ADVANCED ROBOTICS PROJECT

Members: Andy

Flores Carlos

Campos David

Fernandez Jhon

Jacobo Fabio

Jeri

INDEX

1. Problem

Warehouses today are grappling with the challenge of where to store their inventory, due to the high demand of today's market. Basically, there are three issues.

The first is inventory control, since people alone cannot control the large amount of products coming in and out, and know where exactly they are in the warehouse.

Secondly, because of the large warehouses that exist today, personnel must be required to place these products in awkward locations. This increases the risk of accidents for people.

Finally, it requires a large number of personnel to manage storage, which in the long run translates into high costs.

The structure of the warehouses varies, depending on the industry. This project will focus on warehouses in the medical, retail and manufacturing industries, at the sales stage. Because some handling of the products is required, since they are stored in small packages.

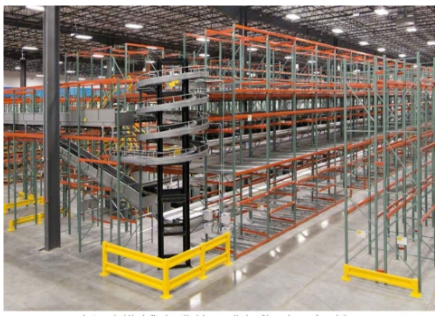


Figure 1. Warehouse for the medical industry [1].

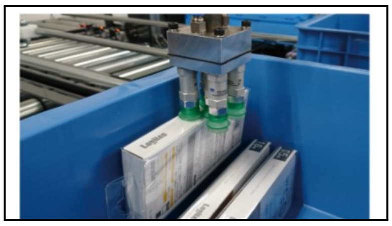


Figure 2. Manipulation of an object inside a shelf [3].

2. Background

2.1. Automated Storage and Retrieval Systems (ASRS)

It is a system to automate most of the tasks in a warehouse, with the objective of optimizing productivity ratios. When a product needs to be stored or retrieved in the automated system, it is moved to the point through horizontal and vertical carousels. Its design optimizes the use of space and knows exactly where each product is located.

The problem is the cost of implementation, since it cannot be adapted to a ready-made warehouse, but must be built in conjunction with the automated system. On the other hand, mobile robots have replaced long-distance transfers in warehouses and are much more versatile. They are also better for large loads, i.e., a certain minimum quantity must be stored as a package, since to handle a small object, a manipulator would be required.

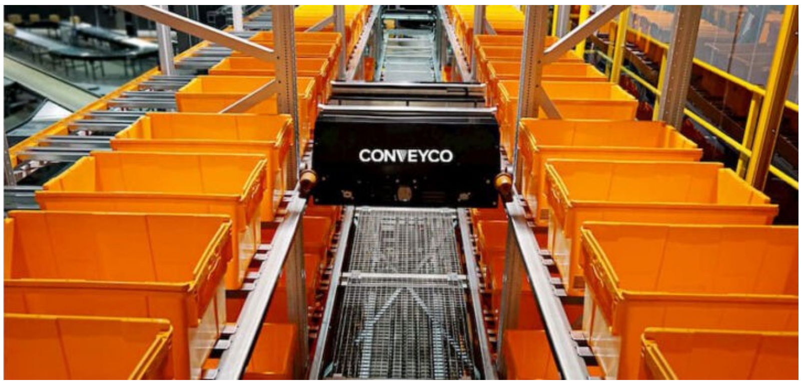
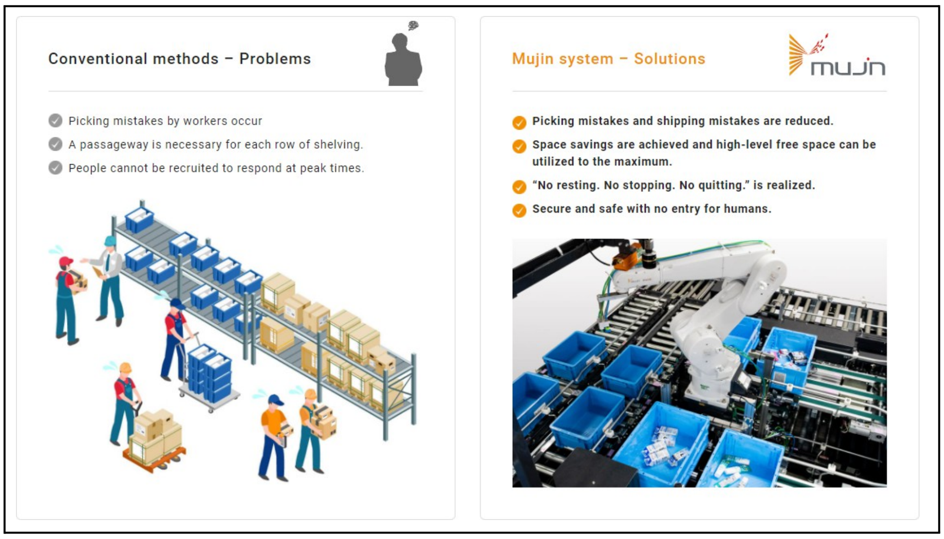


Figure 3. Automated storage and retrieval system [2].

2.2. Industrial robotic arms for warehouse automation

Industrial robotic arms are a great alternative today for warehouses, as they offer the possibility to do pick&place type tasks in a routine way and without the limitations that a person may have such as fatigue, risk of accident and inventory control.

One problem is that robotic arms in high-bay warehouses must be used in conjunction with other types of robots such as mobile robots in order to move products to other points. In the case of high-bay warehouses, they cannot reach these points on their own, unless it is a fairly large robotic arm, which can require high acquisition and maintenance costs.



Conventional methods – Problems

- Picking mistakes by workers occur
- A passageway is necessary for each row of shelving.
- People cannot be recruited to respond at peak times.

Mujin system – Solutions

- Picking mistakes and shipping mistakes are reduced.
- Space savings are achieved and high-level free space can be utilized to the maximum.
- "No resting. No stopping. No quitting." is realized.
- Secure and safe with no entry for humans.

Comparison chart between conventional work in warehouses and the use of robotic arms - Mujin Systems [3].

2.3. Computer vision to automate warehouses

Computer vision helps the robot to obtain the estimate of the position of objects in space and is something that has started to be used in warehouses, to reduce human intervention.

In 2016, Amazon launched a competition called "Amazon Robotics Challenge", where researchers from around the world tried to solve problems to automate warehouses through pick and place robots. Figure 2 shows a solution from the MIT-Princenton team, where they have a robot with 5 degrees of freedom and a gripper that has an RGBD camera to recognize objects in space and estimate their position through advanced algorithms. This solution is quite applicable to objects that are not structured; that is, you have different shapes and different colors on a shelf and you have to differentiate between them. But we know that warehouses, especially large-scale warehouses, orderly place one type of product on each shelf. On the other hand, this solution does not take into account the fact that warehouses are very high, for example those in the medical industry, so systems are required that can reach these difficult points.

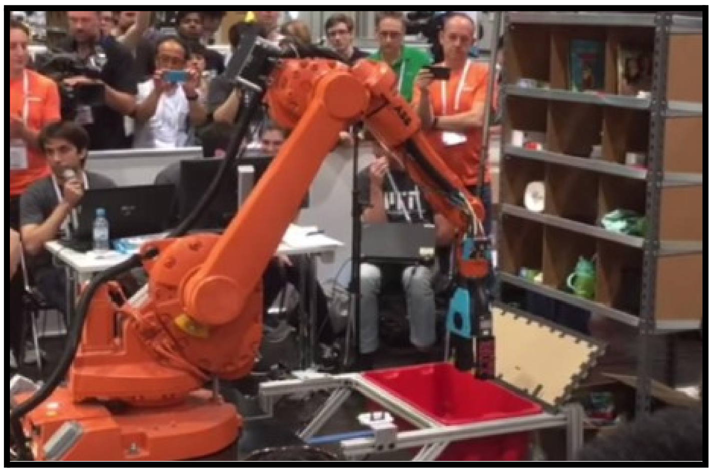


Figure 5. pick&place system of the MIT-Princenton team in the Amazon Robotics Challenge 2017 [4].

3. Proposal

Considering the background, it is proposed to use the manipulation advantages of robotic arms, as well as the versatility to reach complicated ASRS locations. Structured storage within the shelves is assumed, so that computer vision is not required to obtain the object position.

The proposed system is a combination of a Cartesian robot and an articulated robot, with a total of 7 degrees of freedom. Two degrees of freedom will be used for the displacement of the robot in the plane of a warehouse shelf (X,Z) and 5 degrees of freedom will be for the manipulation of the objects on the shelf.

The robot will first position itself close to one of the shelf cabinets containing the boxes. With the gripper, it will unfold a cabinet outward and pull out one of the products inside the box and return the cabinet inward. Finally, the robot will dispatch the box, heading to the customer's panel.

This system works completely without the assistance of any operator. The only thing your operators are in charge of is programming and maintenance. In addition to that, your customers can interact with the system using the self-service front panel. The customer can go to the panel, choose the desired product and the system does all the work behind the scenes: finding the product, dispensing and updating the inventory in real time.

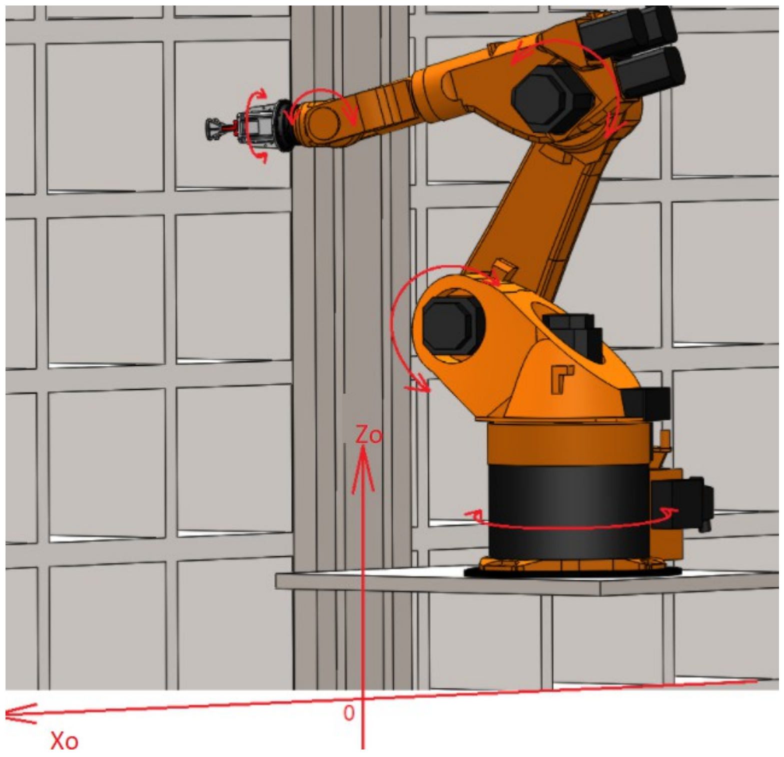


Figure 6. Proposed storage system with a 7GL robot.

4. Target

The main objective of this project is to find a possible solution to the problem of storage and management of products in a warehouse, by automating the process using a robotic manipulator with 7 degrees of freedom.

5. Robot structure

a) Definition of structure and parameters Denavit Hartenberg

Taking the image above as a reference, the diagram of the joints and axes was made in order to define the Denavit-Hartenberg parameters. The variables with the symbol "*" are those that will vary in the system.

The system has the first two joints as prismatic joints with which the robot will move through X and Z to be able to move along the shelf, as described in the description of the robot. Likewise, in order to be able to perform the maneuvers mentioned in the description and also shown in Figures 4 and 5, the rotational joints 3,4,5,6 and 7 are considered.

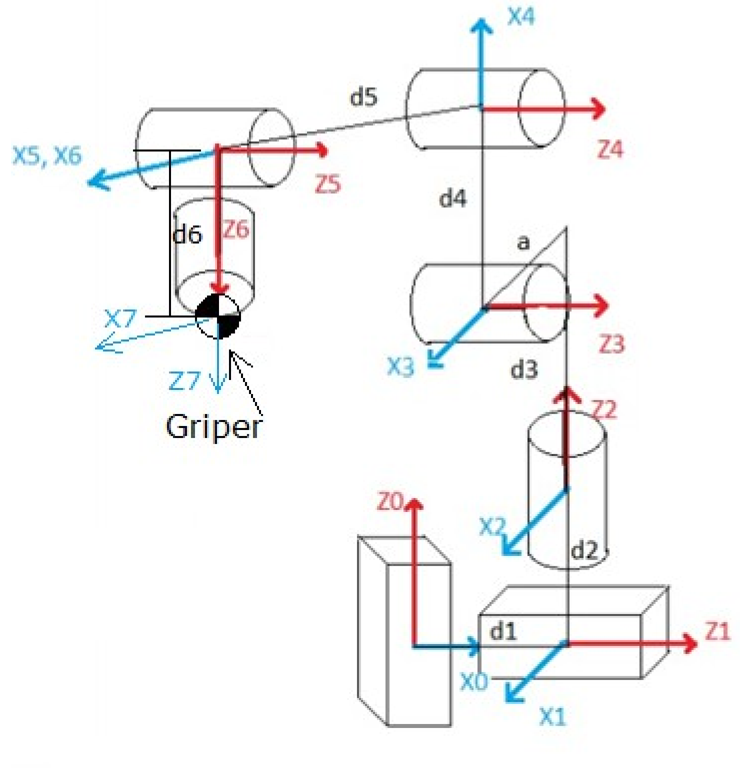


Figure 7. Automatic mechanism of 7GL

Table 1. DH - 7GL

Articulaciones	θ	d	a	α
1	-90	d1	0	-90°
2	0°	d2	0	90°
3	θ3*	d3	a	-90°
4	θ4*-90	d4	0	0
5	θ5*+90	d5	0	0
6	θ6*	0	0	-90°
7	θ7*	d6	0	0

b) Robot dimensions

The **shelf** dimension is 3.5m wide x 3.5m long. These would be the dimensions for the first two degrees of freedom of the robot. Therefore:

- d1* is 3.5m maximum
- d2* is 3.5m maximum

To define the size of the remaining links, it is necessary to define the dimensions of the boxes.

- The boxes are 300 mm wide x 250 mm high with a thickness of 50 mm, in these boxes objects are stored.
- The cabinets are 350 mm wide, including the dimensions of the slides, 350 mm high and their maximum opening length is 500 mm.

Figure 4 shows schematically the robot picking up the box and removing it from the cabinet. Figure 5 shows a top view where the robot removes the sliding cabinet by 50 mm and then returns it.

The following can be obtained from Figures 4 and 5:

d4 = 250 mm + 100 mm = 350 mm

d3 = 100 mm + 350 mm = 450 mm

a = 100 mm

$$d5 = \sqrt{(175 + 100 + 175)^2 + 250^2)} - 100 = 415mm$$

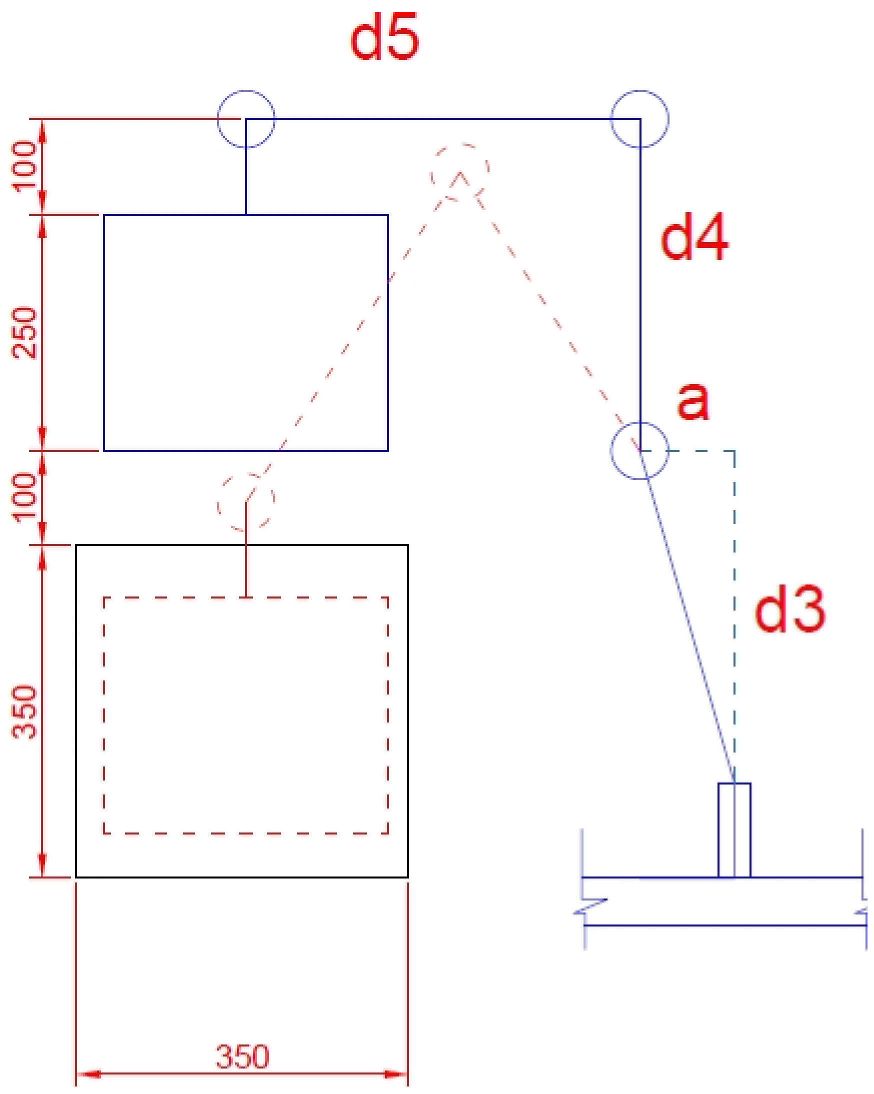


Figure 8. Robot elevation view.

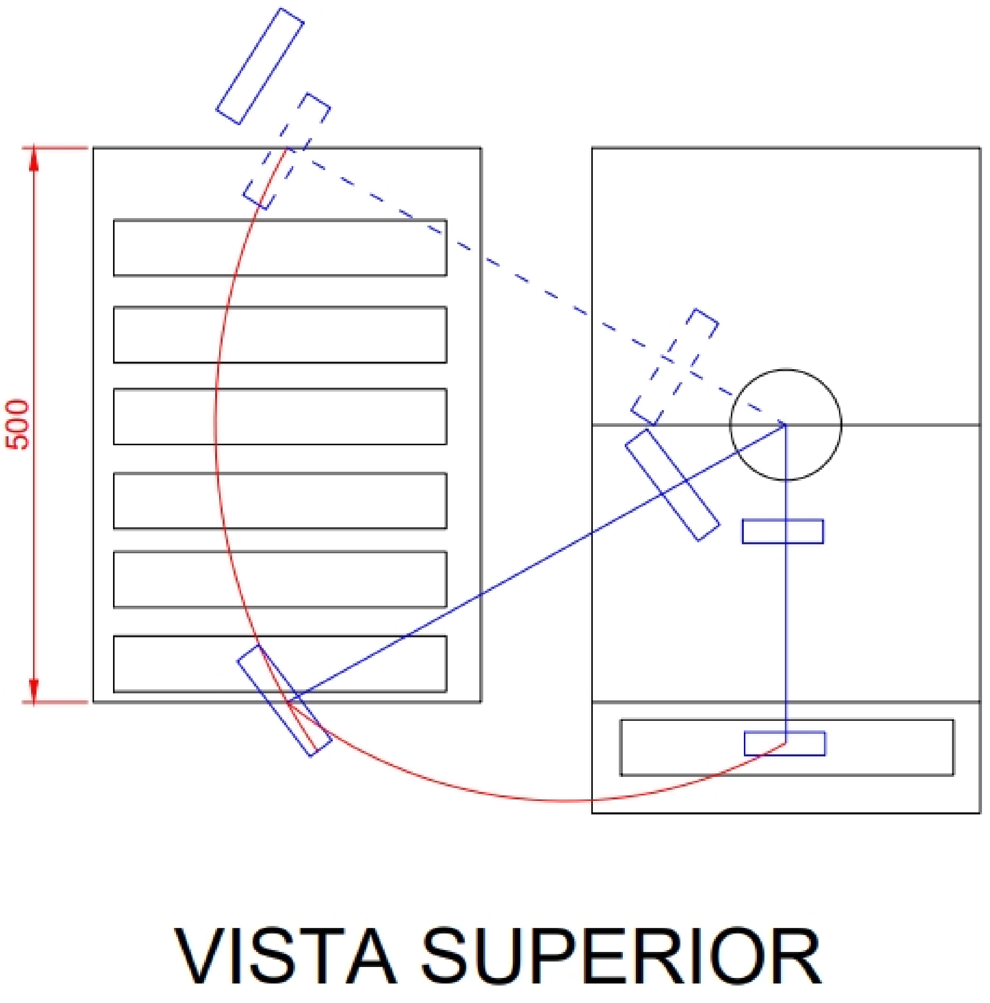


Figure 9. Top view of the robot

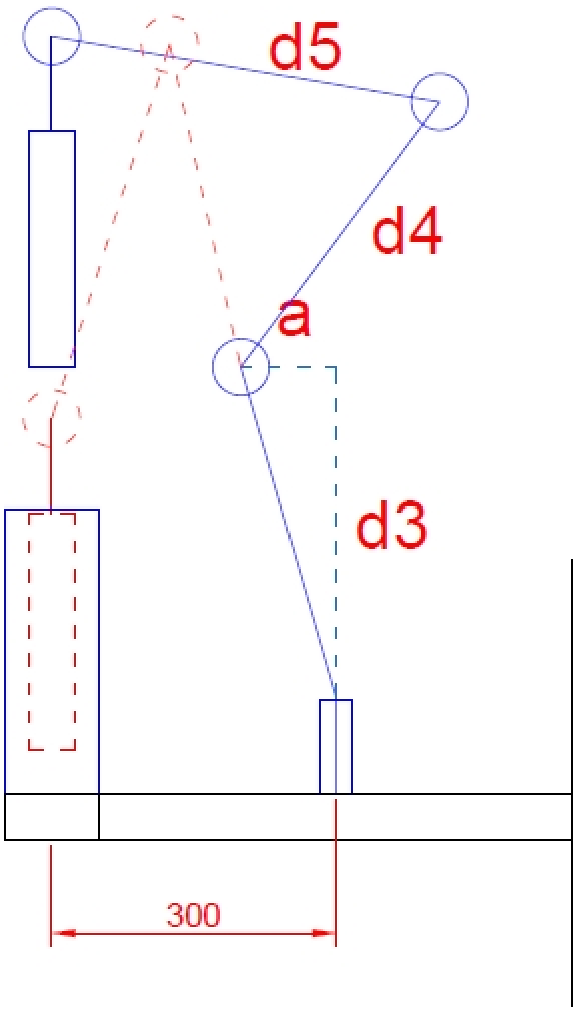


Figure 10. Elevation view 2

6. Modeling

The robot has 7 degrees of freedom. Two of them are prismatic and 5 are rotational degrees of freedom. Solidworks and VREP have been used for the modeling of the robot.

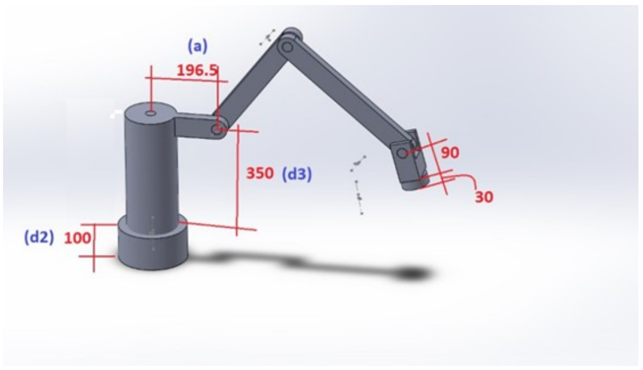


Figure 11 - Robot modeling in Solidworks

7. Positioning tracking and trajectory tracking control

7.1. Position and trajectory control

Control in operational space based on the kinematic model is used. For this, inverse kinematics has been used to obtain the desired position in joint space. For trajectory control, the pseudo inverse is used to have the desired velocity in the joint space.

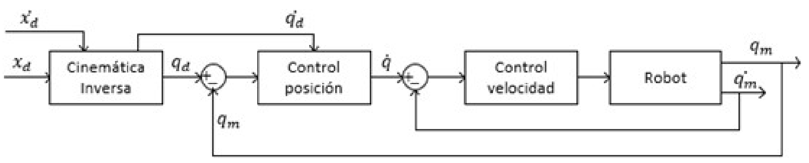


Figure 12 - Trajectory control in the operational space based on the kinematic model

Control de posicionamiento

$$\dot{q} = K_p(q_d - q_m)$$

Control de seguimiento de trayectoria cinemática

$$J_a^{+} = J_a^{T}\left(J_a J_a^{T}\right)^{-1}$$

$$c = J_a^{+}(\dot{x}_d)$$

$$\dot{q} = K_p(q_d - q_m) + \dot{q}_d$$

Below is the Matlab code, which represents what is shown in Figure 12. The *inverse_robot()* function gives us the desired joint position and the analytical Jacobian that will help us to obtain the desired joint velocity.

```
[qd,Ja]=inverse_robot(xd);

JaT = Ja'*inv(Ja*Ja');

qd_p = JaT*(xd_p);

q_p = Kp*(qd-q) + qd_p' = Kp*(qd-q) + qd_p'.
```

7.2. Inverse kinematics

For the manipulator, the inverse kinematics is performed, taking into consideration that z5 is always equal to -z0, since this is enough to perform the pick and place tasks for this specific case. In Figure 13, the initial position of the manipulator is shown. First, we separate the inverse kinematics of position and orientation, defining the wrist center (WC) at Z3. So we have Xc = Xo, Yc = Xo, Zc = d5 + Zo.

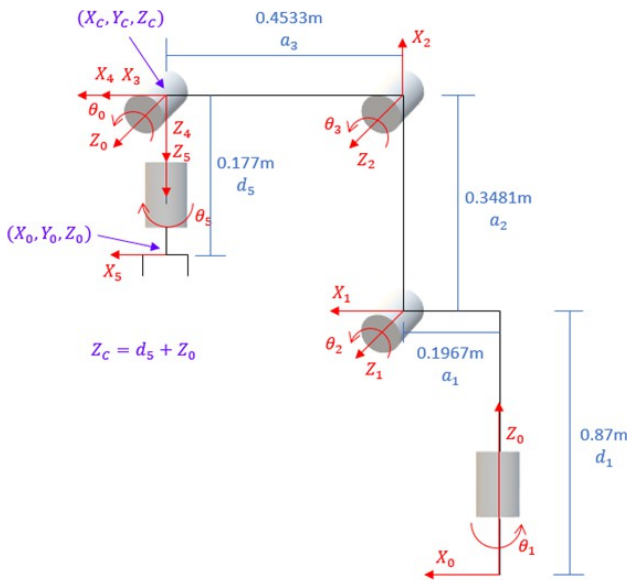


Figure 13 - Obtaining the WC

Figure 14 shows the calculation of teta 1, which will only depend on xc and yc.

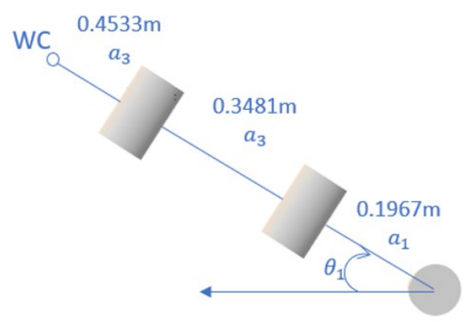


Figura 1

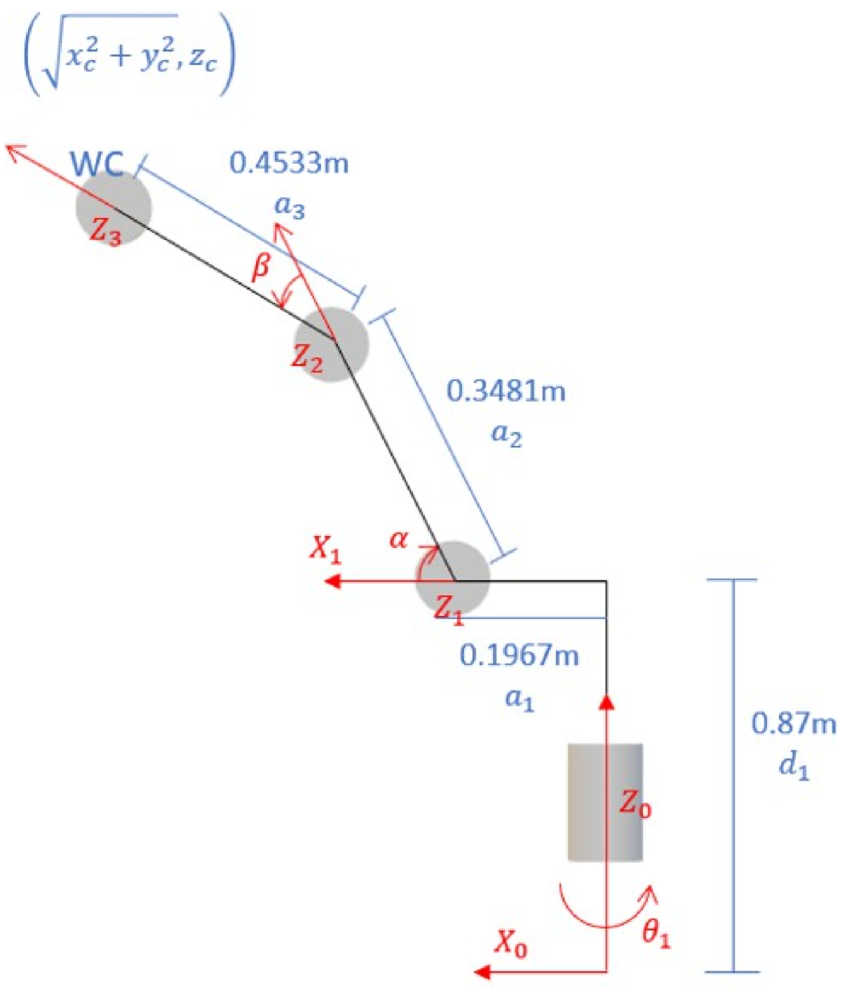
$$\theta_1 = ata2(y_c, x_c)$$

Figure 14 - Calculation of teta1

The Matlab code for obtaining teta 1 is shown below.

```
function [q_d,Ja]=inverse_robot(xd)

Orientation to be achieved in the XY plane

gamma = xd(4);

Position to reach O

=[xd(1);xd(2);xd(3)];

%Calculation of Oc

Oc = [O(1),O(2),O(3) + d5].

%Inverse of position

t1 = atan2(Oc(2),Oc(1));

t1g=t1*180/pi;
```

In Figure 15, the calculations for the angles teta 2 and teta3 are shown. First the angles are calculated by geometry and these angles are related to teta 2 and teta 3.



$$\cos(\beta) = D = \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2a_3}$$

$$r = \sqrt{x_c^2 + y_c^2} - a_1$$

$$s = z_c - d_1$$

$$\beta = atan2\left(D, \pm\sqrt{1 - D^2}\right)$$

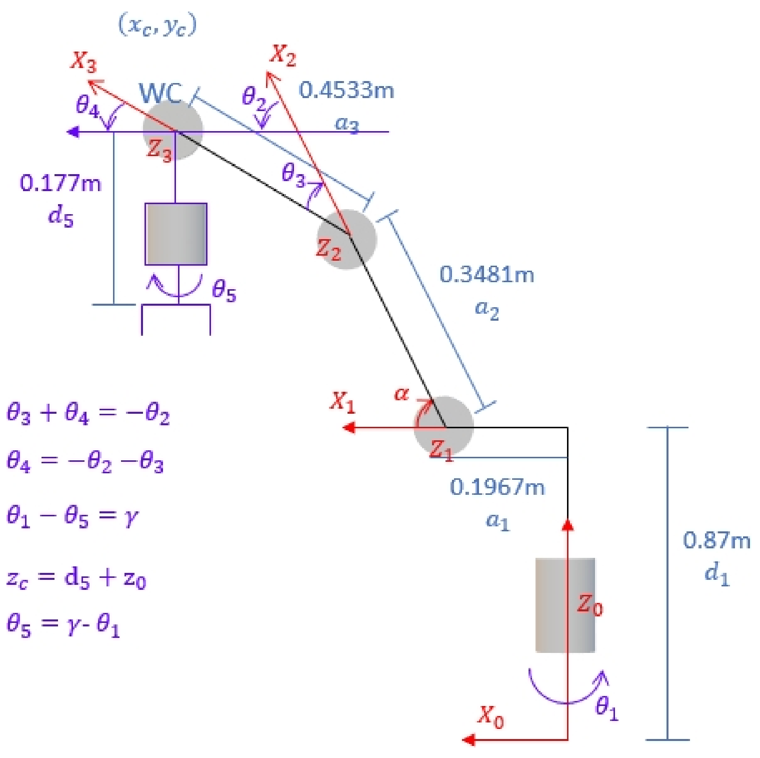
$$\alpha = atan2(r, s) - atan2(a_2 + a_3C_3, a_3S_3)$$

Figure 15 - Calculation of teta2 and teta 3

The Matlab code for obtaining teta 2 and teta 3 is shown below. These do not coincide with teta 2 and teta 3, so the conversion is performed, considering the corresponding rotations.

```
r=sqrt(Oc(1)^2+Oc(2)^2)-a1;

s=Oc(3)-d1;

D = (r^2+s^2-a2^2-a3^2)/(2*a2*a3)

beta=atan2(-sqrt(1-D^2) , D);

alpha=(atan2(s,r)-atan2(a3*sin(beta) , a2+a3*cos(beta)));

alphag=alpha*180/pi;

betag=beta*180/pi;

q2=90-alphag;

q3=-betag-90;

t2g = (q2-90);

t3g = (q3 +90);
```

Finally, teta 4 and teta 5 are obtained. Teta 4 depends on teta 2 and teta 3m.



$$\theta_3 + \theta_4 = -\theta_2$$

$$\theta_4 = -\theta_2 - \theta_3$$

$$\theta_1 - \theta_5 = \gamma$$

$$z_c = d_5 + z_0$$

$$\theta_5 = \gamma - \theta_1$$

figure 16. theta and theta 5 calculation

8. Robot simulation

The robot modeled in VREP has been connected to Matlab in order to perform a simulation of the movement. The sequence of movements of the robot is defined below.

- Starting position

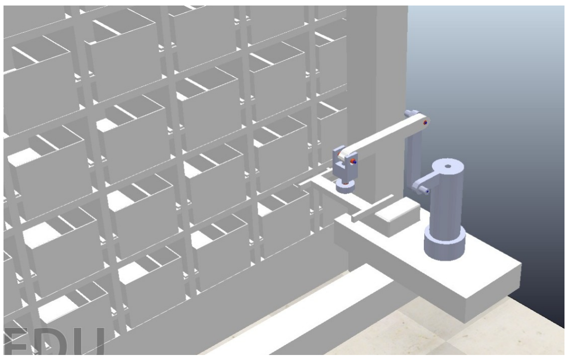


Figure 17. Initial position of the manipulator

- Shelf positioning

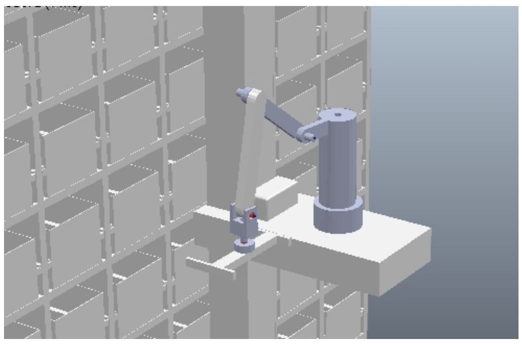


Figure18. Positioning on the shelf

- Positioning for drawer removal

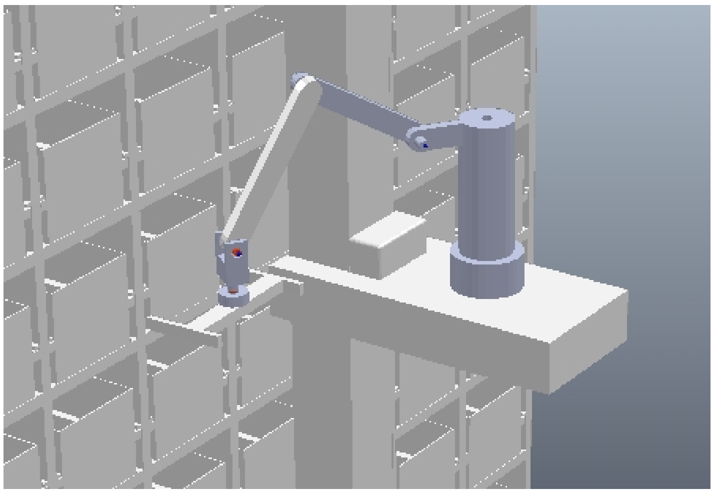


Figure19. Positioning for drawer removal

- Positioning to remove the drawer

  At this stage it remains to place a grip on the box so that the claw can pull. For now, the simulation is done in a representative way.

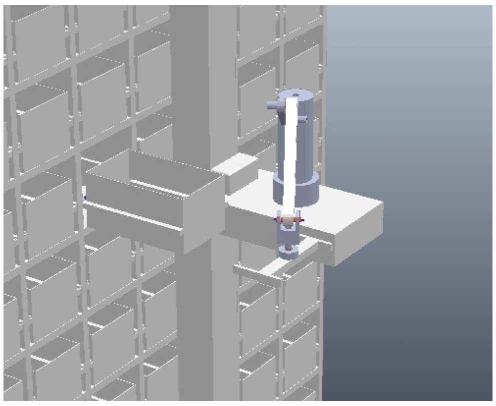


Figure20. Positioning for drawer removal

- Positioning to remove the object

  This stage is yet to be completed as it also requires a modeling of the drawer to be able to position the objects inside and remove it. A reference image of how the robot would remove the object is shown.

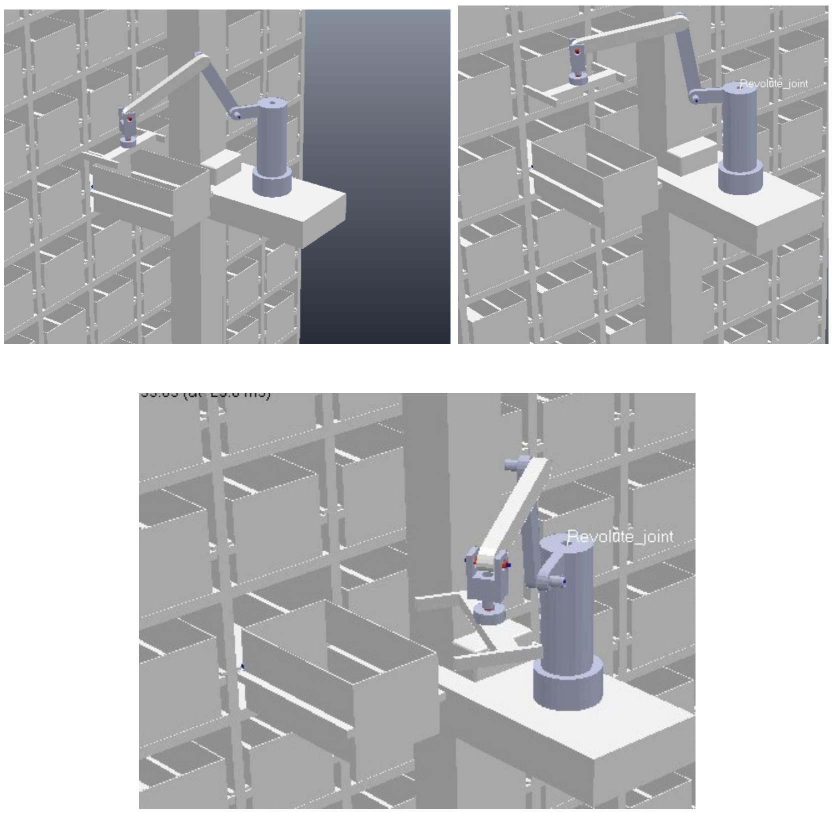


Figure 21. Positioning for object removal

- Positioning to return the drawer in position

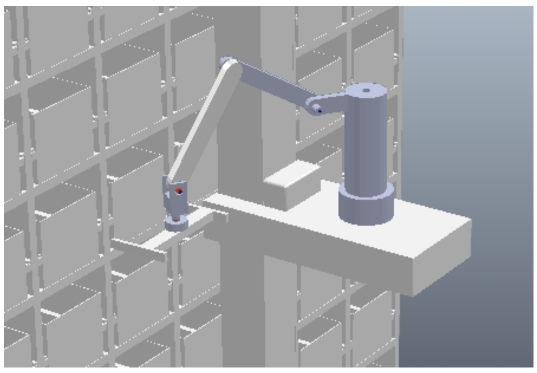


Figure 22. Pocicionamiento to return the drawer to its initial position.

- Return

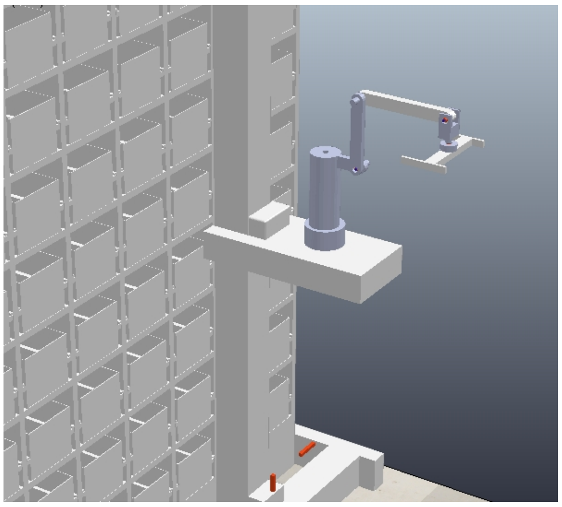


Figure 23. Manipulator return

The Matlab code used for the connection to VREP can be found in Appendix 1.

9. Bibliography

[1] Mecalux Esmena, Pharmaceutical logistics: radiography and challenges of the sector, January 2, 2020. Retrieved from: https://www.mecalux.es/blog/logistica-farmaceutica

[2] What is an automated storage and retrieval system?, WILL ALLEN, May 2021, Retrieved from
from:
https://6river.com/what-is-an-automated-storage-and-retrieval-system/

[3] Piece and picking, Mujin Technologies, 2021, Retrieved from: https://www.mujin.co.jp/en/solution/distribution/picking/

[4] Andy Zeng, Kuan-Ting Yu, Multi-view Self-supervised Deep Learning for 6D Pose Estimation in the Amazon Picking Challenge, Massachusetts Institute of Technology, May 2017.

10. Annexes

Annex 1 - Matlab code

```
Make sure to have the server side running in V-REP:

% in a child script of a V-REP scene, add following command

% to be executed just once, at simulation start:

%

% simRemoteApi.start(19999)

%

% then start simulation, and run this program.

%

% IMPORTANT: for each successful call to simxStart, there

% should be a corresponding call to simxFinish at the end!


clc

clear all

disp('Program started');

vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)

vrep.simxFinish(-1); % just in case, close all opened connections

clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);


if (clientID>-1)

       disp('Connected to remote API server');



[returnCode,handle1]=vrep.simxGetObjectHandle(clientID,'Revolute_joint',vrep.simx_o
pmode_blocking );
```

```
[returnCode,handle2]=vrep.simxGetObjectHandle(clientID,'Revolute_joint0',vrep.simx_
opmode_blocking );

[returnCode,handle3]=vrep.simxGetObjectHandle(clientID,'Revolute_joint1',vrep.simx_
opmode_blocking );

[returnCode,handle4]=vrep.simxGetObjectHandle(clientID,'Revolute_joint2',vrep.simx_
opmode_blocking );

[returnCode,handle5]=vrep.simxGetObjectHandle(clientID,'Revolute_joint3',vrep.simx_
opmode_blocking );

[returnCode,handle6]=vrep.simxGetObjectHandle(clientID,'Prismatic_joint',vrep.simx_
opmode_blocking );

[returnCode,handle7]=vrep.simxGetObjectHandle(clientID,'Prismatic_joint0',vrep.simx
_opmode_blocking );

[returnCode,handle8]=vrep.simxGetObjectHandle(clientID,'Prismatic_joint1',vrep.simx
_opmode_blocking );




[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-100*pi/180,vrep.simx
_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,50*pi/180,vrep.simx_o
pmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,30*pi/180,vrep.simx_o
pmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,-80*pi/180,vrep.simx_
opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-99*pi/180,vrep.simx_
opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle6,1.1,vrep.simx_opmode_
blocking);
```

```
[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle7,1.89,vrep.simx_opmode
_blocking);

       pause(4);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-125*pi/180,vrep.simx
_opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,60*pi/180,vrep.simx_o
pmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,1*pi/180,vrep.simx_op
mode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,-60*pi/180,vrep.simx_
opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-125*pi/180,vrep.simx
_opmode_blocking);

       pause(5);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle8,-0.6,vrep.simx_opmode
_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-60*pi/180,vrep.simx_
opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-60*pi/180,vrep.simx_
opmode_blocking);

       %here

       pause(4);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-125*pi/180,vrep.simx
_opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,30*pi/180,vrep.simx_o
pmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,1*pi/180,vrep.simx_op
mode_blocking);
```

```
[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,-30*pi/180,vrep.simx_
opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-125*pi/180,vrep.simx
_opmode_blocking);

pause(4);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-125*pi/180,vrep.simx
_opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,10*pi/180,vrep.simx_o
pmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,1*pi/180,vrep.simx_op
mode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,-10*pi/180,vrep.simx_
opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-125*pi/180,vrep.simx
_opmode_blocking);

pause(2);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-60*pi/180,vrep.simx_
opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,10*pi/180,vrep.simx_o
pmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,1*pi/180,vrep.simx_op
mode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,-10*pi/180,vrep.simx_
opmode_blocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-125*pi/180,vrep.simx
_opmode_blocking);



        pause(2);
```

```
[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-60*pi/180,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,60*pi/180,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,1*pi/180,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,-60*pi/180,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-60*pi/180,vrep.simx_opmode_blocking);


%here

       pause(4);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle8,0,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,-125*pi/180,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,-125*pi/180,vrep.simx_opmode_blocking);

       pause(4);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle1,0,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle2,0,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle3,0,vrep.simx_opmode_blocking);

[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle4,0,vrep.simx_opmode_blocking);
```

```
[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle5,0,vrep.simx_opmode_bl
ocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle6,0,vrep.simx_opmode_bl
ocking);


[returnCode]=vrep.simxSetJointTargetPosition(clientID,handle7,0,vrep.simx_opmode_bl
ocking);

        pause(6);


        % Now send some data to V-REP in a non-blocking fashion:

    % vrep.simxAddStatusbarMessage(clientID,'Hello
V-REP!',vrep.simx_opmode_oneshot);


        Before closing the connection to V-REP, make sure that the last command sent
out had time to arrive. You can guarantee this with (for example):

        vrep.simxGetPingTime(clientID);


        % Now close the connection to V-REP:

        vrep.simxFinish(clientID);

else

        disp('Failed connecting to remote API server');

end

vrep.delete(); % call the destructor!


disp('Program ended');

%end
```