia al poder-se programar amb llenguatges d'alt nivell com python, java...i la gran quantita d'informació al basar-se en open source(el software del robot funciona en linux, però no es open source, no ens donen el codi font:-(

A més te un controlador, electrònica de potència per al motors, font d'alimentació, càlculs de cinemàtica directa-inversa...i també 8 entrades/sortides, 2 en l'extremitat de la eina(tool), i un modul ethernet i MODBUS per comunicació amb altres dispositius de l'instal·lació com automats, i altres oridinadors diversos(SCADA)

universalrobotssetup.mp4
https://www.youtube.com/watch?v=18VJhTsU96w

El que els fa novedosos es que son robots col·laboratius("cobot"). Son robots(braços o braç amb mobilitat)dissenyats per treballar conjuntament amb les persones i no tancats a una gàbia per motius de seguretat. La seguretat es mitjançant el programa que pot limitantar la força, la potencia, per la qual cosa el robot contínuament mesura la força que està fent i que nosaltres hem limitat amb el software. Les normatives, com sol ser habitual, van endarrerides, encara no hi ha cap normativa clara sobre la robòtica col·laborativa.

Coordenades X,Y,Z RX, RY, RZ

Es tracten de robots de 6 eixos(motors). Per definir un objecte a l'espai necessitem 6 coordenades. Les X, Y, Z indiquen la posició respecte els eixos, però tot objecte te una **orientació** al espai. Per exemple un avió a mes de les posicions X,Y i Z te una orientació no es mateix que vagi cap avant o de costat, de fet deixaria de volar. Altre exemple, un robot fent una soldadura a més de la **posició** X,Y,Z del cordó la eina en aquest cas l'elèctrode, fil, ... ha de tenir una inclinació i una direcció adient. Aquestes ultimes es representen amb angles de rotació a cascun dels eixos RX, RY, RZ.

Les X,Y,Z son conegudes com **posició**, i els angles RX, RY, RZ com **orientació**.

x_y_and_z_position._zacobria_universal-robots_community_.g.pdf

Ens centrarem en X,Y,Z per entendre millor el funcionament del moviment i de les coordenades.

Els robots UR son de 6 eixos, per fer el que sembla un senzill moviment a la posició X,Y,Z necessita fer uns moviment coordinats del diferents motors. Al ser braços articulats l'eix X no el mou un sol motor com passa una fresadora CNC, o una impresora 3D, on si que mou cada eix un motor diferent.

Posem el robot a X=0, Y=430 mm Z=400

Programació

Una vegada ja hem entès els moviments XYZ i les orientacions de les eines. Els programes es basen en fer una seqüencia de moviments. Per fer programes tinguem veries possibilitats, que normalment i son a tots el robots i que son mes o menys complicades i potentes, normalment quan més complicades mes potents:

1. Programació bàsica amb el software del mateix robot.

programacio_basica.pdf

2. Programació amb scripts i llenguatges de programació del mateix robot. Amb aquests scrpits es poden definir variables, controlar el flux del programa(programa estructurat tipus c), dividir les tasques en subprogrames. També son més còmodes al poder utilitzar editors de text en compte d'escriure les comandes al software del robot. Podem interactuar amb el host(ordinador on està el script) i generar fitxers per control de producció...interactuar amb altres PLC, altres robots... amb el MODBUS....

programcio_script_client-server_example.pdf

,

programacio_modbus_registers_input_and_output_handling_via_port_502.pdf

.

3. Programació amb llenguatges com python o c++, que es comuniquen amb el robot mitjançant una API(llibreria, driver). L'avantatge d'aquest sistema es que pot integra-se amb eines ja desenvolupades com OpenCV per la visió artificial, TensorFlow per a intel·ligència artificial etc, i totes aquestes està reunides al ROS(Robotic Operating System) que està convertint-se amb un estadard al mon de la robotica. Tots els fabricats estan fent les (API, llibreries, drives) per poder fer-los servir amb el ROS. El ROS es un Linux-Ubuntu on ja venen instal·lades totes les eines necessàries per fer simulacions, visió artificial, intel·ligència artificial, comunicacions entre robots, entre persones i robots(reconeixement de veu), IOT(Internet de les coses....)

ur_ros_setup_tutorial.pdf

Mes o menys els diferents tipus de programació corresponent a les tres generacions de robots, 1a generació que repeteix una seqüencia, 2a generació que interactua amb el exterior, 3a generació amb intel·ligència artificial. Si volem programar un robot de tercera generació cal aprendre a programar el llenguatges típics d'alt nivell. (C, C++, JAVA, PYTHON ...).

Així que si volem afegir Visió artificial, Inteligencia artificial, IOT... caldrà fer una programació de tipus 3.

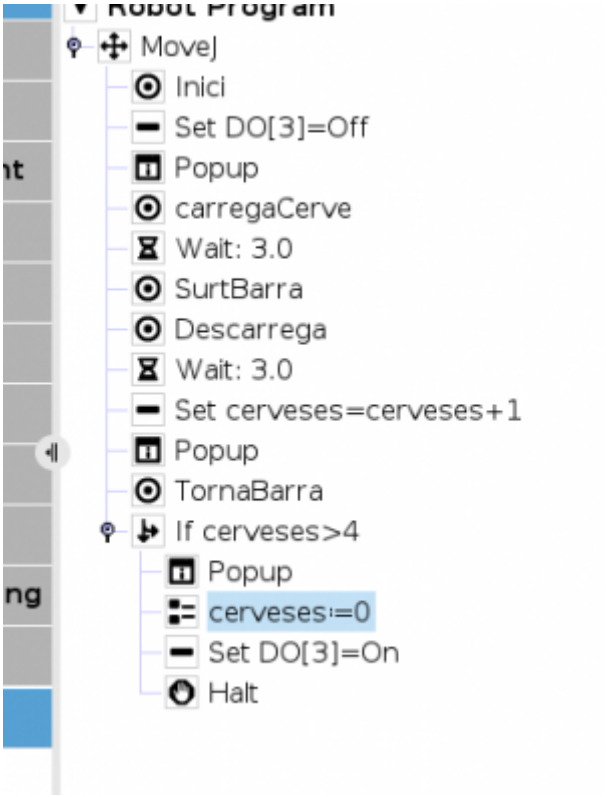
software lliure vs software privat

La filosofia dels fabricats tant d'autòmats com de robots ha sigut, i malauradament hi segueix, es d'oferir-te totes les possibilitats de programació amb un software propietari, on per exemple si vols

afegir-li visió artificial tinguis que comprar el seus accessoris i els seu software que complementa el robot. Si es veritat que simplifica molt la programació, però te limitacions econòmiques(son sistemes molt cars) i no son ni ampliables ni modificables,no podem afegir funcionalitats, no tinguem el codi font. Semblaria que aquest sistema seria millor que un basat en software lliure, però normalment les llibreries de software lliure tenen tanta o més qualitat que les que utilitzen els software privats, que de vegades amb una falta d'ètica impressionant es basen amb elles. Per exemple llibreries com OpenCV per la visó artificials i/ tensorFlow(intel·ligència artificial Google), reconeixement de veu... son difícilment superables per una empresa de robòtica. De tota manera això està canviant perquè les empreses de robots estan oferint el drives per a ROS, amb la qual cosa tenim un nexe entre el robot propietari i el software lliure.

robots.ros.org_table_manipulators.pdf

Exemple programa amb el UR



Per fer aquest programa he seguit els següents passos:

1. Waypoints

- Inici
- CarregaCervesa
- SurtBarra
- Descarrega
- TornaBarra

2. Wait

Després de CarregaCervesa i Descarrega un wait de 3s

3. Popup

Després de inici “Vols una cervesa polsa continuar”

4. Variables Instal·lació + SET

Defineixo una variable que es cerveses que s'incrementarà en cada cicle(**SET increment installation variable**), després d'incrementar el nombre de cerveses posarem un popup amb el contador actual

5. if

A adveced tinguem el if de la programació típica on comprovarem que has fet mes de 4 cerveses, en cas de fer mes de 4 cerveses el programa s'atura **HALT** i cal donar-li al play des de la consala perquè es torni a posar en marxa.

6. Assignment

Posarem el nombre de cerveses novament a 0.

7 SET DO[3]=ON

Per avisar que ja tinguem massa cerveses activarem la sortida digital(Digital Output) 3, que aturarem quan posem el comptador de cerveses a 0.

programaur_basic.mp4

Per últim guardem el programa, l'extensió de l'arxiu es .urp. Hauria de guardar-se a la carpeta del escriptori que hi ha al costat del programa.

Altre exemple

Per fer altre exemple, podríem pintar una superfície, per la qual cosa el moviment hauria de ser lineal(moveL) en compte de moveJ